TELA : Structure et algorithmes pour la traduction fondée sur la mémoire

Emmanuel Planas

THESE
présentée et soutenue publiquement par

Emmanuel Planas

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Spécialité
INFORMATIQUE

UNIVERSITE GRENOBLE 1
U.F.R. MATHEMATIQUES APPLIQUEES ET INFORMATIQUE

TELA
Structures et algorithmes pour la Traduction Fondée sur la Mémoire

---

Date de soutenance

7 Juillet 1998

Jury :

Président :	Yves Chiaramella
Directeur :	Christian Boitet
Rapporteurs :	Michel Divay
	Christian Fluhr
	Jean Véronis,
	Eric Wehrli
Examinateur :	Mathieu Lafourcade

THESE PREPAREE AU SEIN DU LABORATOIRE GETA (IMAG, UJF & CNRS)
ET A L'UNIVERSITE DES NATIONS UNIES DE TOKYO

Introduction

1.1 Situation, Motivations

1.1.1 Situation Industrielle

Le marché de la traduction industrielle, qui représentait déjà vingt milliards de dollars en 1997[1], continue de progresser fortement au niveau international. La demande en traduction automatisée augmente de façon encore plus marquée car les industriels cherchent à traduire toujours plus et toujours moins cher. Or il se trouve que certains documents consistent en des textes ou des hypertextes dont le contenu, s'il n'est pas purement recopié, est fortement inspiré de documents précédents, ou donne lieu à plusieurs versions similaires. Les manuels d'utilisation de logiciels (de plus en plus souvent en ligne) en sont un exemple classique. D'autres types de documents consistent en l'utilisation systématique de phrases modèles, comme les contrats d'assurances.

Pour apporter une première solution à l'automatisation de la traduction de ce type de documents, un certain nombre d'outils d'aide à la traduction humaine ont vu le jour. Parmi ces outils, certains sont fondés sur le stockage et la réutilisation de traductions déjà effectuées. On les appelle Outils d'aide à la Traduction Fondée sur la Mémoire (OTFM). Certains restent confidentiels ou utilisés seulement en interne (Localix d'ILE, outil de Oki Electric), d'autres sont à faible diffusion (DéjàVu d'Atril), mais au moins trois d'entre eux ont eu un succès important (IBM Translation Manager, Star Transit, Trados Workbench). On peut voir un signe de l'importance stratégique de ces outils dans le passage, opéré en 1997 par Microsoft, à une utilisation systématique de Trados Workbench, pour la traduction de ses logiciels et des manuels associés, passage qui s'est accompagné par une prise de capital de 30% de la société Trados qui produit ce logiciel. Un autre signe est la décision de Xerox de lancer sur le marché un nouveau produit de ce type, en juin 1998.

Si l'intérêt économique de ce genre d'outils semble acquis, les efforts menés pour leur développement restent toutefois réduits. Hormis les programmes généraux du type d'OTELO de la Communauté Européenne et l'effort récent de Xerox, bien peu d'outils réellement efficaces sont disponibles, et l'on peut par exemple s'étonner qu'il n'existe pas de produits de ce type au Japon, à part quelques essais internes par exemple d'Oki Electric à Osaka [Sukehiro 1995]. Cela laisse d'ailleurs le marché à deux sociétés allemandes, STAR qui distribue Transit et qui est installée à Tokyo depuis quelques années, et TRADOS qui est venue concurrencer l'année dernière la première avec le logiciel Workbench.

[1] Source : Multilingual Computing, n° 12.

Ces outils sont pourtant loin d'être parfaits, bien qu'ils permettent des réductions de coût appréciables (de 20 à 80% selon les éditeurs, de 5 à 60% dans la réalité, car on doit prendre en compte la mise en place et la maintenance par des ingénieurs technico-linguistes par exemple). On trouvera plus bas plus de précisions sur leurs performances. Ces outils permettent et ont permis l'accumulation d'un nombre important de corpora de segments bilingues alignés qui fournissent une source d'information importante et souvent très adaptée au type de documents traités, puisque collectée (de façon automatique ou semi-automatique) lors du travail des traducteurs sur les mêmes types de documents.

1.1.2 Situation technologique

La technologie des trois produits principaux du marché actuel repose, d'une part sur une base de données indexée d'unités de mémoires, et d'autre part sur des algorithmes de recherche de similitudes utilisant en général des fonctions de coût ou des statistiques. La représentation interne de ces outils, suivant un "langage" interne (proche de SGML dans la plupart des cas) permet de conserver l'ensemble des données sous ce même formalisme. Les algorithmes de recherche s'appuient sur cette représentation interne. Si des analyseurs linguistiques sont utilisés, c'est en général pour la consultation de la base terminologique, mais pas pour la recherche dans la base phraséologique.

Les propositions d'unités de traduction dépendent donc souvent du type de document traité, et toujours de leur format. Cela provient essentiellement du fait que les recherches s'effectuent sur la représentation interne qui est hétérogène et correspond plus ou moins fidèlement au code trouvé dans les documents, rendant les mémoires dépendantes du format du document. Cela est par exemple totalement vrai dans Trados Workbench, mais moins dans IBM Translation Manager qui utilise un vrai format interne. Il est de toutes façons en général très difficile d'utiliser une mémoire de traduction faite sur un document possédant un certain format, pour traduire un document sous un autre format, même si les phrases sont linguistiquement comparables ou égales.

Ces outils posent de sérieux problèmes au niveau de la gestion et du transfert de la mise en forme du texte et des objets non linguistiques comme le prouve le test du chapitre 3 de ce document. Dans la pratique, il n'est possible de récupérer que les formats qui ont été enregistrés en mémoire, et qui dépendent donc souvent du document d'origine. Les objets non linguistiques (les images, les index, etc.) sont au mieux reconnus et proposés au traducteur en mode interactif, et ce sont les traducteurs eux-mêmes qui les placent. En aucun cas, ils ne sont insérés automatiquement au bon endroit par l'outil.

Leurs interfaces sont pourtant souvent bien conçues et offrent de nombreuses fonctionnalités intéressantes comme le défilement parallèle des phrases source et cible, un système de fenêtrage affichant le texte à traduire, les unités de mémoire proposées, des dictionnaires, un accès direct à une base terminologique et autres renseignements utiles au travail du traducteur.

1.1.3 Situation au niveau de la recherche

Ces outils manquent surtout d'une formalisation de leur comportement et de leur fondements théoriques. A part quelques articles dispersés comme [Langé, Gaussier & Daille 1997, Witkam 1988, ou Kugler 1991] qui ne sont pas très techniques, il n'existe pas d'études présentant le domaine de façon précise.

Il existe cependant quelques approches connexes intéressantes, qui n'ont pas donné de système commercial, mais au moins une maquette ou des essais, comme l'application du raisonnement fondé sur les cas [Collins & Cunningham 1996] ou l'application de méthodes statistiques comme par exemple dans [Doi & Muraki 1993], [Hai, Kawtrakul & Poovorawan], [Chen & Chen], [Kitano 1993] ou [Li & Choi 1997].

Les outils d'aide à la traduction fondée sur la mémoire ont a priori pour but de proposer des traductions qui sont d'origine strictement humaine. Il est dans ce cas essentiel de ne pas introduire du "bruit" provenant de la machine, de façon que l'utilisateur soit sûr d'être en face d'une solution qui provient d'un humain, et donc en laquelle on peut avoir confiance. Pourtant, les systèmes actuels ne permettent pas, ou permettent mal, de transférer le format ou les éléments non linguistiques, comme nous l'avons déjà dit. Or cette tâche est justement lourde pour les traducteurs humains. En outre, l'observation du comportement de ces outils à grande échelle montre qu'une partie du silence observé pourrait être réduit par une génération simplement morphologique, même très faible (passage à un pluriel par exemple), ou une composition d'unités de traduction présentes en mémoire. Nous explorons ces axes dans cette étude.

Parallèlement, de nombreuses recherches existent dans un domaine proche qui est la Traduction Automatique Fondée sur l'Exemple (Example-based Machine Translation) ou encore Traduction Automatique Fondée sur l'Analogie [Analogy-based Machine Translation]. Depuis le lancement du concept par [Nagao 1984], de nombreuses études ont été menées sur l'utilisation de modèles appelés "exemples" [Nagao 1984] et [Sato 1989], ou "patrons" [Sadler 1989], [Sumita & Iida 1992], [Furuse & Iida 1994, 1996], [Mima 1997], ou encore sur des approches statistiques [Brown 1997].

Mais cette approche impose d'une part des temps de calculs importants qui en viennent à une approche fondée sur la programmation parallèle dans certains cas ([McLean 1992], [Sumida & Iida 1993]). Elle implique d'autre part de construire des structures linguistiques (arbres par exemple) de façon largement manuelle. Cela rend ces techniques difficiles à utiliser de façon automatique sur un simple corpus de segments sources et cibles alignés, qui est la source d'information de base pour la traduction fondée sur la mémoire.

1.2 Intérêt scientifique

1.2.1 Structuration des données dans un outil de traduction fondée sur la mémoire

Les données utilisées dans les outils de traduction fondée sur la mémoire sont généralement perçues et traitées comme un ensemble hétérogène. Cette disparité apparaît dans la représentation interne de ces outils, tant au niveau du texte à traduire, et des unités de mémoires, qu'au niveau de la représentation des données externes, linguistiques ou non linguistiques. Le premier apport de ce travail est de montrer que cette disparité existe de façon systématique dans les outils actuels.

Une première formalisation consiste à homogénéiser le flot de l'information en présentant une unité de traduction sous la forme structurée qu'est un segment XML. Dans un second temps, nous proposons de voir une unité de texte à traduire (nous parlerons de *segment*) comme une suite d'éléments de différentes natures. Parmi ces éléments, on trouve, entre autres, des caractères exprimés sous un codage et une transcription qu'il faudra éventuellement manipuler, des marques de mise en forme, et des objets non linguistiques comme des images ou des variables manipulées par le logiciel qui lit le document.

L'apport suivant consiste à proposer un éclatement de ces données suivant leur type en une structure plus forte : un ensemble de *treillis* disposés par *étages* (un étage par type de données) et connectés par des *liaisons*. Les caractères de la chaîne de l'expression à traduire, les mots eux-mêmes, les balises doubles et monobalises XML, les lemmes des mots rencontrés dans le segment, et d'autres types de données forment autant d'étages. Cette structure est appelée *TELA* pour Treillis Etagés et Liés pour le traitement Automatisé.

Nous montrons comment la manipulation de ces données, via cette structure, autorise non seulement les opérations classiques des outils de première génération avec plus d'efficacité

structurelle, mais ouvre aussi de nouvelles perspectives quant aux traitements linguistiques ou non linguistiques. Parmi les opérations classiques mieux réalisées figurent une meilleure recherche de similarité entre les chaînes, un meilleur support du transfert de la mise en forme du segment source au segment cible, et une ouverture vers une faible génération ainsi que la composition de mémoires de traduction.

En outre, la complexité dynamique des algorithmes de manipulation de la structure reste linéaire, offrant par là-même une efficacité procédurale non négligeable. La complexité statique restant faible, l'espace mémoire nécessaire à la manipulation des données est tout à fait raisonnable

1.2.2 Définition de la similitude entre deux segments exprimés dans une même langue

Le principe des outils de traduction fondée sur la mémoire est de proposer les unités de traductions stockées en mémoire les plus *proches* possible de l'unité à traduire. Mais que signifie donc *proche* ? Les outils de première génération se basent sur des fonctions de coût intuitives ou statistiques. La formalisation de cette notion n'est pas claire, et son efficacité dépend trop du type de texte traité et de paramètres "à régler".

Pour remédier à ce manque, nous proposons une définition claire de la similitude entre deux segments documentaires exprimés dans une même langue, qui se base sur la notion de distance d'édition introduite dans [Wagner & Fischer 1974]. La distance précise alors la notion de similitude selon les types de données portées par les différents étages d'une structure TELA. L'utilisation de cette distance permet alors une recherche robuste et efficace des unités de mémoires candidates pour la traduction du segment d'entrée. Un pas dans cette direction a été fait dans [Cranias, Papageorgiou et Piperitis 1997], qui proposent une distance d'édition sur un segment hétérogène composé de lemmes, catégories grammaticales, et "mots fonctionnels" (that, whom, whereas,..), mais sans donner une formalisation complète.

1.2.3 Définition de traductibilité entre une expression source et une expression cible

Les systèmes de première génération s'arrêtent à cette recherche de similarité. Il nous a semblé qu'on pourrait faire plus. Il a été pressenti, comme par exemple dans [Langé, Gaussier et Daille 1997], qu'une certaine *généricité* (voir ci-après) des unités de traduction devrait amener un silence moins important dans la recherche d'unités de traduction, mais surtout des propositions plus proches de la traduction recherchée.

Cela nous a mené à l'étude de la liaison existant entre un segment source et sa traduction. Nous parlerons de traductibilité d'un segment source par un segment cible. Pour préciser cette notion de façon formelle, une seconde distance est proposée. Cette distance est une fonction des distances associées à chacun des types de données, dont la plupart se fondent sur la distance d'édition de [Wagner & Fischer 1974].

1.2.4 Définition de l'analogie de deux unités de traduction

Une unité de traduction (S1, T1) comporte une partie source S1 et une partie cible T1. Sa traduction par une unité de traduction (S2, T2) provenant de la mémoire procède d'une certaine ressemblance entre les deux couples. Reprenant les travaux de [Lepage 1996, 1997, 1998] qui définissent la notion d'analogie entre deux couples d'une même langue, nous l'étendons à deux couples d'unités de traduction via les notions de similitude et de traductibilité.

1.2.5 Généricité des unités de traduction

La mise en œuvre de la notion d'analogie permet alors de proposer de façon "propre" des traductions approchées sur la base des unités de traduction de la mémoire. Parce que l'on maîtrise alors les différences et l'endroit où elles se situent entre l'unité modèle provenant de la mémoire et l'unité à traduire, il devient possible de générer une traduction adaptée à l'unité à traduire. En outre, nous montrons comment la structure TELA et les algorithmes associés permettent d'aller vers une composition des unités de traduction, qui se rapproche, au moins dans l'idée, de ce qui se fait (ou voudrait se faire) dans les travaux de Traduction Fondée sur l'Exemple ou l'Analogie (voir travaux cités plus haut). Trois heuristiques sont proposées pour :

- transférer le format du segment source à traduire vers le segment cible final
- autoriser une génération faible dans le cas où l'unité de traduction trouvée en mémoire est *très proche* de l'unité à traduire
- composer les unités de traduction via la notion d'expression pivot

1.3 Plan du document

1.3.1 Partie 1 : état de l'art

La première partie est composée de trois chapitres. Le premier chapitre rappelle ce qu'est un outil de traduction fondée sur la mémoire. Nous exposons ensuite le problème pratique de la traduction de documents dans l'industrie, en mettant l'accent sur les problèmes de base comme celui de la manipulation de plusieurs formats de fichiers, ou la présence dans les document d'autres données que des mots à traduire. Nous insistons sur une activité importante en traduction : la traduction de logiciels et de leur documentation (localisation). Le chapitre 1 se termine par une réflexion sur les gains de traduction que peuvent apporter de tels outils.

Le second chapitre présente dans le détail les trois outils de traduction fondée sur la mémoire les plus utilisés. On y trouve une description de leurs fonctionnalités et de leurs principes de fonctionnement. Les différences entre les trois outils, et leurs originalités sont soulignées, et il est montré comment les données sont codées dans le format interne.

Le troisième chapitre propose une évaluation de ces outils. Cette évaluation se base sur la traduction d'un fichier test au format RTF dans lequel sont rassemblés des problèmes qui ont trait à la traduction elle-même, à la gestion de la mise en page, et à celle des objets non linguistiques classiquement rencontrés dans un document. Un certain nombre de faiblesses sont alors relevées, et un retour vers la représentation interne des données montre qu'elles proviennent principalement de cette dernière.

1.3.2 Partie 2 : proposition d'une nouvelle structure de données : TELA

Ayant montré que les limites des outils de traduction fondée sur la mémoire de première génération (OTFM1g) proviennent en partie de la structure de représentation des données, nous proposons dans la partie 2 (qui commence par le chapitre 4) de construire une représentation qui leur soit plus adaptée. Ce sera une représentation en deux parties, mais reflétant des niveaux d'analyse différents : la représentation primaire sert à l'échange de ces données, et la représentation plus structurée à leur manipulation. Comme les données à représenter proviennent des documents à traduire, nous étudions les différents types de tels documents pour y trouver quels sont les types de données à représenter. Les formats d'échange et représentation de données textuelles au niveau général, puis entre système de traduction, et enfin dans les OTFM1g sont alors passés en revue. On rappelle ce qui y est codé, et comment. XML est alors proposé comme choix de la forme primaire. Pour choisir la forme

structurée, nous examinons les structures de données qui semblent adaptées. Le choix porte finalement sur un ensemble de treillis étagés, et connectés par des liaisons. Cette structure est dénommée TELA.

Le chapitre 5 s'attache à décrire par une approche empirique ce que doit contenir la structure TELA. On y détaille les différents étages de la structure en les affectant à un type particulier de données ; c'est ainsi que le premier étage est dédié aux caractères du segment à traduire, le second aux mots, et le cinquième aux lemmes des mots. La notion importante d'expression pivot est introduite à cette occasion : c'est le premier pas vers une généricité des outils de traduction fondée sur la mémoire.

Le chapitre 6 revient sur la définition de TELA et la fige mathématiquement. Une description informatique générale est ensuite proposée, permettant de clarifier l'utilisation des différentes parties de la structure, et ses opérations de base. Pour faciliter la manipulation, et l'échange de structures TELA, deux langages de description de TELA sont donnés : une description basée sur XML permettant de coder l'ensemble des deux parties de la structure sous le même formalisme, et une description par objets pour l'algorithmique de manipulation des données.

1.3.3 Partie 3 : algorithmes et utilisation de TELA

La partie 3 (chapitres 7, 8 et 9) propose des algorithmes adéquats pour manipuler les structures TELA d'une part, effectuer les calculs de similarité d'autre part, et introduit trois heuristiques de traduction.

Le chapitre 7 reprend la description par objets de TELA donnée au chapitre précédent, et l'implémente en montrant à quoi correspondent les champs des objets, en une structure adaptée à la traduction fondée sur la mémoire suivant les recommandations du chapitre 5. Cela donne une instance de TELA appelée TELAM. L'algorithmique de construction d'une structure TELAM à partir d'un segment XML y est décrite.

Le chapitre 8 introduit la notion abstraite de similitude et propose d'adopter une distance d'édition pour son fondement mathématique. Il est ensuite montré comment la similitude est adaptée à la recherche d'unités de traductions pour traduire un segment d'entrée, et quels sont les algorithmes nécessaires. Le procédé de recherche d'unités de mémoires par les algorithmes basés sur la notion de similitude est alors explicité, et l'on montre comment ces algorithmes s'appliquent aux données représentées par les différents étages d'une structure TELAM.

Le chapitre 9 achève la recherche d'un environnement complet et formalisé pour la traduction fondée sur la mémoire en introduisant le notion de traductibilité et montrant que, combinée avec celle de similitude, elle donne celle d'analogie qui formalise le processus de traduction fondée sur la mémoire. Les algorithmes nécessaires y sont décrits. Quelques heuristiques de traduction dont la génération à base d'expressions à cases et d'expressions pivot sont proposées pour augmenter la généricité du modèle. Enfin, quelques recommandations qui nous semblent essentielles sont proposées pour la construction d'un système complet de Traduction Fondée sur la Mémoire.

En conclusion nous rappelons le problème de départ, notre démarche, les solutions proposées, et les améliorations apportées au problème. Une ouverture vers la coopération avec d'autres paradigmes de traduction est énoncée. Enfin, nous réfléchissons sur la proximité et les liens entre la traduction fondée sur l'exemple, ou l'analogie, et la traduction fondée sur la mémoire. Les perspectives montrent quelles pourraient être les améliorations de notre modèle, et quels systèmes concrets il peut permettre de construire.

Table des matières

Chapitre 3 : Faiblesses et espoirs des outils de première génération

Partie II

Structures

Chapitre 4 : Fondements de la structure

Chapitre 5 : Définition empirique

Chapitre 6 : TELA : une formalisation des structures

Partie III

Algorithmes

Chapitre 7 : TELAM : spécialisation de TELA pour la Traduction Fondée sur la Mémoire

Chapitre 8 : Similitude

Chapitre 9 : Similitude, et Traduction Fondée sur la Mémoire

Annexes

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

Chapitre 7

Chapitre 8

Chapitre 9

TFM	Traduction Fondée sur la Mémoire
OTFM1g	Outils de Traduction Fondée sur la Mémoire de première génération
TM	Translation Manager (IBM©)
WB	Workbench (Trados©)
TR	Transit (STAR©)

2. Sociétés et Marques commerciales citées

Dans la suite, les sociétés et les marques commerciales (™) suivantes seront citées. Elles ne comporteront pas le signe "™" ou "©" ou "®".

Marques
Collins® on line™, Berlitz® Interpreter™, Correcteur 101™, Hugo Plus™, Cordial™, Larousse® Electronique™
ATAO™, Termex™
Spirit™
Systran®™, Logos®™, Power Translator™, Transend™, LMT™
Eurolang® Optimizer™ (LANT®), Transit™ (STAR®), Translation Manager™ (IBM®), Workbench™ (Trados®), ForgeinDesk™, PowerGlot™, Reword Studio™, GDK localization suite™ d'Accent®, LXEdit™ d'ILE®, Catalyst™ de Corel®, or OTM™ d'Oracle®, DéjaVu™ d'Atril®, et XL8™.
Trados® Multiterm™, STAR® TermStar™
STagger for Interleaf™, STagger for FrameMaker™
Microsoft® Excel™, Adobe® Page Maker™, Adobe® FrameMaker™, MIF,
Adobe® Illustrator™
Amipro™, Claris® Works™, Interleaf™, Latec™
Microsoft® HTML 2, 3, 4
Microsoft® Word™ 2, 6, 7, 8, RTF
Mosaic® HTML 2, 3, 4
Netscape® HTML 2, 3, 4
Nisus®Writer™, Quark Xpress™, Troff™, Ventura™, WordPerfect™, Works™, Write™
ILE® OpenTag™

IBM® BookMaster
IBM® OS2
Apple® Macintosh™
Apple® Macintosh™ Système 7™ et Système 8™
Microsoft® Windows 95™ et Windows NT™
ArborText®
Metal®

Sociétés

Adobe®
Apple®
Microsoft®
IBM®
ILE®
ITP®

Groupes d'intérêt, Associations

LISA (Localization Industry Standards Association)
OTELO (Open Translation Environment for Localization)
OSCAR (Open Standards for Container/Content Allowing Re-use)
TMX (Translation Memory eXchange)
SGML (Standard Generalized Markup Language)
TEI (Text Encoding Initiative)

Remerciements

Un jour, une grenouille et un scorpion se rencontrent sur la berge d'une rivière. Le scorpion entre en conversation avec le batracien et lui demande : "Grenouille, ma mie, je dois traverser cette rivière mais ne sais point nager. Pourrais-tu me porter sur ton dos, je t'en serais reconnaissant". La grenouille, tout en se méfiant de l'insecte, réplique : "Mais tu es un scorpion, et les scorpions possèdent un venin dangereux pour nous, tu me fais peur". Finalement le scorpion, beau parleur, convainc la grenouille. Et nos deux aventuriers de traverser la rivière.
Mais soudain, au plus fort du courant, le scorpion pique mortellement la grenouille. Alors, tout en s'ankylosant sous l'action du poison violant, la grenouille demande au scorpion : "mais pourquoi as-tu fais ça ? Nous allons mourir tous les deux maintenant !". Le scorpion, qui sent déjà l'eau envahir ses entrailles répond alors : " Je n'y peux rien, bonne Grenouille, il est dans ma nature de piquer...".

Proverbe populaire chinois

Les personnes qui m'ont aidé à mener à bien cette thèse tiennent à la fois de la grenouille et du scorpion. Leur caractère positif et leur compétence leur a permis de jouer le rôle du passeur, comme la grenouille. Mais je suis convaincu qu'ils tiennent aussi du scorpion : cela est ancré au plus profond d'eux, d'être ce qu'ils sont. C'est, je crois, en reconnaissant ceci en eux que je puis les remercier le mieux de m'avoir aidé à traverser ce fleuve torrentiel qu'est la rédaction d'une thèse.

Parmi les personnes auxquelles je pense, il y a notamment :

Mes parents, mon frère et bien sûr Mami et Ken qui ont bien voulu m'accorder un temps dont ils n'ont pas bénéficié,
Le Professeur Christian Boitet et mes collègues du GETA et du CLIPS,
Le Professeur Tarcisio Della Senta et l'Université des Nations Unies.

Chapitre 1

Introduction aux outils de Traduction Fondée sur la Mémoire

Contenu du chapitre

Introduction

Les outils de TFM justifient leur existence par le fait que, pour certains ensembles de documents, les mêmes phrases ou unités de texte se retrouvent au moins une fois d'un document à l'autre. Cette redondance**{**xe "redondance"**}** peut se situer au sein d'un même document (on parlera d'intra-redondance**{**xe "intra-redondance"**}**), ou d'un document à l'autre (on parle alors d'inter-redondance**{**xe "inter-redondance"**}**).

Dans l'industrie, il s'agit souvent de récupérer la traduction de la version précédente d'un document**{**xe "version d'un document"**}**, pour traduire une nouvelle version (passage de Microsoft Word 6 à Microsoft Word 7, par exemple). On peut d'ailleurs se demander si dans ce cas on ne ferait pas mieux de gérer correctement les révisions**{**xe "révisions"**}** entre versions.

Dans le cas où la mémoire est réellement volumineuse et si l'on se limite à un sous-ensemble de documents (les manuels de logiciels d'édition de Microsoft, par exemple), on peut essayer aussi de retrouver dans de nouveaux documents (il ne s'agit pas alors de versions d'un même document), des phrases précédemment traduites.

Les exemples qui suivent sont souvent empruntés à la traduction de logiciels car ce domaine se prête particulièrement bien à la traduction fondée sur la mémoire.

1. Introduction aux Outils de Traduction Automatique Fondés sur la Mémoire de première génération (OTFM1g)

1.1 Positionnement des Outils de Traduction Fondée sur la Mémoire par rapport au Traitement Automatique du Langage Naturel.

Le domaine du Traitement Automatique du Langage Naturel (TALN) comprend maints domaines et préoccupations, couvrant depuis le traitement de la parole jusqu'au traitement de documents écrits. C'est dans la deuxième partie que se situe notre travail. Le traitement des documents écrits est cependant encore très vaste, et l'on peut préciser le positionnement de notre domaine en le classant dans la Traduction Assistée par Ordinateur (TAO).

La TAO peut être elle-même découpée selon plusieurs axes : les techniques utilisées, les théories linguistiques, ou encore l'application que l'on fait du système développé. Christian Boitet [Boitet 93, 94b, 94c, 94d] nous a souvent rappelé qu'il existe quatre types d'application de la TAO :

La traduction du veilleur

Il s'agit de comprendre la nature de ce qui se trouve dans le document à traduire. On veut en savoir assez pour classifier le document, comprendre ses idées générales, ou décider s'il est pertinent, sans pour autant chercher à obtenir une traduction de qualité. Ce type de traduction est utile pour les organismes de renseignements militaires ou politiques par exemple (voir les premiers projets de traduction des années 50), ceux de veille économique ou scientifique (service JapInfo par exemple), ou pour la recherche d'information ; c'est ainsi que les navigateurs pour Internet proposent de plus en plus un module de traduction sommaire, par exemple pour les titres de pages HTML.

La traduction du réviseur

Cette application de la TAO vise à faire traduire automatiquement par la machine le document à traiter de la façon la plus complète possible, puis à donner en révision le résultat à un traducteur humain. C'est une technique en général appliquée après le passage d'un gros système de traduction comme ARIANE, KANT, ou SYSTRAN par exemple. Cette approche peut poser des problèmes si l'on ne se base pas sur un système adéquat. C'est ainsi que par exemple la tentative d'utiliser LOGOS pour traduire les manuels de COREL a mené à un échec.

La traduction du rédacteur

Le rédacteur de documents assiste dans ce cas l'ordinateur, pour arriver à une traduction de qualité. La première approche de ce paradigme consiste à utiliser un langage contrôlé : le rédacteur est contraint de respecter un vocabulaire et/ou une grammaire connus par l'ordinateur. Des interfaces d'assistance au rédacteur sont alors possibles comme dans JETS d'IBM Japon. La seconde approche est celle de la désambiguïsation : dans ce cas l'auteur accepte de passer en général autant de temps que celui qu'il met à rédiger, pour aider l'ordinateur à comprendre totalement le texte écrit. Une fois cette étape achevée, le système est alors capable de traduire de façon exacte, et cela peut par exemple être rentable lorsqu'il s'agit de traduire d'une langue source vers un nombre important de langues cibles. On peut par exemple se reporter à [Blanchon 1994] pour une étude de cette approche intéressante de la traduction.

La traduction du traducteur

Si les trois paradigmes précédents se focalisent sur la machine en supposant qu'elle fonctionne de façon indépendante pour le premier, ou requiert l'aide du traducteur a posteriori ou a priori pour respectivement les deuxièmes et troisièmes, la traduction du traducteur tend à considérer que l'homme occupe une position centrale et prédominante dans le processus de traduction, et

qu'il est assisté par la machine. On parle alors de Traduction Humaine Assistée par la Machine (THAM).

Dès les années 80, on a vu se développer l'utilisation de dictionnaires et bases de données linguistiques en ligne. Les plus simples de ces outils consistent à donner au traducteur un lexique monolingue ou un glossaire bilingue interrogeable en ligne. Les outils comme les CD spécialisés non modifiables de la Maison du Dictionnaire, les bases terminologiques complètes comme ou Eurodicatom ou les bases de données personnalisables comme Termex TermStar, ou Multiterm en sont des exemples. Ces outils ont évolué vers de vrais dictionnaires ou encyclopédies électroniques en ligne monolingues (Larousse électronique), bilingues (Collins on Line) ou multilingues (Berlitz Interpretor). Ces derniers types de dictionnaires font éventuellement appel à un analyseur permettant de retrouver la forme de base qui est stockée.

Les volumes de demande traduction étant en constante croissance, et notamment avec l'apparition du problème de la traduction des logiciels et de leurs manuels, papier ou électroniques d'une part, et l'augmentation de la puissance des ordinateurs permettant la gestion de gros volumes de données d'autre part, la fin des années 80 et les années 90 ont vu se développer un nouveau type d'aide au traducteur : les *Outils de Traduction Fondée sur la Mémoire* (OTFM), aussi appelées *mémoires de traduction*. Ces outils aident le traducteur en gardant la *mémoire* des traductions précédentes, et en les proposant à nouveau lorsque des phrases similaires sont rencontrées. Ce sont ces outils que nous nous proposons d'étudier maintenant.

1.2 Principe

L'idée de base des Outils de Traduction Automatique Fondés sur la Mémoire**{**xe "Outils de Traduction Automatique Fondés sur la Mémoire"**}** est de construire une mémoire**{**xe "mémoire de traduction"**}** de phrases (sources), et de leur traduction (cibles). L'implémentation en est très simple : on constitue un ensemble de couples du type (source, cible). Cela donne un ensemble de "bi-textes**{**xe "bi-textes"**}**" (ou "bi-segments"). On ne parle pas de "bi-phrases" parce que les textes en question ne sont pas nécessairement des phrases bien formées. Il peut s'agir de syntagmes nominaux, par exemple :"Process to be applied:"-"Opération à suivre :".

Cela donne ainsi une structure proche de celle de la Figure 1. Les fichiers contenant ces "bi-textes" sont appelés "mémoires**{**xe "mémoires de traduction"**}**", et peuvent posséder quelques informations supplémentaires comme le nom de la personne qui les a créés, la date de création, les fichiers de provenance, etc.

La construction de cette mémoire, qui peut être automatisée, se base donc sur deux documents : le document source qui contient les "segments" à traduire, et le document cible qui contient les segments**{**xe "segments"**}** traduits. La recherche de la correspondance entre segments ("l'alignement**{**xe "alignement"**}**"), se fait par similitude statistique.

La Figure 1 représente une telle mémoire pour un manuel informatique traduit du français vers l'anglais (il s'agit d'un vrai exemple, mais banalisé), traitée avec l'OTFM1g Trados Workbench**{**xe "Workbench"**}**.

```
<TrU>
<Seg L=UKE>Describe the Security architecture used by IIS.

<Seg L=FRE>Décrire l'architecture de la sécurité utilisée par IIS.
</TrU>
<TrU>
<Seg L=UKE>In a specific situation, determine what type of security should be used. This includes using the following:
<Seg L=FRE>Dans une situation particulière, déterminer quel type de sécurité doit être utilisé et ainsi utiliser les éléments suivants :
</TrU>
```

<TrU>
<Seg L=UKE>This includes the following:
<Seg L=FRE>Cela comprend les éléments suivants :
</TrU>
<TrU>
<Seg L=UKE>Configure *Produit* to use *Marque* Index Server.
<Seg L=FRE>Configurer *Produit* pour utiliser *Marque* Index Server.
</TrU>

Figure 1: Un alignement avec Trados TAlign (de l'anglais vers le français)

L'utilisation de cette mémoire**{**xe "mémoire"**}** par un OTFM1g est illustrée par la Figure 2. Chaque segment**{**xe "segment de traduction"**}** est comparé statistiquement**{**xe "comparaison statistique"**}** (similarité des caractères de la chaîne représentant la phrase à traduire) à la première partie des bi-textes de la mémoire. Si une chaîne correspondante est trouvée dans la mémoire, la partie cible est copiée à droite du segment à traduire dans le document traité, et un pourcentage de correspondance est inscrit entre les deux. Le pourcentage s'étend d'un seuil choisi par l'utilisateur (70% par exemple, correspondance approchée**{**xe "correspondance approchée"**}**) jusqu'à 100%, qui signifie une correspondance parfaite**{**xe "correspondance parfaite"**}**. Dans le cas où la correspondance est inférieure au seuil, le segment à traduire est recopié à droite, et un taux de 0% est indiqué entre les deux, de telle façon que le traducteur puisse éditer et traduire ce segment.

{0>Describe the Security architecture used by IIS.<}100{>Décrire l'architecture de la sécurité utilisée par IIS.<0}

{0>In a specific situation, determine what type of security should be used.<}79{>Dans une situation particulière, déterminer quel type de sécurité doit être utilisé et ainsi utiliser les éléments suivants :<0} {
0>This includes using the following:<}81{>Cela comprend les éléments suivants :<0}
{0>Configure Produit to use Marque Index Server 1.1.<}89{>Configurer Produit pour utiliser Marque Index Server.<0}
{0>Active Server Pages (ASP)<}0{>Active Server Pages (ASP)<0}

Figure 2 : Une traduction avec Trados Workbench

On notera que dans ce document de travail, la partie source ainsi que les pourcentages et les balises**{**xe "balises"**}** de début et de fin de texte sont formatés en "texte caché", ce qui fait que l'on peut les faire disparaître pour donner au document de travail l'aspect de la version finale du document. Cela donne alors l'aspect de la Figure 3.

Décrire l'architecture de la sécurité utilisée par IIS.

Dans une situation particulière, déterminer quel type de sécurité doit être utilisé et ainsi utiliser les éléments suivants

Cela comprend les éléments suivants :
Configurer IIS pour utiliser Microsoft Index Server.
Active Server Pages (ASP)

Figure 3 : Aspect du document de travail non interactif et masqué sous Workbench

Les couleurs personnalisables indiquent le niveau de traduction : dans l'exemple ci-dessus, le magenta est utilisé pour une "traduction à 100%", bleu pour une traduction approchée**{**xe "traduction approchée"**}** ("fuzzy" en anglais), noir si aucune traduction n'a été trouvée. La

présentation et les fonctionnalités annexes peuvent varier d'un logiciel de TFM à l'autre, mais l'idée générale est celle donnée ici.

1.3 Document littéral et document formaté

Les industriels de la traduction{xe "industriels de la traduction"} ne voient presque jamais de fichier de texte “ nu ”. En fait, les textes à traduire ont généralement été créés à l'aide d'un logiciel du marché comme Microsoft Word{xe "Word"}, FrameMaker{xe "FrameMaker "} ou Interleaf{xe "Interleaf"}. Ces logiciels utilisent un format propriétaire{xe "format propriétaire"}. Ainsi, sous les éditeurs en question, les textes à traduire ont plutôt l’aspect suivant (Figure 4) :

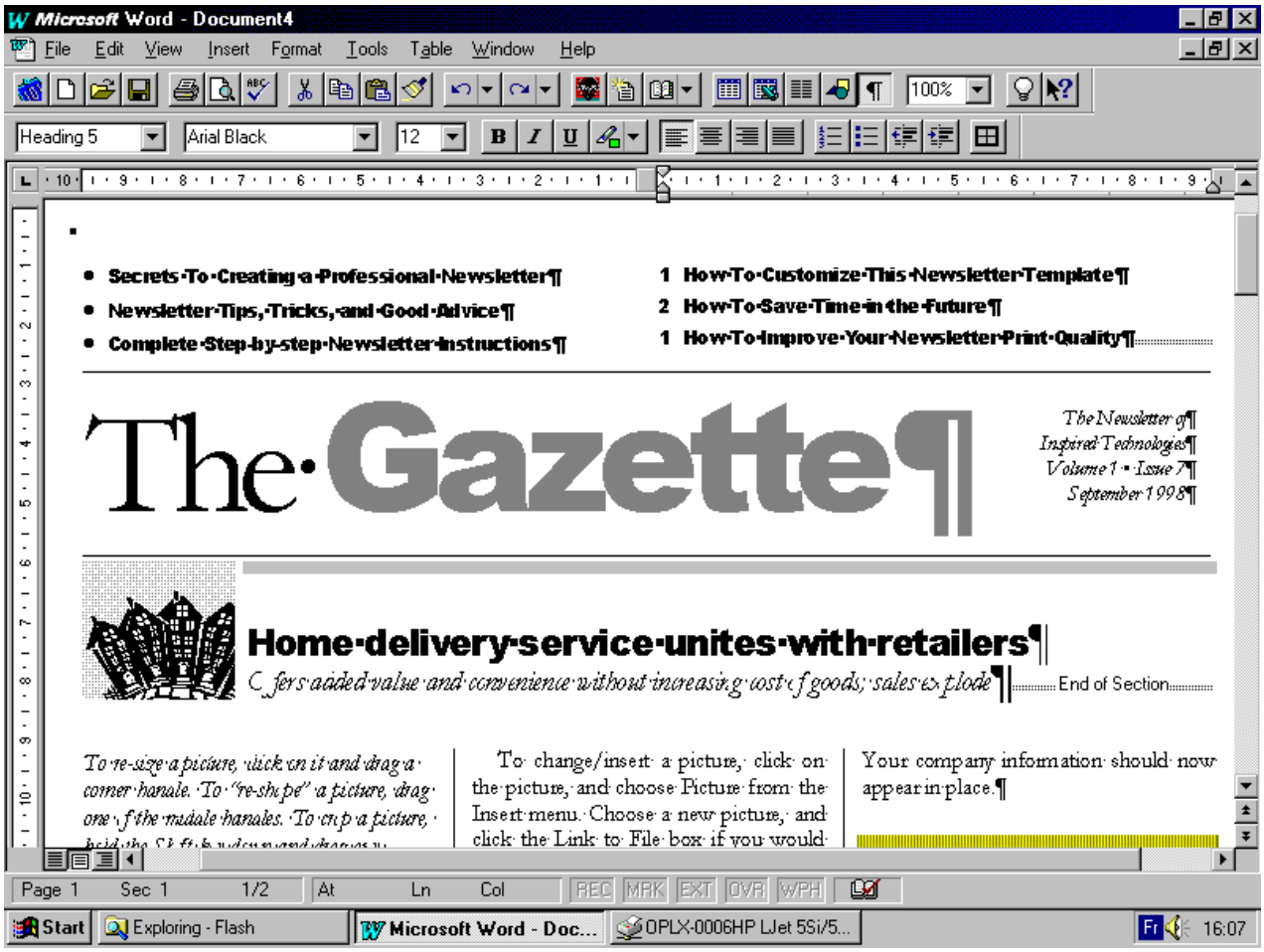


Figure 4 : Document sous Microsoft Word

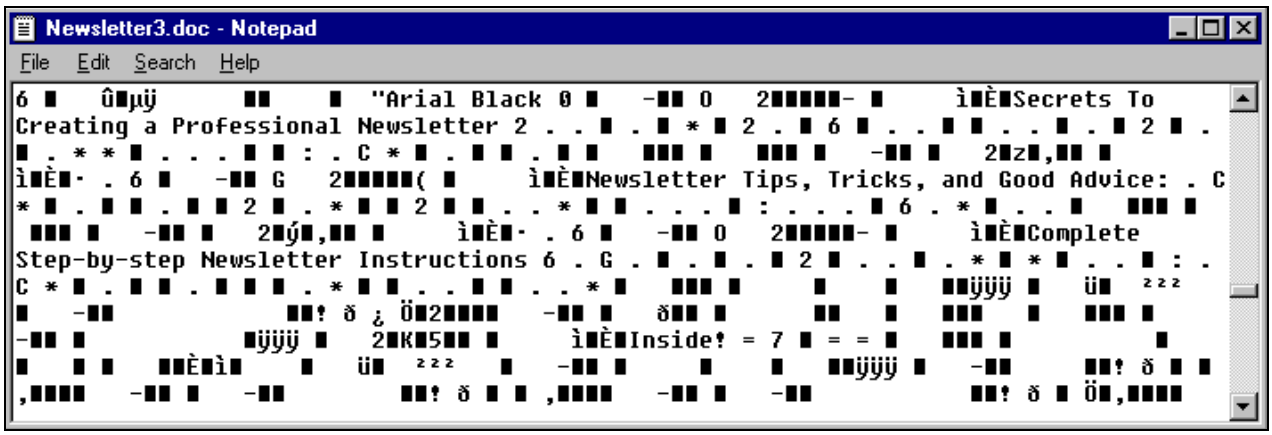


Figure 5 : Code du texte Word de la figure 4

Une partie cohérente de document (un paragraphe, une phrase, un syntagme, un mot : un “segment{xe "segment"}” dans notre jargon) se compose donc de son contenu linguistique{xe

"contenu linguistique"**}**, et c'est ce à quoi l'on pense quand on parle de traduction, mais aussi, ce qui est de la plus grande importance, de sa mise en forme**{**xe "mise en forme"**}**. Le traducteur**{**xe "traducteur"**}** qui traite cette phrase, est bien sûr tenu de rendre un travail propre. Pour cela, il assure traditionnellement ce double travail de traduction et de mise en forme. Substituer la machine à l'homme pour cette tâche de traduction professionnelle**{**xe "traduction professionnelle"**}** demande donc de tenir compte non seulement de la traduction, mais aussi de cette mise en forme.

Les recherches qui portent sur la Traduction Assistée par Ordinateur, traitent presque toujours, à juste titre, la première partie, c'est à dire la traduction littérale elle-même. Nous nous efforcerons ici de tenir aussi compte du transfert de la mise en forme (ou “formatage**{**xe "formatage"**}**”) depuis le document source**{**xe "document source"**}**, vers le document traduit**{**xe "document traduit"**}**.

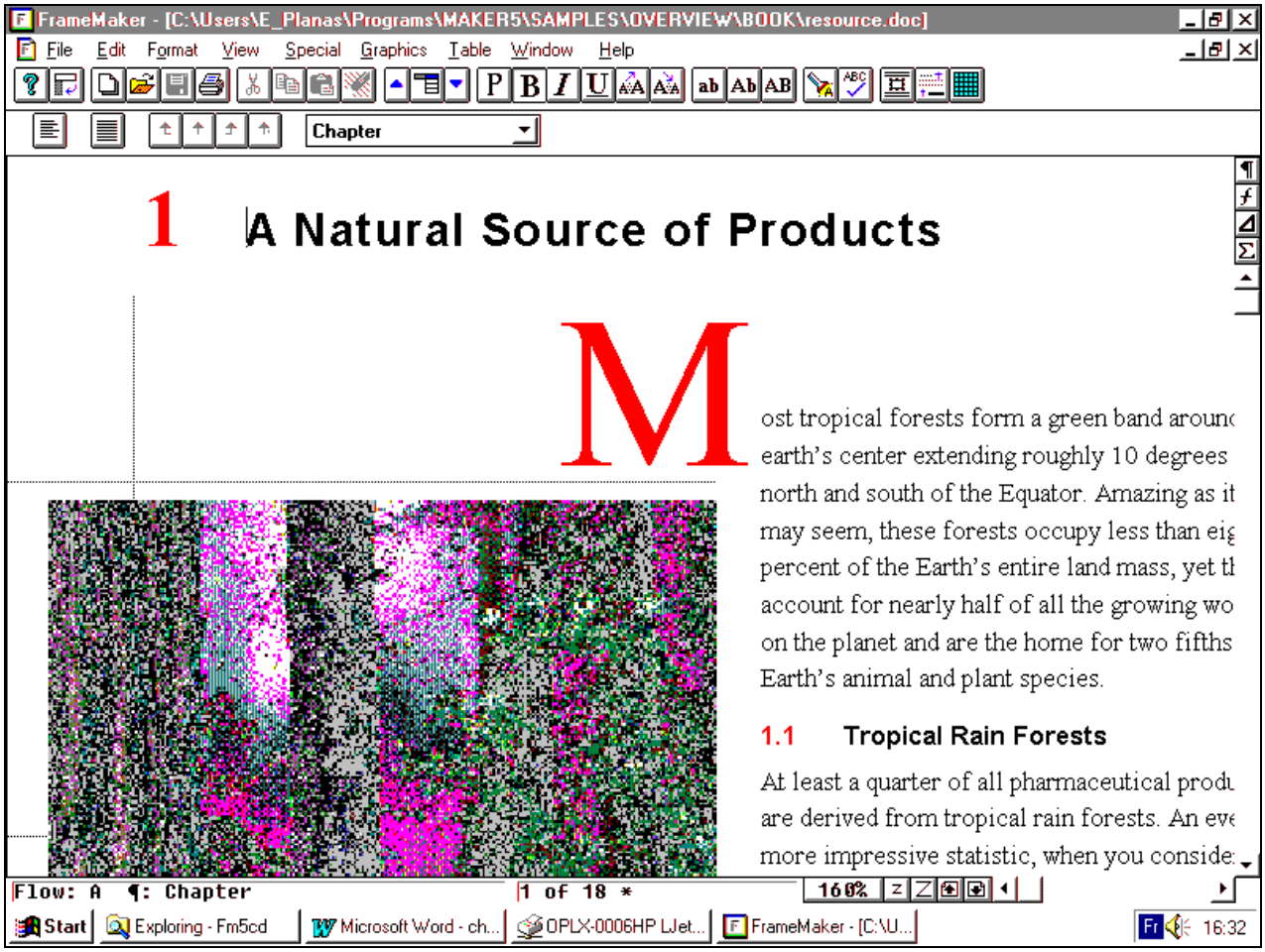


Figure 6 : Document sous FrameMaker

Si l’aspect extérieur des documents édités sous différents traitements de texte paraît assez proche (et c’est souhaitable pour respecter les standards de l’édition), la représentation interne**{**xe "représentation interne"**}**, c’est-à-dire le code brut de ces fichiers est lui complètement différent, comme le montrent les Figures 5, et 7. Ce code dépend du format de représentation du document. Il change avec chaque éditeur. Le paragraphe suivant rappelle quelques formats de l’industrie de l’édition.

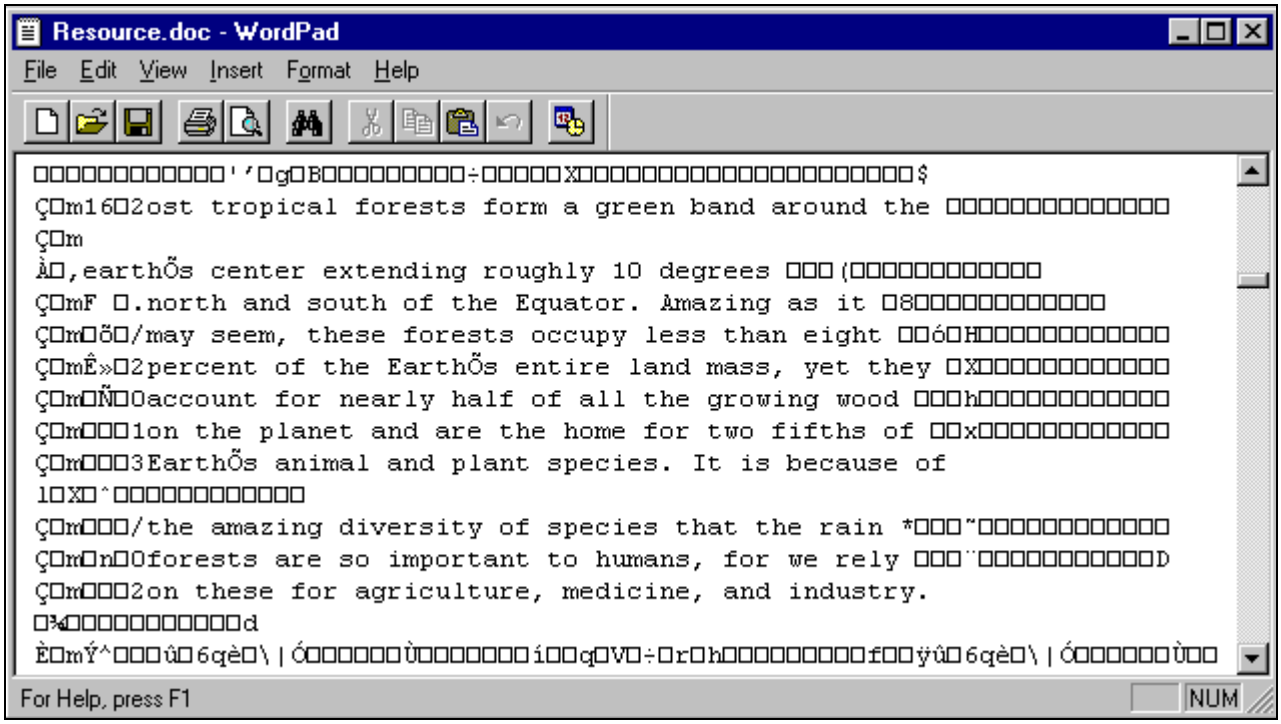


Figure 7 : Code du document FrameMaker de la figure 6

1.4 Les formats rencontrés sur le marché

Aujourd'hui, une kyrielle de formats d'édition{xe "formats d'édition"} existe sur le marché. En fait, chaque éditeur utilise son propre format et parfois un "format d'échange" complémentaire, lui aussi propriétaire (par exemple MIF{xe "MIF"} pour FrameMaker{xe "FrameMaker"}, RTF{xe "RTF"} pour Microsoft). De plus, la spécification de ce format change d'une version à l'autre du logiciel d'édition, et parfois d'un système d'exploitation à l'autre. Voici une liste indicative et loin d'être exhaustive, de formats rencontrés sur le marché{xe " formats rencontrés sur le marché"} :

- Adobe FrameMaker+ MIF
- Adobe Illustrator
- Adobe PageMaker
- Amipro
- ClarisWorks
- Interleaf
- LaTeX
- Microsoft HTML 2, 3, 4
- Microsoft Word 6, 7, 8 + RTF (Macintosh et Windows)
- Mosaic HTML 2, 3, 4
- Netscape HTML 2, 3, 4
- NisusWriter
- Quark Xpress
- Ventura
- WordPerfect
- Works
- Write

Cela étant, parmi cette jungle de formats, certaines tentatives d'uniformisation existent. On peut penser au métalangage pour l'édition qu'est SGML{xe "SGML"}, et dont les

différentes versions de HTML{xe "HTML "} en sont des implémentations. Certains groupes relatifs à SGML travaillent sur les problèmes spécifiques aux langues et aux format (TEI{xe "TEI"}), et sur la représentation des documents multimédias (XML{xe "XML"}, voir Chapitre 2 pour plus de précisions). Comme le formatage pose des problèmes aux sociétés de traduction qui doivent jongler avec eux, et demander à leurs traducteurs de savoir se servir d'autant de logiciels différents, certaines ont commencé l'étude de formats d'échange{xe "formats d'échange"}. Parmi eux par exemple, la société ILE travaille sur OpenTag{xe "OpenTag"}. Aussi, un logiciel de traduction qui veut traiter les documents du marché doit aussi tenir compte de ces formats, et des spécificités engendrées.

2. Utilisation industrielle des OTAF1g

2.1 Les étapes de la traduction à l'aide d'un outil de traduction fondée sur la mémoire

2.1.1 Étapes de base

L'utilisation industrielle efficace et rentable d'un outil de traduction fondée sur la mémoire (TFM) requiert une méthodologie qui comporte des étapes autres que la simple traduction à l'aide de l'outil de TFM. Dans la suite est esquissé un exemple de méthodologie de traduction**{**xe "méthodologie de traduction"**}**, adaptée aux outils de TFM.

Construction de la mémoire

Rassemblement des documents parallèles

Ce n'est pas nécessairement une étape simple, en particulier si le client est nouveau pour la société de traduction. En outre les sociétés de traduction ne conservent pas nécessairement les documents traduits pendant longtemps.

Filtrage et pré-traitement du formatage des documents parallèles

Outre le filtrage vers le format traité par l'aligneur[1], pour que l'alignement soit efficace avec les aligneurs de première génération, un pré-traitement**{**xe "pré-traitement"**}** peut être nécessaire. Un des pré-traitements classiques consiste à appliquer un retour chariot après les phrases (ou les unités de textes) pour qu'elles soient clairement séparées, et que l'alignement entre le fichier source et le fichier cible soit plus facile.

Alignement des documents pour obtenir une mémoire brute

L'alignement proprement dit peut alors débuter. Des outils cousins des outils de traduction permettent souvent de réaliser cet alignement**{**xe "alignement"**}** de façon automatisée. C'est le cas par exemple pour le logiciel Workbench de TRADOS qui utilise l'outil TAlign ou WinAlign**{**xe "TAlign"**}**, pour Eurolang Optimizer de la société LANT qui utilise l'outil Aligner**{**xe "Aligner"**}**. Transit de STAR et Translation Manager d'IBM, possèdent un aligneur interne.

Relecture et correction de la mémoire : mémoire finale

L'alignement**{**xe "alignement"**}** est cependant rarement parfait. L'humain doit réviser ces alignements pour s'assurer que des fautes n'ont pas été commises. Le terminologue peut imposer dès ce niveau une terminologie à respecter, en révisant et appliquant les termes corrects à cette mémoire.

Traduction

Pré-traitement du formatage des fichiers

Parce que les fichiers à traiter sont rarement "prêts" pour le logiciel de TFM au niveau de leur format d'enregistrement, mais aussi au niveau des objets à traduire, ces fichiers doivent être pré-traités. Ce pré-traitement peut par exemple consister à appliquer un format "caché" à certaines parties du fichier qui ne doivent pas être traduites. En général, une conversion du format d'enregistrement est aussi nécessaire.

[1] Logiciel qui permet de construire une mémoire à l'aide d'un texte et sa traduction. Ce logiciel met alors en correspondance chaque phrase source avec sa traduction (aux correspondances non bijectives près). On dit qu'il aligne le texte source et le texte cible.

Pré-traduction par l'outil de TFM

Une fois les fichiers préparés, l'outil de TFM peut alors s'appliquer. Les phrases sont pré-traduites{xe "pré-traduction"}. Cela donne un document de travail hétérogène au niveau de la traduction. Certaines phrases sont en effet totalement traduites (100%), d'autres partiellement (entre un seuil et 100%), et d'autres pas du tout, car aucune phrase de la mémoire{xe "mémoire de traduction"} ne correspond de près ou de loin à la phrase à traduire.

Traduction humaine{xe "Traduction humaine"}

Le traducteur humain utilise alors ces pré-traductions pour finaliser le travail. Il vérifie et modifie les traductions partielles{xe "traduction partielle"}, et traduit complètement les phrases pour lesquelles aucune traduction n'a été trouvée. De plus, et ce n'est pas une étape négligeable, le traducteur transfère le formatage et les objets imbriqués du texte source sur le texte cible. Cette phase peut s'effectuer sous l'éditeur spécial de ces outils de traduction, ou sous un éditeur classique.

Révision linguistique{xe "Révision linguistique"}

C'est une étape qui est en général assurée par une tierce personne. Cela permet par exemple aux sociétés de traduction{xe "sociétés de traduction"} de faire vérifier le travail sous-contracté à des traducteurs indépendants, par leurs réviseurs internes de haut niveau, ou connaissant bien les produits traduits.

Post-traduction{xe "Post-traduction"} par l'outil de TFM

Cette étape permet à la fois de retirer les marques éventuellement ajoutées par l'outil de TFM dans le document à traduire pour former un document de travail agréable (présentation par phrases sources et cibles séparées par le taux de traduction).

Elle permet aussi de récupérer automatiquement de nouvelles entrées pour la mémoire de traduction et les bases terminologiques, puisque les traductions ont été vérifiées par les réviseurs{xe "réviseurs"}. Cela est presque vrai. En effet, dans le cas de traduction de logiciels, pour obtenir la dernière version de la traduction, il peut être nécessaire d'attendre la reconstruction totale du logiciel et le test technique final pour les corrections d'ordre linguistique technique.

Post-traitement{xe "Post-traitement"} du formatage des fichiers

Une fois le fichier traduit par l'outil de TFM, et la mémoire de l'outil augmentée des traductions humaines supplémentaires, le fichier peut être enregistré à nouveau sous son format d'origine, et les parties cachées à l'étape du pré-traitement des fichiers à traduire, libérées.

Gestion de la mémoire post-traduction{xe "Gestion de la mémoire post-traduction"}

La récupération automatique des mémoires n'étant pas nécessairement sophistiquée dans les outils de première génération, il est nécessaire d'assurer la cohérence de la mémoire en vérifiant par exemple les doublons. Le stockage systématique des fichiers de mémoires de traduction est aussi une des tâches que le terminologue doit assurer, pour constituer les archives de la société de traduction qui serviront en cas de nouveau dossier similaire.

Etapes supplémentaires pour la traduction de logiciels

La traduction de logiciels peut demander d'autres étapes. Nous en parlons plus longuement au chapitre suivant. Voici cependant quelques étapes supplémentaires indicatives.

- extraction des ressources à traduire
- insertion d'icônes localisées
- insertion de copies d'écrans
- révision technique

- test technique

2.2 La traduction de logiciels, ou localisation

2.2.1 Chaine des traitements

Introduction

"Localisation**{**xe "localisation"**}**" est l'anglicisme emprunté à l'américain par les industriels de la traduction pour désigner l'activité qui consiste à traduire les logiciels. Cette activité n'est pas une simple activité de traduction de textes (même formatés). Elle inclut en effet plusieurs activités parallèles allant jusqu'à la programmation. Voici quelques unes de ces activités.

Extraction et réinsertion des chaînes du logiciel à traduire

Les chaînes à traduire se trouvent souvent dans des parties différentes et très internes du logiciel, où il faut aller les chercher. Ainsi, dans une programmation de mauvaise qualité comme on en rencontre, les messages d'aide sont parfois spécifiés "en dur" par des instruction "println" par exemple dans les programmes C.

Heureusement, la situation s'est nettement améliorée depuis quelques années, notamment grâce à quelques études et à la publication de quelques livres de localisation. Ainsi trouve-t-on le plus souvent des "fichiers de ressources**{**xe "fichiers de ressources"**}**" qui contiennent tous les messages d'erreurs apparaissant par exemple dans l'interface de photocopieurs ou d'imprimantes laser. Ces fichiers comportent pourtant certaines notations spéciales qui font que parfois les logiciels de traduction automatique butent sur des signes tels que "&N" ("raccourci clavier pour créer un nouvel élément").

Adaptation de la dimension des objets contenant des chaînes traduites

Certains objets présents dans les logiciels tels que les boutons, les titres, les menus, ou les fenêtres d'affichage de commentaires ou d'erreurs ont une dimension physique (hauteur, longueur) fixée par défaut. Lors du passage d'une langue à l'autre la longueur du texte peut augmenter (de l'anglais au français, par exemple). Cela entraîne que les textes traduits ne sont parfois plus visibles. Il faut alors "retailler" les boutons ou autres objets cités ci-dessus.

"Flashage" et insertion de copies d'écrans

Les aides de logiciels présentent souvent des images montrant une fenêtre de ce même logiciel pour illustrer les commentaires de description d'une tâche utilisant ce logiciel. Ces "copies d'écrans" comportent bien sûr du texte dans la langue de conception du logiciel. Le passage à la version traduite demande donc de procéder au tirage de nouvelles copies d'écran**{**xe "copies d'écran"**}** (le "flashage" dans le jargon du domaine), et au remplacement de ces copies d'écrans sources par celles des cibles.

Adaptation culturelle des icônes

Bien que très peu d'éditeurs procèdent à cette étape, un logiciel a plus de chances d'être utilisé si les icônes sont adaptées à la culture du marché cible. Le remplacement des icônes**{**xe "icônes"**}** sources par les icônes cibles est donc une activité qui devrait être plus souvent assurée par les localisateurs. Si cela n'est que rarement fait, c'est certainement aussi à cause du coût et de la durée de l'opération. Il existe pourtant des solutions d'automatisation, comme celles proposées dans [Semmar, Planas et Fluhr 1997].

Changement du support informatique des langues

Un logiciel interagit avec l'utilisateur. Aussi doit-il afficher, imprimer et comprendre ce que l'utilisateur lui indique, en particulier via le clavier. Cela suppose que les différentes fonctionnalités sont adaptées à la culture du pays cible [Ishida 1998, Kano 1995]. Ainsi les procédures d'entrée des lettres et mots, les procédures d'affichage, de tri, de traitement des dates et des monnaies doivent être adaptées pour offrir un support informatique des langues**{**xe

"support informatique des langues"**}** efficace. Cela peut être plus aisé sous certains systèmes d'exploitation comme les Système 7 ou 8 de Macintosh, mais reste quand même à effectuer.

2.2.2 Cas des fichiers d'aide

Le cas de la localisation des fichiers d'aide**{**xe "fichiers d'aide"**}** est significatif. Ces fichiers deviennent de plus en plus stratégiques pour la traduction d'un logiciel. Les éditeurs en effet ont tendance à augmenter le nombre de pages des fichiers d'aide pour fournir "en ligne" les informations nécessaires à l'utilisateur, et cela au détriment des manuels papiers. Voici un extrait d'un tel fichier d'aide, sous Windows :

Setup Detail Planning
[seealsob.shg]

This menu option opens the **Setup Detail Planning** window where you may set up your salary and asset planning.
The ***Commander Budget Detail Planning*** modules allow your budget holders to enter and analyze their salary, headcount and asset information at a detailed level. The totals per unit can then be automatically pulled into the submission sheets. See Setting Up SubmissionSheets to Pull Planning Figures for instructions on how to set this up for budget holders.
Note: Asset planning is intended to help budget holders calculate depreciation values for new assets only.
See procedure {**BUI031 *Define Budget*}.

Figure 8 : Exemple d'aide en ligne de logiciel

Outre le formatage du texte (gras, italique, souligné), ce fichier comporte quatre éléments imbriqués**{**xe "éléments imbriqués"**}** ("embedded objects", en anglais), qui doivent eux aussi être pris en compte dans la traduction. Ces éléments sont :

- un champ de bouton : Setup Detail Planning
- un champ d'image [seealsob.shg]
- un hyperlien : Sheets to Pull Planning Figures
- Un appel de procédure: *Define Budget*

En mode d'affichage normal, ces objets apparaissent selon le formatage visible qui leur a été assigné, comme ci-dessus. En réalité ce sont des objets complexes qui possèdent une structure propre derrière cette apparence, qui peut être en partie rendue visible si l'affichage des champs est spécifié :

- {macrobutton HdkTopic HDK3B9ACA1DSetup Detail Planning}
- {macrobutton HdkPicture PIC00000009[seealsob.shg]}
- {macrobutton HdkLink LNK00000046 Setting Up SubmissionSheets to Pull Planning Figures}
- {**BUI031 *Define Budget*}
  }

Pour traduire ces objets, il est donc nécessaire d'extraire les chaînes à traduire. Or cela n'est pas tout à fait automatisable sans connaissance de la syntaxe du logiciel, syntaxe qui change souvent d'un éditeur à un autre. Dans notre cas les chaînes à extraire et à traduire sont :

- Setup Detail Planning
- see also
- Sheets to Pull Planning Figures
- *Define Budget*

En fait ces boutons renvoient à des chapitres différents de l'aide électronique par l'intermédiaire d'un hyperlien**{**xe "hyperlien"**}**. Le titre de la partie à laquelle le bouton renvoie est indiqué dans le bouton (ex : Setup Detail Planning). Il faut alors que la traduction de ce titre soit exactement celle du vrai titre, qui se trouve quelques pages avant, ou après. Le traducteur doit alors aussi penser à la cohésion des traductions. Parfois des procédures de l'éditeur permettent de récupérer directement les titres dans les boutons, qui ne doivent alors pas être traduits.

2.2.3 Cas de ressources de logiciel

Les ressources de logiciels**{**xe "ressources de logiciels"**}** sont des fichiers qui regroupent les données nécessaires à un logiciel, par type. Ces données peuvent être des images, des données numériques ou encore les textes des messages, des menus ou des fenêtres. L'extrait suivant provient d'un fichier de ressources regroupant les textes affichés d'un logiciel, et montre deux messages d'erreurs et un message de commande :

```
MSG_COULD_NOT_LOAD_DLL."Could not load the DLL ( %s1 )! Load Library returned: %s2"
MSG_COULD_NOT_GET_INFO_ABOUT_LOCKED_LINES
……."Could not get information about Locked Lines from the database!"
CAP_GNVFILED_SAVE_AS…."Save As"
```

Figure 9 : Exemple de fichier de ressources de logiciel.

La traduction de ce type de fichier obéit en particulier aux lois suivantes :

- On ne doit traduire que certaines des chaînes qui sont entre guillemets.
- On doit conserver les (%1) qui correspondent à des variables.
- Le nombre d'espaces précédant les messages (matérialisés dans notre illustration par des points) doit être conservé car il correspond à un placement dans la fenêtre d'affichage du message.

On peut dans ce cas se poser la question de la conservation de l'information contextuelle au niveau d'une mémoire de traduction. Serait-il intéressant de pouvoir conserver, outre la chaîne source et la chaîne cible, l'information selon laquelle il s'agit de la traduction de tel message de numéro tant, du fichier un tel, de tel logiciel ?

2.3 Conséquences pour les OTFM1g

Les paragraphes précédents nous montrent que la traduction industrielle de documents et de logiciels (localisation) n'est pas une opération s'appuyant sur le seul contenu linguistique, ou sur le seul texte nu des documents traités.

Tout au contraire, il s'agit bien du traitement d'un ensemble d'objets documentaires incluant bien sûr le texte lui-même mais aussi les codes des mise en forme, les objets statiques (marques d'index, images), et les objets dynamiques (liens hypertextes, boutons d'actions). Les OTFM1g doivent donc tenir compte du caractère hétéroclite des documents de telle façon à donner les meilleurs résultats de traduction.

De même les unités de documents (les phrases, les paragraphes, les pages, les chapitres, les livres) peuvent faire partie d'une structure externe non textuelle comme un programme

informatique et ses ressources de chaînes de caractères, ou un fichier d'aide et son architecture. L'architecture combinant ces unités de texte doit être respectée par les outils de Traduction Fondée sur la Mémoire.

3. Les gains de traduction

3.1 Redondance

Introduction

La redondance**{**xe "redondance"**}** d'un texte 2 par rapport à un texte 1, est la répétition d'une ou plusieurs parties du texte 1 dans le texte 2. Voici un exemple.

Texte 1 :

Cliquez sur le bouton droit. Les termes d'interface du logiciel "Draw" seront soulignés d'un double trait. L'interface a été conçue pour faciliter l'utilisation du logiciel "Draw". Cliquez sur le bouton droit.

Texte 2 :

Les messages d'interface seront soulignés d'un trait. L'interface a été conçue pour faciliter l'utilisation du logiciel "Draw". Cliquer sur l'icône rouge.

3.1.1 Intra-redondance

Dans le texte 1, "Cliquer sur le bouton droit" est utilisé une fois en début de texte, une fois en fin. Il est donc répété dans un même texte, la redondance est interne au document, et nous l'appelons "**intra-redondance{**xe "intra-redondance"**}**". Le calcul de l'intra-redondance donnera donc ici, en termes de phrases : 2 phrases sur 4 redondantes, soit encore 50%. En termes de mots, par contre, cela représente (10/35) = 28%.

3.1.2 Inter-redondance

C'est celle qui met en jeu deux documents différents. Ainsi, le fait de retrouver dans le Texte 2 la phrase "L'interface a été conçue pour faciliter l'utilisation du logiciel "Draw".", qui est aussi présente dans le Texte 1 est appelé "**inter-redondance{**xe "**inter-redondance**"**}"** entre les textes 1 et 2. En termes de phrases, l'inter-redondance entre ces deux textes est de 1/3 = 33% (on se base sur le nombre de phrases du texte à traiter, supposé être le Texte 2, ici). En termes de mots, les 12 mots de la deuxième phrase du Texte 2 peuvent être traduits parmi ses 26 mots, ce qui fait 46 %.

3.1.3 Unité de décompte de la redondance

La première conclusion à laquelle on parvient est que pour avoir un chiffre significatif de la redondance**{**xe "redondance"**}**, il faut prendre comme unité traitée le mot, et pas la phrase. Il n'est en effet pas vrai que 50 % du texte 1 a été traité, en traitant 50 % des phrases.

Nous prendrons donc comme unité de décompte le mot.

Nous verrons plus loin que les logiciels de TFM du marché ne savent pas lemmatiser**{**xe "lemmatiser"**}** les mots (ou utilisent mal ces lemmatiseurs). L'unité pour le décompte de la redondance concernant ce chapitre sera donc le mot, et pas le lemme. Gardons quand même à l'esprit, qu'avec des outils capables de lemmatiser les entrées, la meilleure mesure est bien sûr le lemme (pas de différence entre "table" et "tables", ou "écrire" et "écris").

Soit une phrase P_i, de p_i mots. Supposons qu'il y ait o_i occurrences de cette phrase dans le document à traduire, document comportant N mots. Alors le taux de redondance r_a^i intra-document unitaire (pour cette phrase) est :

$r_a^i = o_i * p_i / N$

Formule 1 : Taux de redondance intra-document phraséologique unitaire

Soient n phrases P_i de p_i mots. Soient o_i les occurrences de phrases redondantes dans le document à traduire, document comportant N mots. Alors le taux de redondance r_a intra-document (total) est :

$r_a = \frac{\sum_i o_i * p_i}{N} = \sum_i r_a^i$, où les o_i représentent les occurrences des phrases redondantes.

Formule 2 : Taux de redondance intra-document phraséologique

Remarque importante :

Il est important de compter les mots groupés par phrases (redondance phraséologique{xe "redondance phraséologique"}). En effet, même s'il est vrai qu'un mot répété plusieurs fois constitue une source de redondance au niveau de ce mot, dans la pratique un mot traduit isolé ne sert que peu, car le traducteur doit traduire le texte précédant le mot, puis sauter le mot, puis traduire la fin de la phrase. Seules les phrases (ou les groupes de mots cohérents) doivent être pris en compte dans le calcul de la redondance. Cela est particulièrement vrai pour les OTFM1g qui ne sont pas capables de remplacer des mots isolés.

Exemple :

Soit un texte T composée de p_i phrases d'occurrences o_i vérifiant :

$T = p_1\ p_2\ p_1\ p_3\ p_4\ p_2\ p_5\ p_2$.

et tel que le nombre de mots par phrase vérifie :

phrase	p_1	p_2	p_3	p_4	p_5
nb de mots	10	15	10	20	5
occurrences	2	3	1	1	1

Alors le nombre total de mots du texte est N = (2 x 10) + (3 x 15) +10 +20 +5 = 100

Les phrases p_1 et p_2 étant employées plusieurs fois, elles sont redondantes au sein de ce texte. L'ensemble des mots participant à cette redondance est l'ensemble des mots des instances de ces phrases redondantes, soit (2 x 10) + (3 x 15) = 65. Cela représente un taux d'intra-redondance de :

ra = 65 / 100 = 65 %

3.2 Gains de traduction

3.2.1 Définition

Reprenons le cas de la phrase précédente qui apparaît deux fois dans le document à traduire. Supposons aussi que cette phrase n'existe pas dans la mémoire avant la traduction de ce texte. On va donc pouvoir repérer que cette phrase est en double, et ne la traduire qu'une seule fois au lieu de deux. Cependant, il faut bien se rendre compte qu'elle sera quand même traduite une fois. En fin de compte, le gain**{**xe "gains de traduction"**}** sera de (10-5)/35 (soit 11% du texte), et pas 28%. Dans ce cas on retire en effet les cinq mots de la phrase à traduire).

Le deuxième enseignement de cela est que les gains maximaux de traduction sont inférieurs au taux de redondance.

De plus, ils sont liés à la fréquence de redondance de chaque unité. En effet une phrase devra toujours être traduite au moins une fois. Partant, plus elle est répétée de fois, plus sa fréquence est élevée, et plus les gains de traduction le seront.

$t_a = \frac{\sum_i (o_i - 1) * p_i}{N} = r_a - \frac{1}{N}\sum_i p_i$, o_i étant le nombre d'occurrences des phrases redondantes.

Formule 3 : Gain de production maximal en situation d'intra-redondance

Remarque :

Ce gain est appelé maximal, car il suppose que d'une part toute redondance peut être couverte par l'outil de TFM, et d'autre part que l'utilisation d'un outil de TFM a un coût nul, ce qui est faux dans la plupart des cas.

3.2.2 Inter-redondance et gains de production

Dans le Texte 1 ci-dessus, nous voyons que la phrase doublement soulignée peut être aussi dans le Texte 2. Il y a donc inter-redondance**{**xe "inter-redondance"**}** entre ces deux textes.

Supposons que le Texte 1 soit un document qui a été traité antérieurement au Texte 2. Le traitement de cette redondance se fera donc par fabrication d'une mémoire de traduction fondée sur le premier, et appliquée au second. Le taux de redondance va donc maintenant s'évaluer en faisant le rapport des portions communes entre ces deux textes. Supposons que ce taux d'inter-redondance est r_e = 28%. Dans ce cas le gain de traduction t sera bien égal au taux d'inter-redondance :

$t = \frac{\sum_i o_i * p_i}{N} = \sum_i r_e^i = r_e$

Formule 4 : Gain maximal de production en situation d'inter-redondance

Remarque :

On ne perdra pas de vue que la phrase doit être traduite une fois au moins. Dans le cas où l'on constitue une mémoire a priori, cette traduction provient de l'alignement du premier document (Texte 1, dans notre exemple) avec sa traduction. Or cet alignement a lui aussi un

coût. Le gain maximal ainsi défini plus haut n'est qu'un gain théorique, auquel il faut amputer notamment le coût de l'alignement.

3.2.3 TFM et gains de production

Il est bien clair que plus il y a de parties du document à traduire qui peuvent l'être de façon automatisée, sans que le traducteur ait à intervenir, meilleurs seront les gains de production**{**xe "gains de production"**}**. Cela est vrai sous certaines conditions dont voici les plus significatives :

- le traducteur ne doit pas revenir sur la phrase. Cela sous-entend qu'il ne doit retoucher ni à la traduction, **ni au format de la phrase, ou à tout autre objet relatif à la phrase**.
- la portée de cette redondance doit être la phrase au moins. Le traducteur perd en effet du temps à naviguer entre les mots ou les segments traduits, immergés au milieu de segments que la machine n'a pas pu traduire.
- la traduction automatique par lots est nettement plus rentable que la traduction automatique intéractive.
- le coût supplémentaire des traducteurs et ingénieurs linguistes (pour constituer et maintenir la mémoire, préparer et post-traiter les documents) doit être économiquement inférieur à celui qui sera gagné en ne traduisant pas ce que la machine a déjà traduit.

On n'oubliera pas de remarquer quand même que la traduction automatisée apporte un plus qui n'est pas négligeable : **l'homogénéité**. Cela peut être particulièrement important pour des projets de traduction de documents volumineux faisant intervenir plus d'un traducteur sur les mêmes documents.

3.3 Simulation

A titre d'exemple, examinons le coût de l'alignement d'un texte de N mots. Admettons que de façon simplifiée, la procédure puisse se résumer par les étapes ci-dessous, et avec les taux de production de mots par jour indiqué entre parenthèses :

- alignement automatique (50.000 mots/jour)
- révision de l'alignement (10.000 mots/jour)
- traduction humaine: (2500 mots/jour)
- révision linguistique : (6000 mots/jour)

Alors les coûts de traduction en nombre de jours, d'un texte de N mots, sont les suivants, où α est le taux de traduction automatique.:

- alignement automatique (N/50.000 jours)
- révision de l'alignement (N/10.000 jours)
- traduction humaine: ((1-α)N/2500 jours)
- révision linguistique : ((1-α)N/6000 jours)

En considérant que le coût des acteurs (traducteurs, réviseurs, ingénieurs informaticiens linguistes) est le même pour les quatre tâches ci-dessus[2], le coût A de la traduction de ce texte sans traduction automatique est alors :

$$A = N/2500 + N/6000$$

et le coût avec traduction automatique est :

$$B = N/50.000 + N/10.000 + (1-\alpha)N/2500 + (1-\alpha)N/6000$$

Si l'on veut B < A, cela donne la condition suivante sur α :

$$\alpha > 21\%$$

Ainsi, avec les seuls coûts d'alignement, nous voyons que la ré-utilisation d'un Texte 1 sur un Texte 2 de longueur sensiblement égale (une mise à jour, par exemple), n'est rentable que si un taux de traduction automatique de 21% est atteint. Heureusement, le coût de l'alignement s'amortit avec le nombre de versions textes traduites. Pour la troisième version d'un document, par exemple, la condition à satisfaire est B + C > 2A, avec :

$$C = (1-\alpha)N/2500 + (1-\alpha)N/6000$$

représentant le coût de la traduction du troisième texte, car l'alignement a déjà été fait pour le deuxième, et cela donne comme condition sur α :

$$\alpha > 10\ \%$$

A titre d'information, la traduction de la nouvelle version d'un manuel de logiciel, possède en moyenne un taux d'inter-redondance (au niveau des phrases) compris entre 60% et 90%. Cette proportion est à peu près totalement traitable par les OTFM1g (dans les limites qui sont

[2] Ce n'est en général pas vrai.

présentées dans les deux chapitres suivants) au niveau de la traduction du texte pur, mais si un nouveau format est appliqué à une phrase, alors il n'est pas récupérable avec les OTFM1g.

Conclusion

Il apparaît donc que l'activité de traduction industrielle se fonde sur le traitement de documents complexes, et dans tous les cas non réduits à des lignes de texte pur. Nous avons montré le principe des outils de traduction fondée sur la mémoire, et les étapes de base nécessaires à leur mise en œuvre sur les types de ces documents. Enfin, une évaluation des gains de traduction qu'ils peuvent apporter vient d'être proposée.

Chapitre 2

Etude de quelques outils de Traduction Fondée sur la Mémoire

Contenu du chapitre

Introduction

Le but de ce chapitre est de montrer dans une étude assez complète comment fonctionnent les OTFM1g actuels, quelles sont leurs possibilités et leurs performances.

Plusieurs OTFM1g existent dans le monde occidental. En voici une liste indicative : Eurolang Optimizer de LANT, Transit de STAR, Translation Manager d'IBM, Workbench de Trados, ForgeinDesk, PowerGlot, Reword Studio, GDK localization suite d'Accent, LXEdit d'ILE, Catalyst de Corel, OTM d'ORACLE, Borneo d'ILE, DéjaVu d'Atril et XL8. Certains sont à vocation générale (ex : Workbench), d'autres sont spécialisés pour la localisation (ex : XL8). Nous allons étudier trois des plus répandus des outils à vocation générale :

- Workbench 1.14 de la société TRADOS (Allemagne)
- Translation Manager 2.0.7 d'IBM (Etats Unis)
- Transit de la société STAR 2.6 (Allemagne)

Ce chapitre présente les caractéristiques et fonctionnalités de ces outils. Le chapitre suivant est un test des performances de ces outils. On trouvera en Annexe 2 des tableaux des caractéristiques précises de ces OTFM1g, complétant l'étude présentée ci-après.

1. Workbench de TRADOS[1]

1.1 Généralités

La société Trados**{**xe "Trados"**}** (Allemagne) propose un outil pour la TFM appelé Workbench**{**xe "Workbench"**}**. Cet outil peut être couplé à un gestionnaire de terminologie, Multiterm**{**xe "Multiterm"**}**. En outre, un outil d'alignement, TAlign (sous DOS) ou WinAlign (dernière version, sous Windows)**{**xe "TAlign"**}**, permet de former des mémoires de traduction utilisables par Workbench.

[1] Ce test a pu être mené grâce à l'offre amicale d'une licence de Workbench™ par Trados™ France, Monsieur Stoll et Madame Algardy doivent être ici spécialement remerciés.
TRADOS France s.a.r.l., 40, rue de la Montagne-Ste-Geneviève, F-75005 Paris, http://www.trados.com/

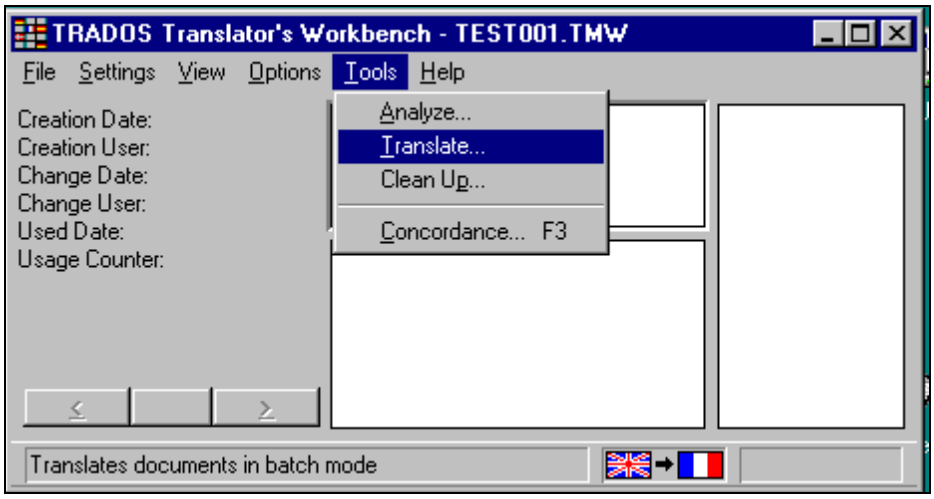


Figure 1 : Trados Workbench, fenêtre principale.

L'outil fonctionne sous Windows 95. La version 1.14 permet de traduire depuis les langues européennes{xe "langues européennes"} (y compris les langues de l'Est comme le polonais ou le tchèque, les langues cyrilliques, le grec, ou les langues régionales comme le catalan et le gaélique), vers celles-ci, et vers les langues asiatiques incluant le japonais {xe "japonais"}, le chinois{xe "chinois "} et le coréen{xe "coréen"}.

Le format de traitement utilisé par défaut est le format Microsoft RTF{xe "RTF"} de Word{xe "Word "}, ou le format WordPerfect{xe "WordPerfect"}. Des outils supplémentaires de filtrage{xe "filtrage "} peuvent permettre de travailler avec les fichiers Interleaf{xe "Interleaf"} (STagger for Interleaf) ou FrameMaker (STagger for FrameMaker{xe "STagger for FrameMaker"}), qui sont alors ramenés à des documents de travail au format RTF. De plus des macros sous Word{xe "macros Word"} permettent de préparer des fichiers particuliers comme les fichiers de ressources Windows{xe "ressources Windows"} (RC), les fichiers SGML{xe "SGML"}, ou d'autres formats. L'aspect de la fenêtre principale de l'outil est indiqué dans la Figure 1.

1.2 Le processus de traduction sous Workbench

1.2.1 Phases préparatoires

Avant de se lancer dans la traduction d'un document, il faut s'assurer que deux étapes essentielles ont été accomplies : la définition d'une mémoire de traduction{xe "définition d'une mémoire de traduction (Workbench)"}, et la définition d'une base terminologique{xe "définition d'une base terminologique (Multiterm)"}.

Créer une nouvelle mémoire de traduction

Définition de la mémoire

Cela consiste sous Workbench à créer une nouvelle mémoire vide, et à définir ses paramètres. Le choix des paramètres essentiels porte sur le couple de langues traitées et le nom de la mémoire de traduction. Les autres paramètres (réglables soit dans la fenêtre de création soit à partir de la commande "File/Setup") concernent essentiellement :

- la gestion du projet de traduction{xe "gestion du projet de traduction"}
  ⇒ mot de passe
  ⇒ classification par client
  ⇒ type de domaine
  ⇒ etc.
- les paramètres textuels

⇒ règles de segmentation : pour la ponctuation en particulier
⇒ parties de textes à ne pas traduire marquées à l'aide d'un style Word
⇒ spécification des polices de caractères par langue
⇒ spécification du transfert de polices d'une langue à l'autre

Import de fichiers alignés

La mémoire ainsi créée est vide. Il est possible de la remplir avec des fichiers alignés manuellement, ou à l'aide de l'outil d'alignement{xe "alignement"} TAlign{xe "TAlign "}.

Définition d'une base terminologique

La base terminologique est facultative pour la traduction. Il n'existe pas sous Workbench de dictionnaire utilisable directement pour la traduction. Les bases terminologiques que Workbench peut utiliser doivent être définies par l'utilisateur, sous le logiciel cousin Multiterm{xe "Multiterm"}. La structure de ces bases est assez libre. Aussi l'utilisateur n'est pas tenu d'entrer les informations linguistiques du mot (comme le genre, ou la catégorie grammaticale, ou le lemme). La seule information nécessaire est la forme de surface du mot en langue source et cible. Si la mémoire de traduction{xe "mémoire de traduction"} est bilingue (source et cible), les bases terminologiques elles, peuvent contenir plusieurs langues. Une fois la base terminologique créée, un index analogique{xe "index analogique (Multiterm)"} doit être calculé (c'est une fonction de Multiterm), et la base qui doit rester activée sous Multiterm, est appelée par lien DDE lors de la traduction sous Workbench.

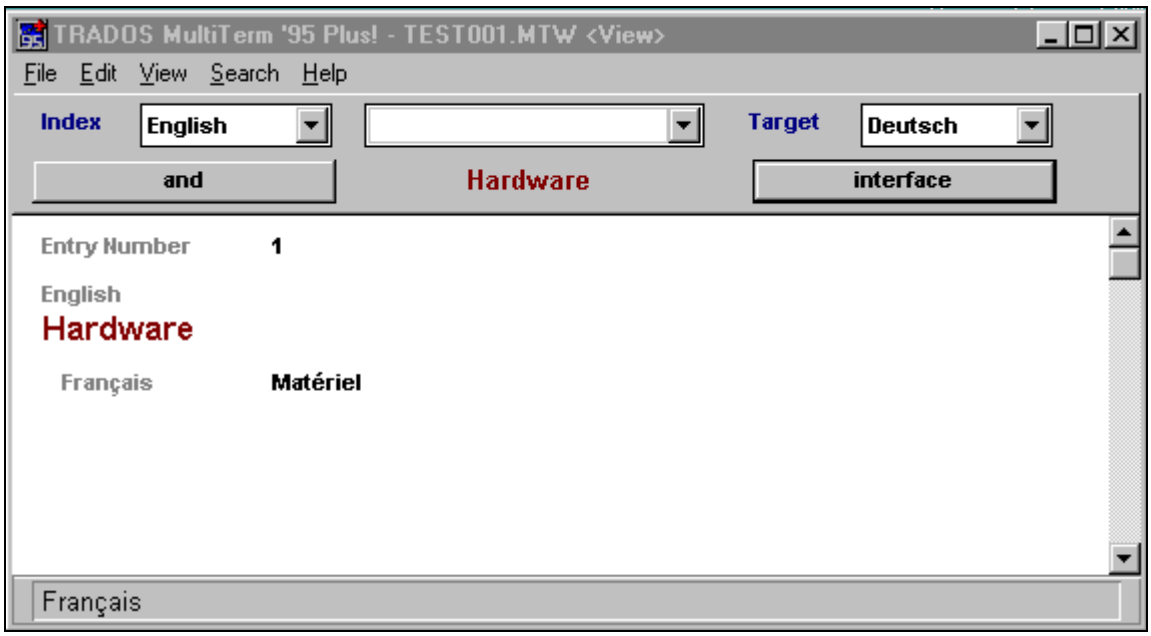


Figure 2 : Affichage d'une base terminologique sous Multiterm

1.2.2 La traduction

Introduction

Il y a deux modes de traduction{xe "traduction (Workbench)"} sous Workbench : la pré-traduction{xe "pré-traduction (Workbench)"}, et la traduction avec interaction{xe "traduction avec interaction (Workbench)"}. La première permet de pré-traduire un document, ou un groupe de documents pour que les traducteurs ou réviseurs puissent les compléter, sous Microsoft Word, même sans posséder Workbench. La seconde permet au traducteur de travailler en interaction, toujours sous Microsoft Word, et de bénéficier de toutes les informations disponibles provenant de la mémoire et de la base terminologique. Le traducteur doit alors posséder une licence de Workbench .

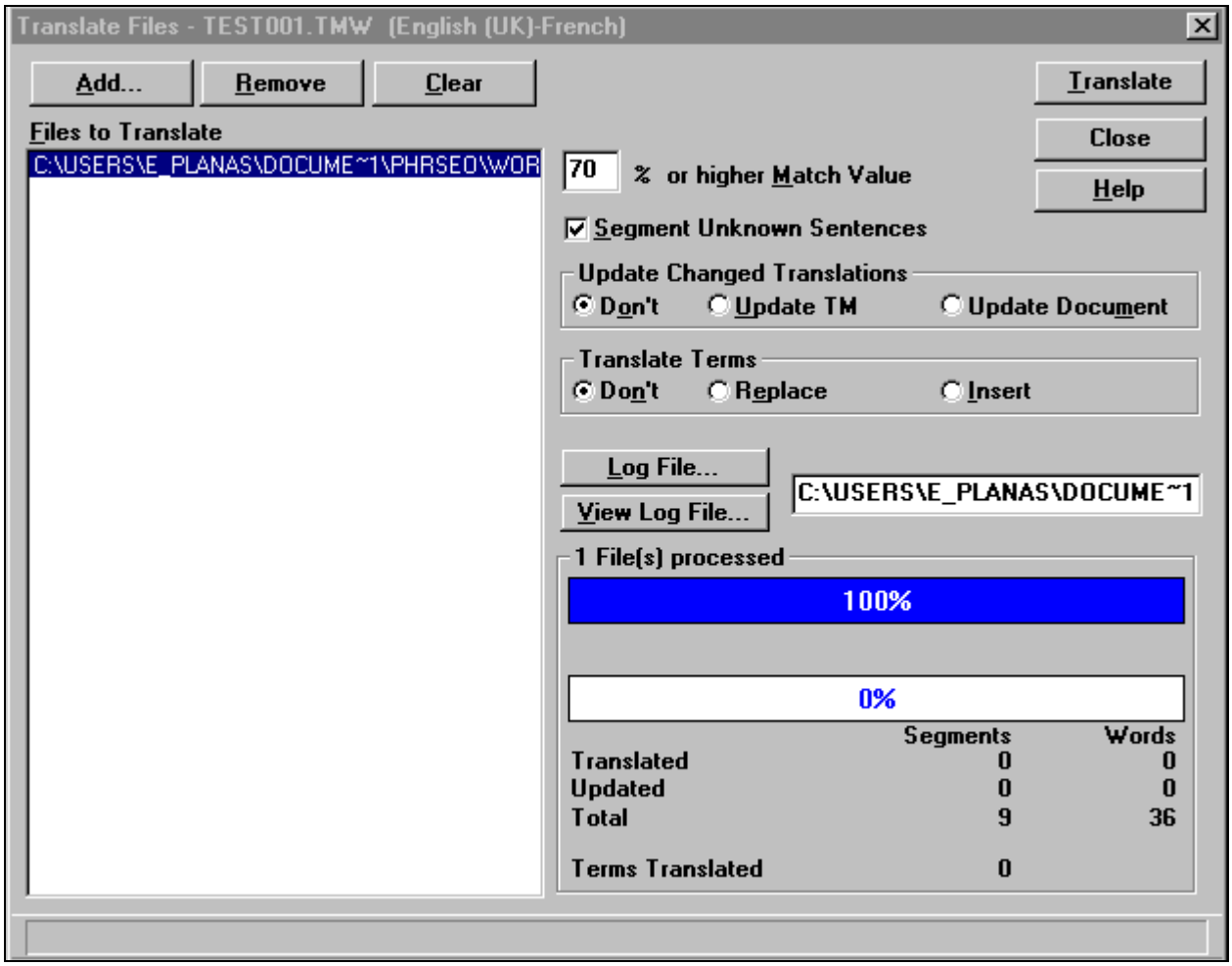


Figure 3 : Fenêtre de pré-traduction automatique de Workbench

La pré-traduction automatique

Une fois la mémoire créée et remplie, la traduction automatique {xe "pré-traduction"} peut enfin commencer. Pour ceci, on règle les paramètres, on indique quels fichiers (au format RTF ou DOC obligatoirement) doivent être traduits, et on lance la traduction automatique. Lors de la traduction, une jauge indique la part du document (en cours de traitement) qui a été réalisée, une autre le pourcentage de documents traités par rapport au lot complet de documents à traiter. Le nombre de segments{xe "segments"} et le nombre de mots correspondants qui ont été traduits, ceux mis à jour d'une précédente traduction, et le total des segments et des mots sont aussi indiqués (cf. Figure 3).

Les paramètres réglables concernent à la fois le seuil de traduction{xe "seuil de traduction"} accepté, le taux de pénalité pour des différences provenant du formatage{xe "formatage"}, ou pour une traduction provenant d'un alignement avec TAlign ou WinAlign (par opposition à un alignement effectué en traduction interactive, ou après une post-traduction (cf. plus haut), et le sort à réserver aux différents segments de traduction selon leur historique. La Figure 4 montre ces différentes options. Les couleurs d'affichage dans le document de travail résultant de la pré-traduction sont précisées dans une autre fenêtre.

Figure 4 : Options de traduction pour Workbench

Une fois le document traduit, les traducteurs peuvent l'éditer sous forme d'un document de travail avec Microsoft Word (6 ou 7), ou WordPerfect (6). Le document se présente sous la forme d'une suite de bi-segments{xe "bi-segments"} (source et cible) séparés par le pourcentage de correspondance du segment relatif trouvé dans la mémoire{xe "mémoire"} (Figure 5). Des couleurs permettent de distinguer les phrases traduites à 100% (exact match), entre le seuil{xe "seuil de traduction"} minimal choisi et 99% ("fuzzy match", traduction approchée{xe "traduction approchée"}), ou non traduites. Dans ce dernier cas, le segment à traduire est recopié dans la partie droite du document de travail, pour que le traducteur puisse l'éditer.

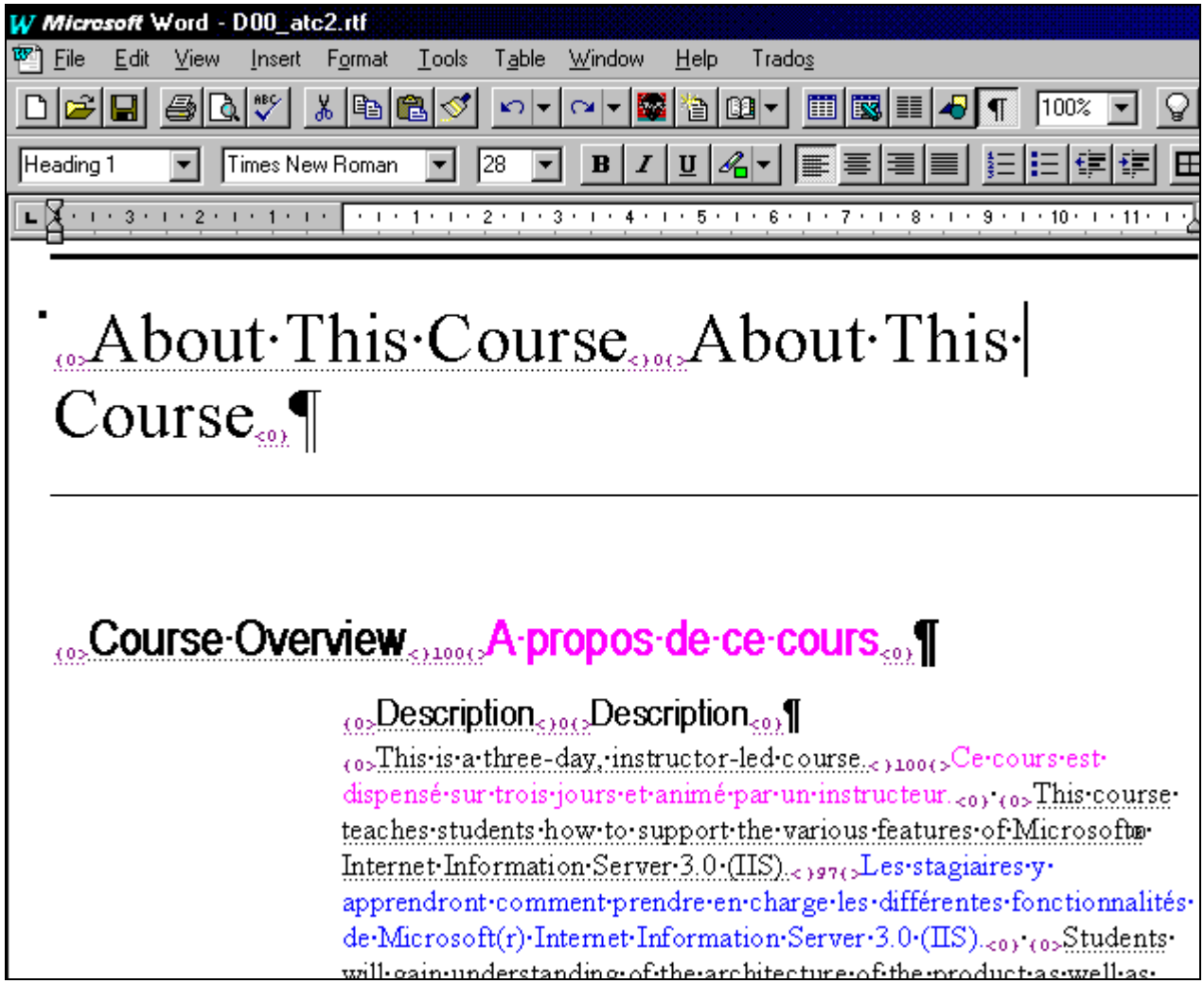


Figure 5 : Document pré-traduit par Workbench

Les parties gauches (c'est-à-dire le texte d'origine), ainsi que les marques insérées par Workbench sont formatées en style caché. Ainsi, en appliquant le style caché, le traducteur peut avoir une vision exacte de la forme finale du document. Si des objets imbriqués{xe "objets imbriqués"} sont présents dans le segment à traduire, cela est indiqué dans la fenêtre de résultat de la pré-traduction, mais ces objets ne sont pas placés par le processus de traduction. De même, certains formatages ne sont pas traités.

La traduction interactive

La traduction peut être réalisée de façon interactive{xe "traduction interactive"}, sous Word, à l'aide d'une barre de macros{xe "macros"} appelant des DLL de Workbench. Cette façon de procéder nécessite donc que le traducteur possède une licence de Workbench.

Ce procédé permet d'accéder à toutes les informations disponibles concernant les mémoires de traduction, la terminologie et les objets imbriqués, via les boutons de la barre des macros de Workbench (légende (1) sur la Figure 6 suivante).

Le traducteur demande la traduction automatique de la phrase courante en appuyant sur le deuxième bouton de la barre des macros de Workbench. Alors les informations correspondant à cette phrase s'affichent sous forme d'un bi-segment{xe "bi-segment"} et à l'aide de fenêtres de couleur (2). Le traducteur peut éditer la solution dans la partie droite de l'unité de traduction (qui possède une couleur particulière).

La fenêtre du haut est celle de Word (3){xe "Word"}. La barre des macros{xe "macros"} de Workbench est située juste au dessous de la barre standard de Word (4). La fenêtre inférieure gauche est la fenêtre principale de Workbench activée pour la traduction interactive (5){xe

"traduction interactive"**}**. La sous-fenêtre supérieure (6) montre l'ensemble des mots qui sont disponibles dans la base terminologique de Multiterm pour le segment actuellement traité, la sous-fenêtre de droite indique quel est le mot activé (7), et la fenêtre située dans le coin inférieur droit de la figure montre l'entrée du mot courant sous Multiterm (8). La sous-fenêtre inférieure de Workbench donne l'entrée correspondante de la mémoire et sa traduction (9). De plus quelques paramètres sont indiqués sur la gauche de la fenêtre de Workbench (10). Lors de cette traduction, le responsable du projet peut choisir de mettre à jour automatiquement la mémoire de traduction**{**xe "mémoire de traduction : mise à jour automatique"**}**. Dans ce cas, chaque phrase éditée par le traducteur sera intégrée à la mémoire de traduction.

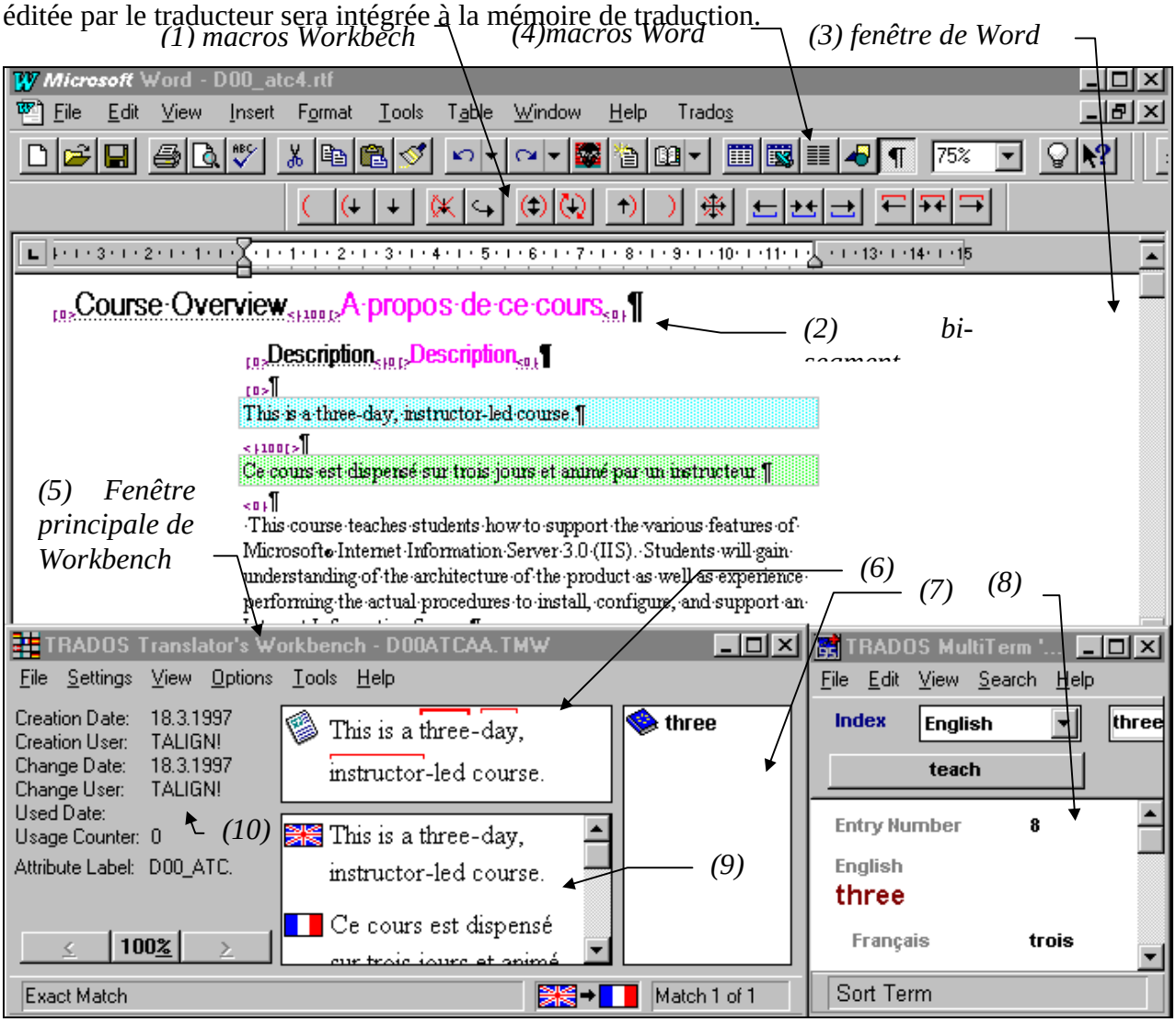


Figure 6 : Traduction interactive avec Workbench.

La post-traduction{xe "post-traduction"}

Pendant la traduction, Workbench ajoute des marques qui permettent l'affichage et l'interaction. Une fois le fichier traduit et révisé, ces marques doivent être retirées pour que le fichier retrouve sa forme originale. Cela est assuré par la commande "Tools/Cleanup". Le résultat est un fichier similaire au fichier d'origine, mais traduit, et enregistré au format RTF.

1.3 Les mémoires de traduction

1.3.1 Leur structure

Physiquement, la mémoire**{**xe "mémoire (Workbench)"**}** correspond à un fichier texte d'extension ".tmw", plus quatre autres fichiers dont un index fondé sur un réseau neuronal**{**xe

"réseau neuronal"**}** (d'extension ".ann" pour le fichier principal, plus ".six" et ".iix" pour les fichiers annexes).

Le fichier ".tmw" contient les bi-segments**{**xe "bi-segments"**}** (source et cibles), qui sont liés au fichier ".ann". Les bi-segments sont simplement juxtaposés, comme le montre la Figure 7. Dans le préambule du fichier sont indiqués les paramètres correspondant à la mémoire, plus des renseignements sur les polices de fichier RTF.

Le format trouvé lors de l'alignement (via TAlign**{**xe "TAlign"**}**, par interaction) est aussi stocké, comme il est trouvé dans le fichier RTF. Cela signifie donc qu'il n'y a pas d'abstraction du formatage. Ainsi même si TAlign est capable d'aligner des fichiers provenant d'autres éditeurs comme Ventura ou Interleaf, les mémoires ainsi réalisées ne pourront être utilisées que sur des documents au même format. Il n'est pas possible de constituer une mémoire**{**xe "mémoire"**}** incluant le formatage à partir de documents RTF, et d'utiliser cette mémoire pour traduire des documents Interleaf !

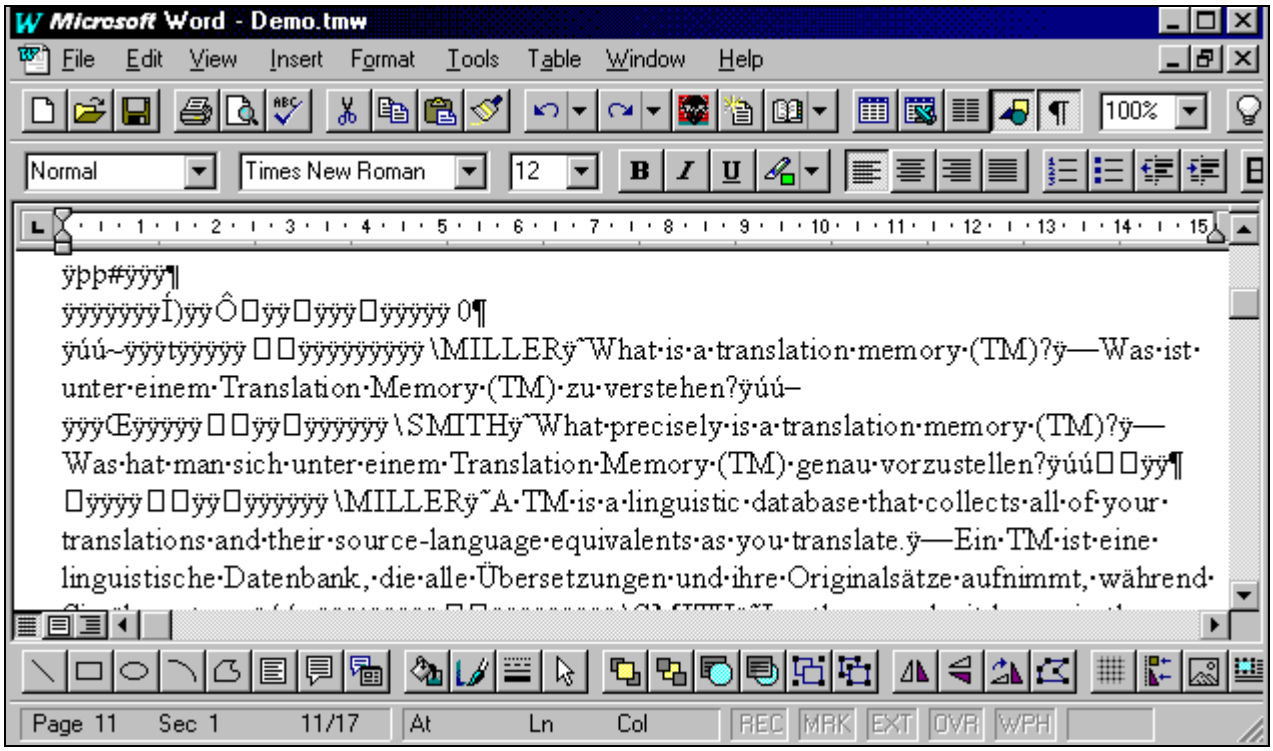


Figure 7 : le fichier mémoire ".tmw" de Workbench

Les fichiers d'extensions ".ann", ".six" et ".iis" constituent l'index de la mémoire**{**xe "index de la mémoire (Workbench)"**}**, et permettent une recherche rapide de correspondances. La structure de ces fichiers est cryptée. On peut cependant deviner que le réseau est construit sur les occurrences des lettres, ou plutôt des caractères ASCII, et structuré selon des "groupes" de nœuds rassemblés par caractères.

1.3.2 La représentation du formatage

Introduction

Le format est représenté comme il est codé dans le document d'origine. Ainsi la Figure 8 montre la structure d'une mémoire formée à partir d'un document RTF. Le formatage des mots est indiqué par des accolades, et une marque par format. Ainsi l'application du formatage italique à "Multiterm" est exprimé dans la mémoire comme ceci:

{¥i Multiterm}

Ce qui est similaire à la représentation du formatage sous RTF. Le même raisonnement est appliqué pour les fichiers au format WordPerfect, sous le logiciel WordPerfect.

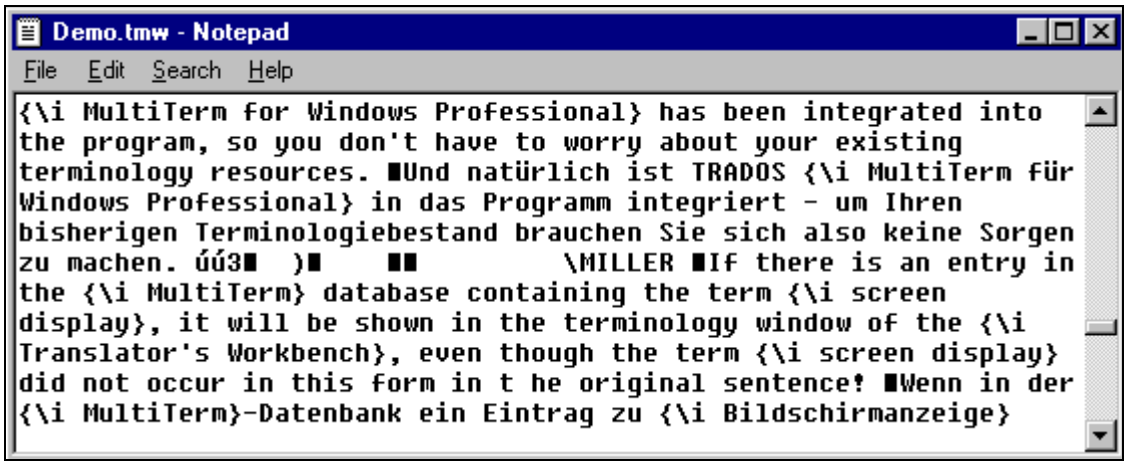


Figure 8 : Représentation du formatage sous Workbench.

Pour les autres formats, Workbench adopte trois stratégies différentes, suivant les groupes de formats suivants : fichiers directement lisibles sous Word, fichiers importables via des filtres de Microsoft Word, fichiers importables via des filtres spécialisés distribués par Trados.

Fichiers lisibles sous Word

Cela concerne principalement les fichiers qui dans leur état de base sont au format “texte”, ou qui après l'application d'un filtre ont été transformés en un fichier texte possédant des balises :

- DCF
- RC
- SGML
- Troff
- Ventura

Le principe consiste alors à appliquer des “macros” (fournies dans un “fichier modèle”-template en anglais - avec le logiciel) sur ce texte ASCII, qui imposent un “style” pour isoler et “cacher” le codage de ces balises. Ces “codes” seront alors traités comme des objets que le traducteur doit replacer dans le segment traduit.

Fichiers importables via des filtres Word

Dans ce cas, les fichiers sont importés avec les filtres de Word, vers le format Word ou RTF, et Workbench les traite sous format RTF. Les codes de formats sont alors ceux de RTF.

Fichier importables par des filtres distribués par Trados

Trados distribue deux filtres{xe "filtres"} particuliers, originellement mis au point par la société ITP de Dublin. L'un permet de convertir les fichiers Interleaf{xe "Interleaf"} en RTF, l'autre les fichiers FrameMaker{xe "FrameMaker"} en RTF. Le filtre traduit donc la structure de ce type de fichiers en RTF basique, et ajoute des balises (comme dans SGML) cachées ou pas pour retranscrire le formatage originel de ces fichiers.

Les balises{xe "balises"} sont alors traitées comme des objets imbriqués, c'est-à-dire qu'elles sont proposées, mais leur placement n'est pas assuré automatiquement dans le cas de modifications par rapport à ce qui est dans la mémoire.

1.3.3 Edition de mémoires de traduction

Il est possible d'éditer les mémoires de traduction{xe "mémoires de traduction (Workbench)"} via l'interface, à l'aide de la fonction "Tools/Concordance". Cette fonction permet de rechercher les segments de la mémoire qui correspondent à un groupe de mots entrés au clavier.

Un fois les correspondances trouvées, chaque unité de traduction est éditable (source et cible). Bien sûr, dans le cas de langues à caractères non latins (arabe, japonais), il faut disposer d'un kit d'entrée des caractères de la langue pour pouvoir écrire cette langue (posséder les polices adéquates permet de visualiser ces langues, mais pas de les éditer).

Il n'y avait pas de moyen de visualiser l'ensemble des unités de traduction sous un éditeur dédié jusqu'à la version 1.14. WinAlign permet depuis la fin décembre 1997 de le faire. La seule solution était alors d'exporter la mémoire en format texte, de l'éditer sous un traitement de texte quelconque, et de l'importer à nouveau. La Figure 9 montre l'aspect de la mémoire exportée au format texte.

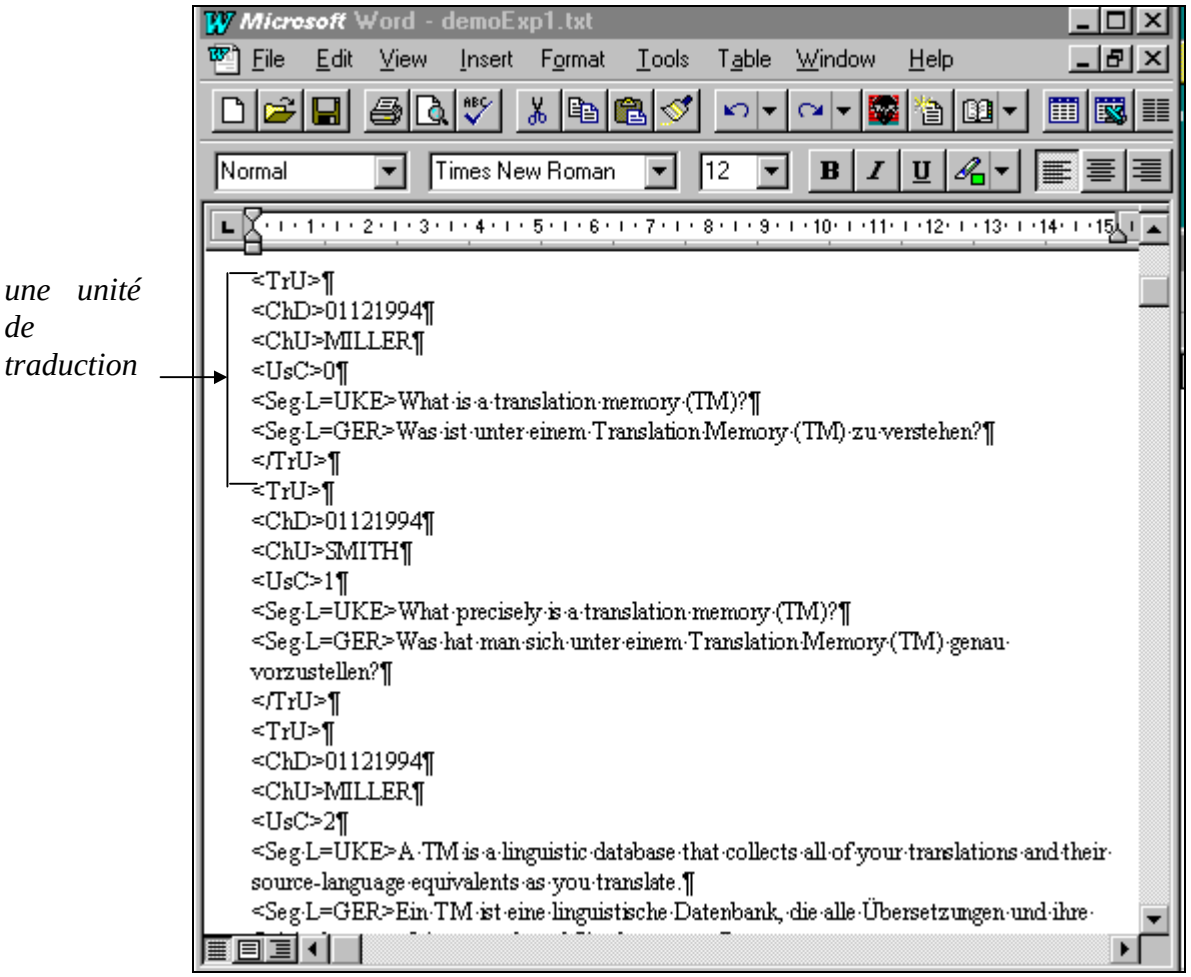


Figure 9 : fichier mémoire ".tmw" exporté par Workbench au format ASCII

1.4 L'alignement automatisé : TAlign

Il existe un programme sous DOS qui permet d'aligner deux documents (un source, un cible), nommé TAlign{xe "TAlign"}. Ces deux document peuvent posséder différents formats, mais il semble que le format le mieux traité pour l'alignement{xe "alignement (TAlign)"} soit RTF. Depuis décembre 1997 cet aligneur est disponible sous Windows (WinAlign), et rend l'alignement beaucoup plus pratique.

La procédure fournit des fichiers alignés prêts à être importés dans une mémoire de Workbench.

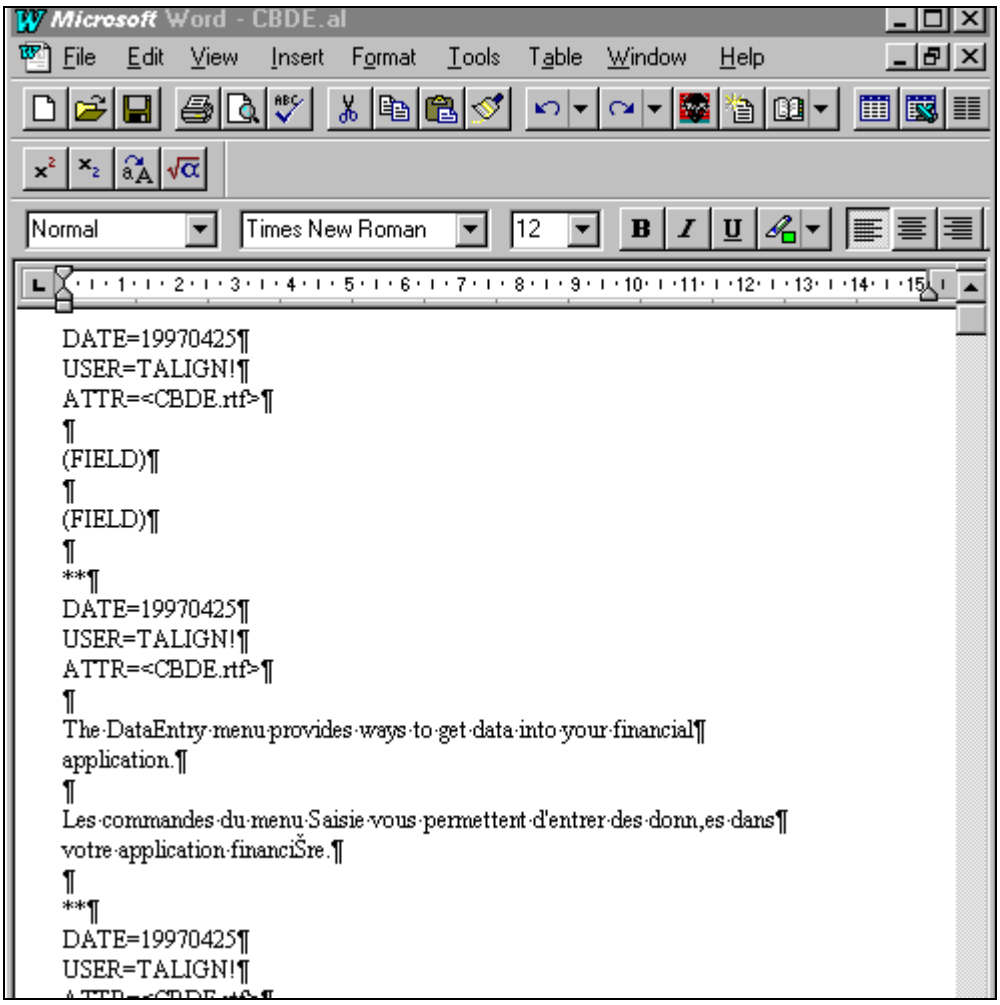


Figure 10 : Fichier aligné avec TAlign

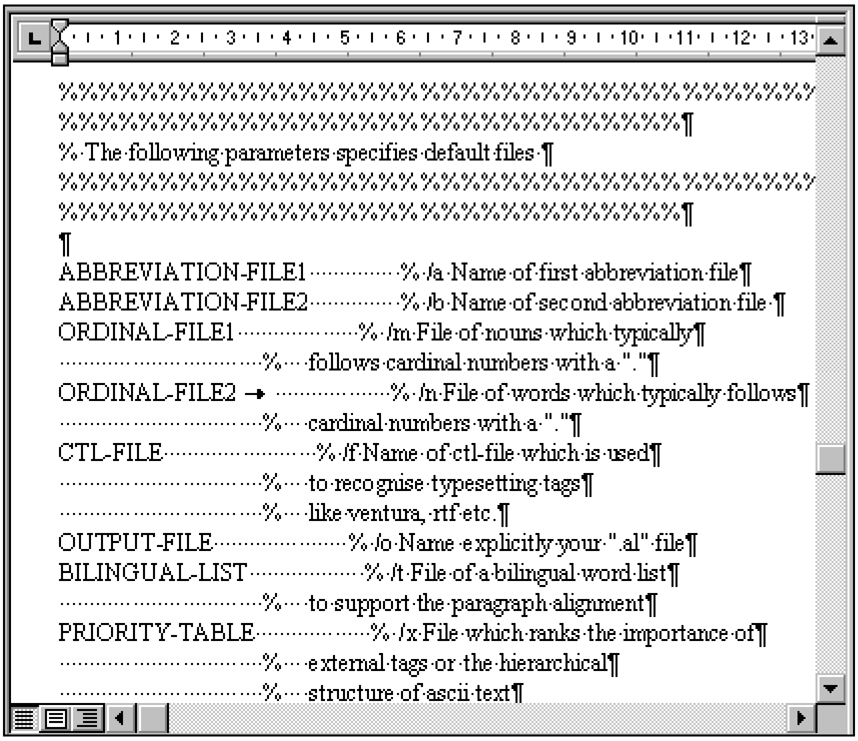


Figure 11 : Le fichier de réglage des paramètres pour TAlign

Le taux d’alignement est souvent satisfaisant, mais dépend grandement de la structure du fichier. Aussi une “préparation” des fichiers est souvent nécessaire. La Figure 10 donne une idée de la structure de ce fichier. Dans la Figure 10 par exemple, l’éditeur utilisé pour la mémoire est Microsoft Word. La version Windows permet maintenant d’éditer ces fichiers. TAlign est hautement paramétrable à travers le fichier TAlign.cfg pour la version DOS (cf. Figure 11). Il est ainsi possible d'attribuer un taux d’importance à la prise en compte des balises, des espaces, de la ponctuation, des chiffres, etc.

TAlign permet d’utiliser une liste bilingue de termes comme indices de correspondance. Cela étant, cela reste toujours une recherche de correspondance strictement statistique, les structures linguistiques des phrases correspondantes ne sont pas utilisées.

1.5 La terminologie sous Multiterm

La recherche de mots dans Multiterm**{**xe "Multiterm"**}** se fait aussi de façon statistique. Il est donc possible de trouver des mots proches, ou des formes différentes d’un lemme (manger, manges, mangeons) par approche statistique. Cependant, il n’existe pas de moyen rigoureux de définir les lemmes et les formes fléchies correspondant aux mots. Il est possible d’indiquer les informations linguistiques correspondant aux termes de la base (catégorie grammaticale, genre, nombre), de la même façon que l’on peut donner d’autres renseignements (provenance du terme, fréquence d’utilisation), mais cette information linguistique n'est pas utilisée par Workbench.

Par contre, elle peut l’être par le logiciel de traduction par règles Transend, qui peut être couplé à Workbench (cf. ci après). Workbench n’est pas capable de remplacer une forme de lemme par une autre, et nous verrons dans les exemples qui suivent que si “button” et “bouton” se correspondent dans la base terminologique**{**xe "terminologie"**}**, il n’est pas possible de traduire “buttons” par “boutons” !

1.6 Connexion à un outil de traduction automatique fondée sur les règles : Intergraph Transend

1.6.1 Généralités

Workbench peut être couplé avec un outil de traduction fondée sur les règles, Transend{xe "Transend"} de la société Intergraph{xe "Intergraph"}. L'utilisation de ce module de traduction est très simple. Soit un segment (une phrase) à traduire. Workbench regarde d'abord les solutions possibles dans la mémoire. Si le taux de correspondance est inférieur au seuil fixé par l'utilisateur, Workbench passe la main à Transend qui fournit une traduction automatique pour le segment. Cette solution doit bien sûr souvent être post-éditée.

1.6.2 Utilisation de la terminologie stockée sous Multiterm

Il est à noter que Transend peut utiliser la terminologie stockée dans Multiterm (le dictionnaire de Workbench), pourvu que l'utilisateur ait entré les informations grammaticales nécessaires dans la base correspondante de Multiterm{xe "Multiterm"}. Ces informations concernent :

- la catégorie grammaticale (Part Of Speech) : adj, adv, v, n, nm, nf, nn)
- le genre et le nombre : f, fpl, m, mpl, n, npl

et doivent respecter une syntaxe assez peu contraignante :

- le nom du champ doit commencer par "Gram" pour la catégorie grammaticale (Grammaire, Grammar, Grammatik, et Gram sont donc acceptés)
- le nom du champ doit commencer par "Gen" pour le genre et le nombre (Genre, Gender, Genus et Gen sont acceptés)

1.7 Travail en équipe, à travers le réseau

Workbench est utilisable en réseau{xe "traduction en réseau (workbench)"}. Cela signifie que les fichiers de mémoire peuvent résider sur un serveur auquel se connectent les utilisateurs. Une gestion basique du blocage d'une mémoire pour administration et la définition de "login" utilisateurs est aussi prévue. L'administrateur des mémoires de traduction peut alors directement intervenir sur le serveur pour gérer les bases phraséologiques. Les mêmes fonctionnalités réseau sont disponibles pour les bases terminologiques sous Multiterm.

1.8 Deux fonctionnalités originales : "Analyse" et "Mise à jour"

1.8.1 Analyse

Lorsqu'un document est à traduire, on peut se demander si cela vaut la peine de passer par l'outil de TFM ou pas. Workbench apporte une solution à cette question avec la commande "Tools/ Analyse". Cette commande permet d'effectuer le même travail que la pré-traduction, pour un ensemble de fichiers. La différence est qu'elle n'affecte pas les fichiers. En revanche, elle fournit un journal qui détaille les traductions possibles, par tranches de pourcentage (100, 95-100, 85-95, etc.), et par fichiers. Cela permet donc d'évaluer les gains de production{xe "Analyse et gains de production (Workbench)"} possibles.

Un point très intéressant réside en le fait que l'on peut aussi demander la création d'un fichier des segments fréquents{xe "segments fréquents"} non présents dans la mémoire, et dont l'occurrence dépasse un nombre ajustable (>2). Ce fichier, une fois traduit, peut alors être directement injecté dans la mémoire pour l'augmenter, et cela permet à la fois de gagner du temps par le traitement de cette intra-redondance{xe "intra-redondance"} (voir le chapitre 1), mais aussi de garantir l'homogénéité{xe "homogénéïté"} du document à traduire.

1.8.2 Mise à jour

Une première traduction ayant été réalisée, il est possible de traduire à nouveau automatiquement le document, et donc de mettre à jour le document.

Si cette deuxième traduction se fait sans ajout extérieur de nouveaux éléments dans la mémoire, cela peut permettre d'uniformiser la traduction de certaines phrases qui auraient pu être traduites différemment par deux traducteurs travaillant sur le même gros projet.

Si il y a ajout de mémoire entre temps, il peut s'agir d'une mise à jour interne au projet, ce qui arrive fréquemment pour des traductions de logiciels. Cela fonctionne avec les segments provenant de la mémoire, mais pas avec la terminologie, ce qui est dommage car les mises à jour**{**xe "mise à jour"**}** de terminologie sont fréquentes.

2. Translation Manager d'IBM[2]

2.1 Généralités

Translation Manager{xe "Translation Manager"} est édité par IBM{xe "IBM"}. Ce logiciel permet de traduire par mémoire des fichiers enregistrés sous plusieurs formats (IBM Bookmaster, RTF, Interleaf{xe "Interleaf"}, FrameMaker{xe "FrameMaker"}, WordPerfect{xe "WordPerfect"}, Amipro, HTML2,...), pour un grand nombre de langues (cf. Annexe 2.3.).

Le logiciel peut fonctionner sous les systèmes OS/2{xe "OS2"} et Windows 95, et Windows NT{xe "Windows 95"}. La gestion des dictionnaires et des mémoires se fait sous le même logiciel. L'utilitaire d'alignement{xe "alignement (Translation Manager)"} "Eqfitm.exe" permet d'obtenir une mémoire à partir d'un document et de sa traduction. Notre étude se base sur la version 2.0.7.

L'organisation générale de la fenêtre principale se compose de différentes sous-fenêtres dynamiques (cf. Figure 12). Celle des dictionnaires montre les différents dictionnaires bilingues disponibles. Translation Manager (TM) possède en effet par défaut des dictionnaires bilingues qui servent à l'analyse et à la traduction des phrases. La fenêtre des mémoires montre les mémoires disponibles. En cliquant sur un dictionnaire ou une mémoire, on accède à une fenêtre de recherche dans laquelle on peut trouver un terme ou un segment de mémoire, et l'éditer.

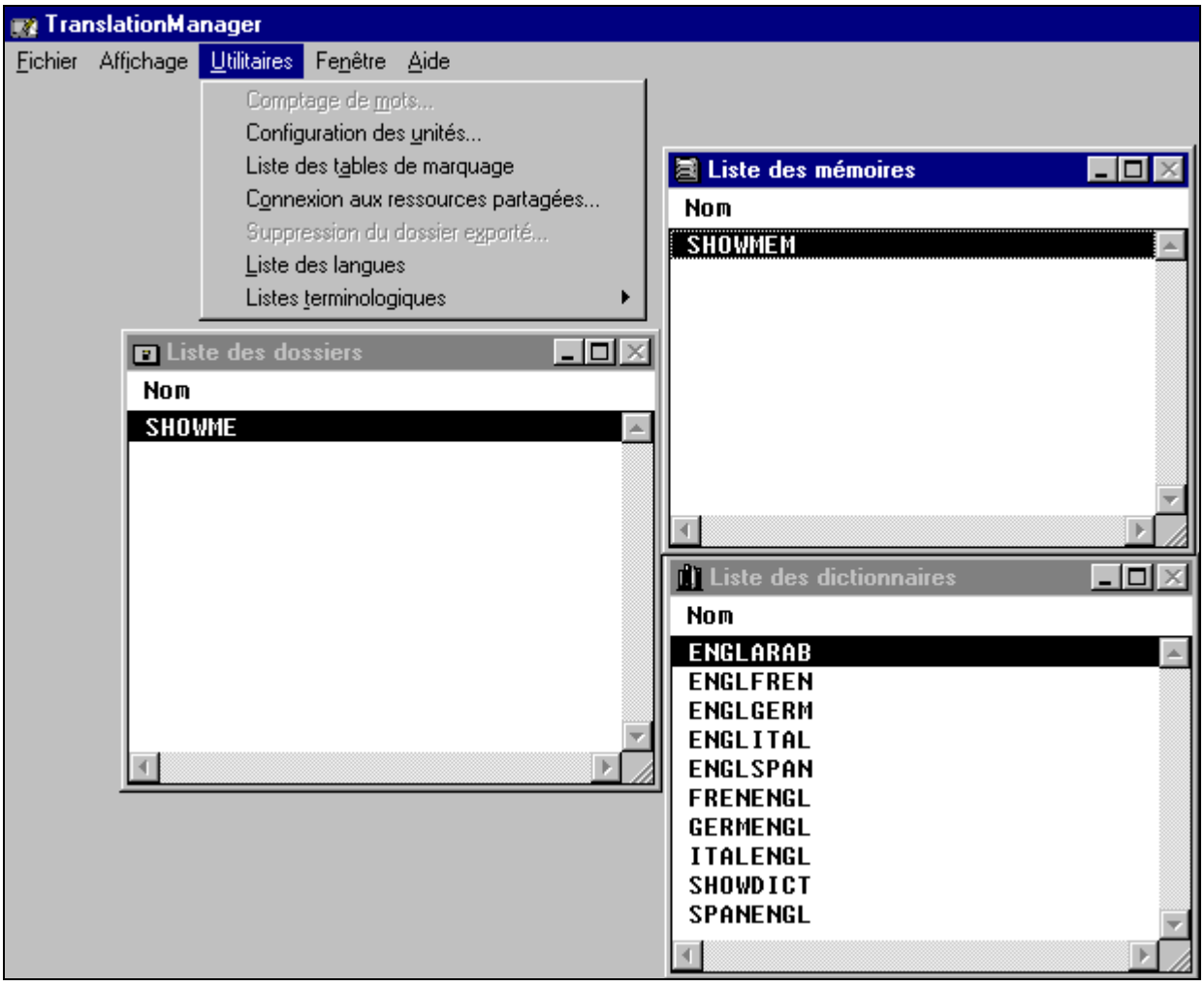


Figure 12 : fenêtre principale de Translation Manager

[2] Une licence gratuite de deux mois est disponible sur le site interne d'IBM consacré à Translation Manager. Cette possibilité a été précisée par le service client d'IBM Translation Manager, qui doit ici être remercié. Ce site se trouve à l'adresse http://www.praetorius.com/.

La fenêtre des “dossiers” montre les dossiers disponibles. Un clic sur un dossier ouvre une nouvelle fenêtre qui présente les fichiers à traduire du dossier. Le menu est dynamique : il s’adapte à la position de la souris dans les différentes fenêtres.

2.2 Le processus de traduction sous Translation Manager

2.2.1 Phases préparatoires

Création d’un dossier de traduction

Translation Manager est structuré par dossiers. Ainsi, avant de traduire un document, il faut créer un dossier{xe "dossier de traduction (Translation Manager)"} qui spécifie les caractéristiques requises. Cela se fait en appelant la fonction “Fichier/Nouveau” (cf. Figure 13). Le fichier du dossier est alors obligatoirement situé dans un répertoire fixe de Translation Manager, se trouvant à la racine du disque. Outre le nom du dossier, une mémoire de traduction existante doit obligatoirement y être associée. Un dictionnaire{xe "dictionnaires (Translation Manager)"} peut aussi l’être. Le format des fichiers qui seront inclus dans le dossier est à préciser.

Une fois cela réalisé, les fichiers à traduire peuvent être intégrés au dossier. Il est possible d’insérer un fichier enregistré dans un format différent de celui du dossier, et de lui attribuer des langues source et cible différentes.

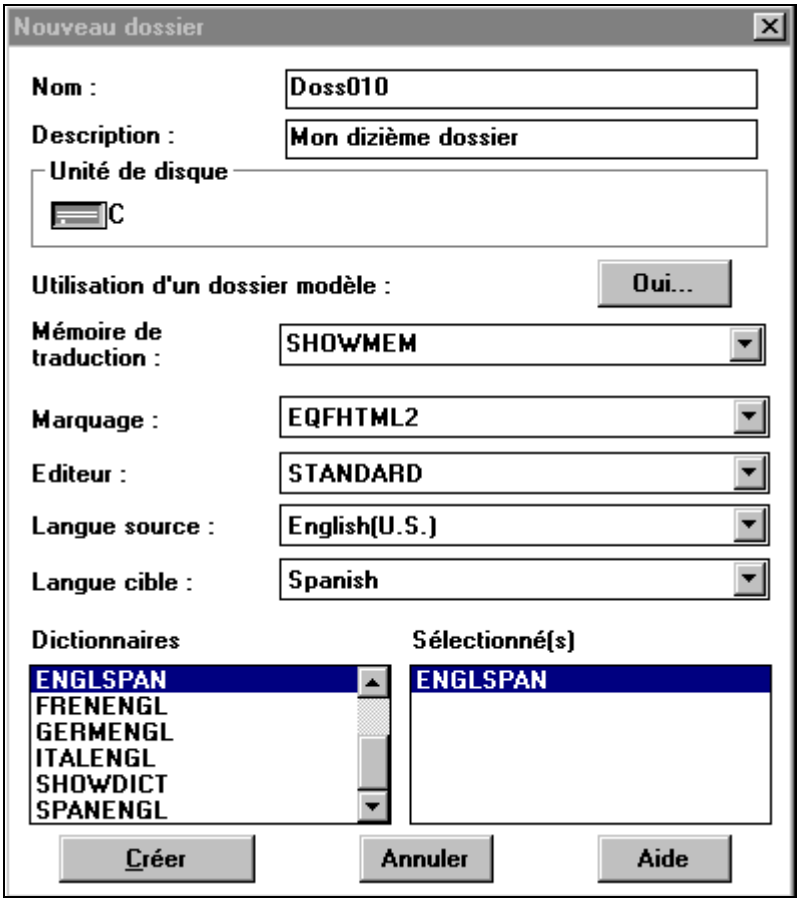


Figure 13 : Création d’un nouveau dossier sous Translation Manager

Créer une nouvelle mémoire de traduction

Définition de la mémoire

Il est bien sûr possible de créer ses propres mémoires{xe "mémoire (Translation Manager)"} de traduction. Cela se fait en accédant à la fenêtre des mémoires, et en actionnant la commande “fichier/nouveau”. Peu d’options sont à préciser. Parmi elles, on doit indiquer si la mémoire à

créer sera partagée ou non pour un travail en équipe à travers le réseau. La mémoire créée est alors vide.

Remplissage de la mémoire

La gestion des mémoires permet d'importer une mémoire dans une autre (pourvu que les paramètres se correspondent), ou de fusionner deux mémoires. En outre la création de nouvelles mémoires par alignement{xe "alignement (Translation manager)"} de fichiers parallèles est possible grâce à un module d'alignement (voir plus loin).

Créer un nouveau dictionnaire

Translation Manager propose des dictionnaires{xe "dictionnaires (Translation manager)"} par défaut pour les principaux couples de langues. Cela étant, il est bien sûr possible de créer son propre dictionnaire. Il y a cependant quelques restrictions : la langue source doit être une des cinq langues FIGES[3]. La structure principale d'une entrée est imposée, mais on peut renommer certains champs, et en ajouter d'autres. Lors de l'ajout de termes, on n'est pas tenu de préciser la catégorie grammaticale ou d'autres informations linguistiques.

2.2.2 La traduction

Introduction

La traduction se fait dans un l'éditeur propriétaire de TM. Cela signifie que chaque fichier extérieur à traduire doit être filtré pour qu'il puisse être affiché sous l'éditeur de TM, et une fois sa traduction réalisée, il doit être enregistré au format de départ.

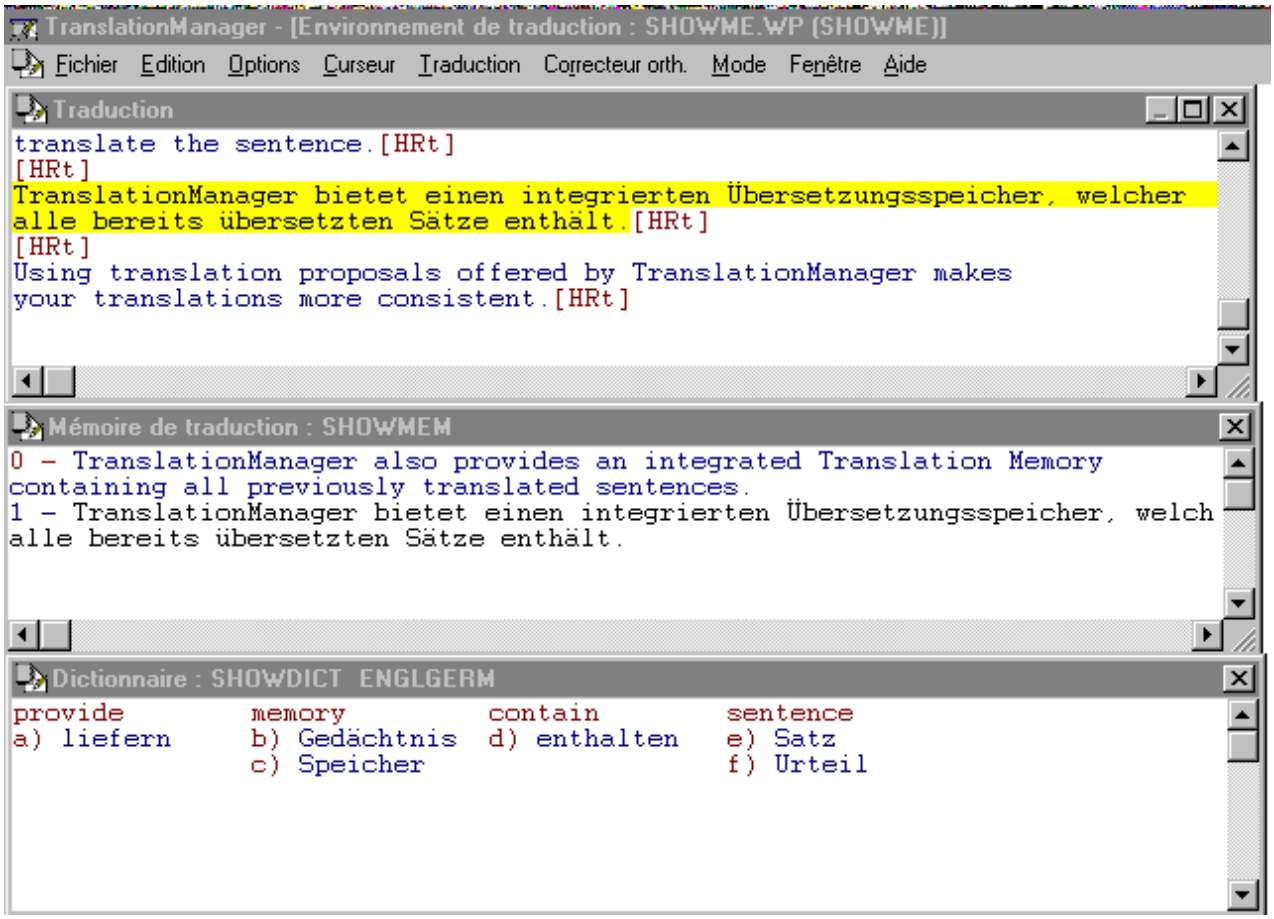


Figure 14 : La fenêtre d'édition de Translation Manager

Translation Manager (IBM) possède donc autant de filtres{xe "filtres (Translation manager)"} aller et retour que de formats traités.

[3] French Italian German English and Spanish

La segmentation et l'analyse de départ

Un dossier ayant été créé, la première phase de la traduction consiste donc à formater le document, à le segmenter**{**xe "segments (Translation Manager)"**}** en unités de traduction, puis à évaluer pour chacune de ces parties si la mémoire possède un segment équivalent. Cette phase s'appelle l'analyse, et est conduite dès que l'on clique deux fois sur le fichier que l'on veut traduire. Une fois celui-ci pré-traduit, la fenêtre d'édition du fichier apparaît, ainsi que les fenêtres de la mémoire et du dictionnaire correspondants (Figure 14). La fenêtre des fichiers du dossier (cachée sur la Figure 14 par les fenêtres nouvellement apparues) montre le pourcentage de phrases traduites automatiquement.

La traduction interactive

Il est alors possible de se déplacer de segment en segment pour finir la traduction**{**xe "traduction interactive (Translation Manager)"**}**, en se servant des segments proposés par la mémoire, et du vocabulaire proposé par le dictionnaire. Ces éléments sont facilement incorporables dans le texte à l'aide de raccourcis claviers. S'il y a plusieurs solutions, elles sont présentées par ordre de pertinence, et numérotées. Le taux de correspondance entre un segment à traduire et le segment relatif de la mémoire de traduction n'est pas indiqué.

La traduction par lots

Il est possible de traduire en une seule fois une série de fichiers, et de demander l'introduction automatique des seuls segments à correspondance exacte. L'exportation de ces fichiers pré-traduits peut fournir des fichiers partiellement traduits que les traducteurs peuvent alors éditer sous l'éditeur d'origine, sans avoir besoin de posséder TM. Les phrases traduites ont simplement remplacé les phrases sources, et cœxistent avec les phrases non traduites, sans marquage spécial. Il n'est donc pas possible de récupérer automatiquement des phrases partiellement correspondantes. Il n'y a pas de réglage des paramètres influençant la recherche de segments similaires dans le processus de pré-traduction**{**xe "pré-traduction"**}**.

2.3 Les mémoires de traduction

2.3.1 Leur structure

La mémoire**{**xe "mémoire (Translation Manager)"**}** est constituée d'un fichier de base de données crypté (cf. Figure 15).

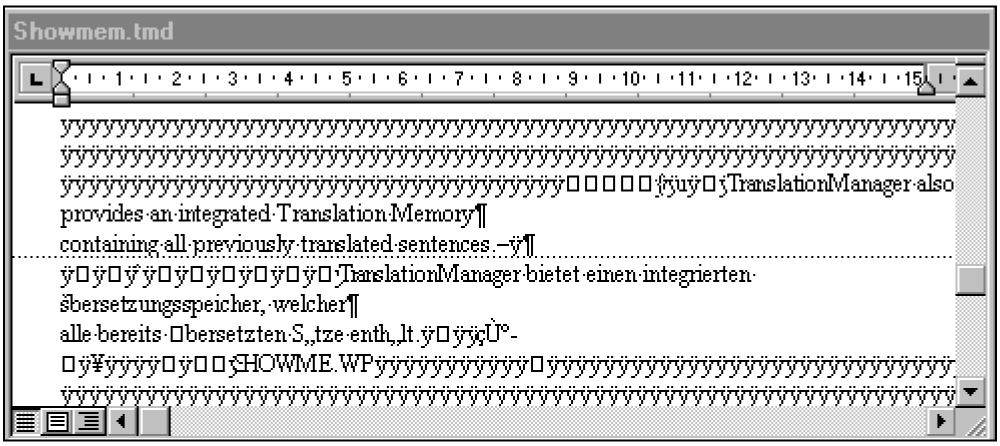


Figure 15 : mémoire ".tmd" brute de Translation Manager

L'exportation de cette mémoire au format ASCII montre que sa structure suit les spécifications de SGML, comme on le voit sur la Figure 16.

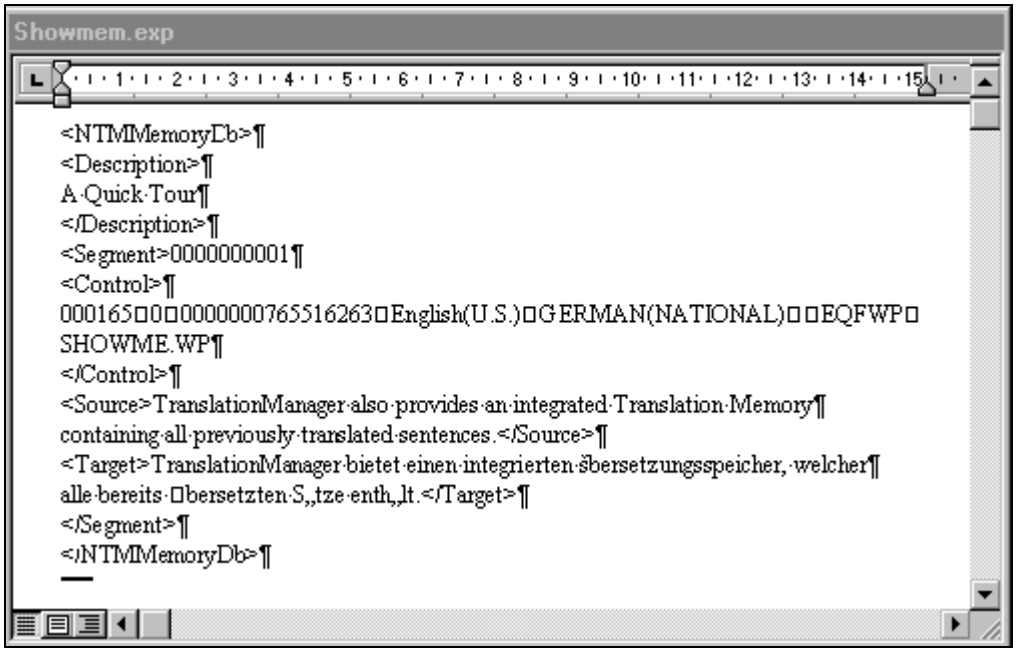


Figure 16 : mémoire ".tmd" exportée au format DOS ASCII

La mémoire consiste donc en une suite de bi-segments (sources et cibles) non structurés, c'est-à-dire dont on donne seulement les chaînes de caractères. Bien sûr, les paramètres correspondant à chaque segment ainsi qu'un en-tête de paramètres généraux sont stockés dans la mémoire. Le fichier mémoire d'extension ".tmd" est accompagné d'un fichier d'index de même nom, et d'extension ".tmi".

2.3.2 La représentation du formatage dans la mémoire de TM

Principe

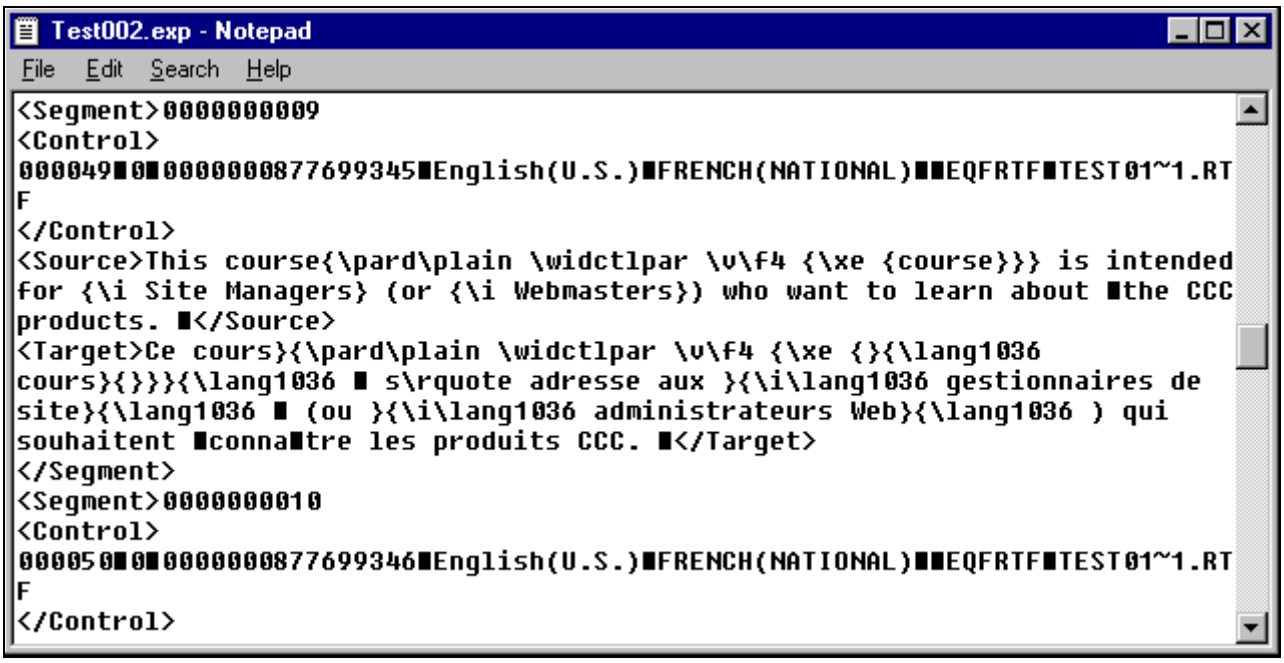


Figure 17 : Exportation en fichier texte d'une mémoire créée sous Translation Manager.

La Figure 17 montre un extrait de l'exportation d'une mémoire créée par l'aligneur**{**xe "alignement (Translation Manager)"**}** automatique de Translation Manager (eqfitm.exe), sur la base des deux fichiers tests anglais et français du chapitre 3 suivant (figures 1 et 2). Les unités de traduction sont isolées, et présentées sous un formalisme proche de SGML. Le contenu des balises

représentant le texte et la mise en forme du document de départ n'est cependant pas interprété, et reste proche du code de départ. On peut voir dans l'exemple de la Figure 17 que le marquage provient d'un document au format RTF.

Expérience

Essayons maintenant de nous servir de cette mémoire constituée à partir de documents RTF, pour traduire le même texte, enregistré sous un autre format de traitement de texte, comme HTML2 par exemple.

Dans cette expérience, le même texte, reprenant exactement les mêmes formats, a été créé sous HTML2 (Figure 18). Les balises d'index sur "course" n'ont pas été insérées, ainsi que la note de pied de page sur "structure". Par contre, tous les "gras", italiques, et styles de textes ont été appliqués.

Lorsque l'on applique la mémoire créée à partir de fichiers RTF sur ce même texte HTML, Translation Manager (IBM) ne trouve de correspondance exacte que pour les phrases qui ne comportent pas de format. Pour récupérer celles qui en comportent, il faut passer au mode interactif. Alors, lorsque l'on demande de recopier la proposition de la mémoire, le code RTF du formatage provenant du texte RTF est copié intégralement, ce qui n'a bien sûr aucun sens pour un fichier HTML (Figure 19).

Translation Manager n'est donc pas capable d'utiliser le formatage de texte d'une mémoire créée à partir d'un format particulier de document, pour un document de format différent.

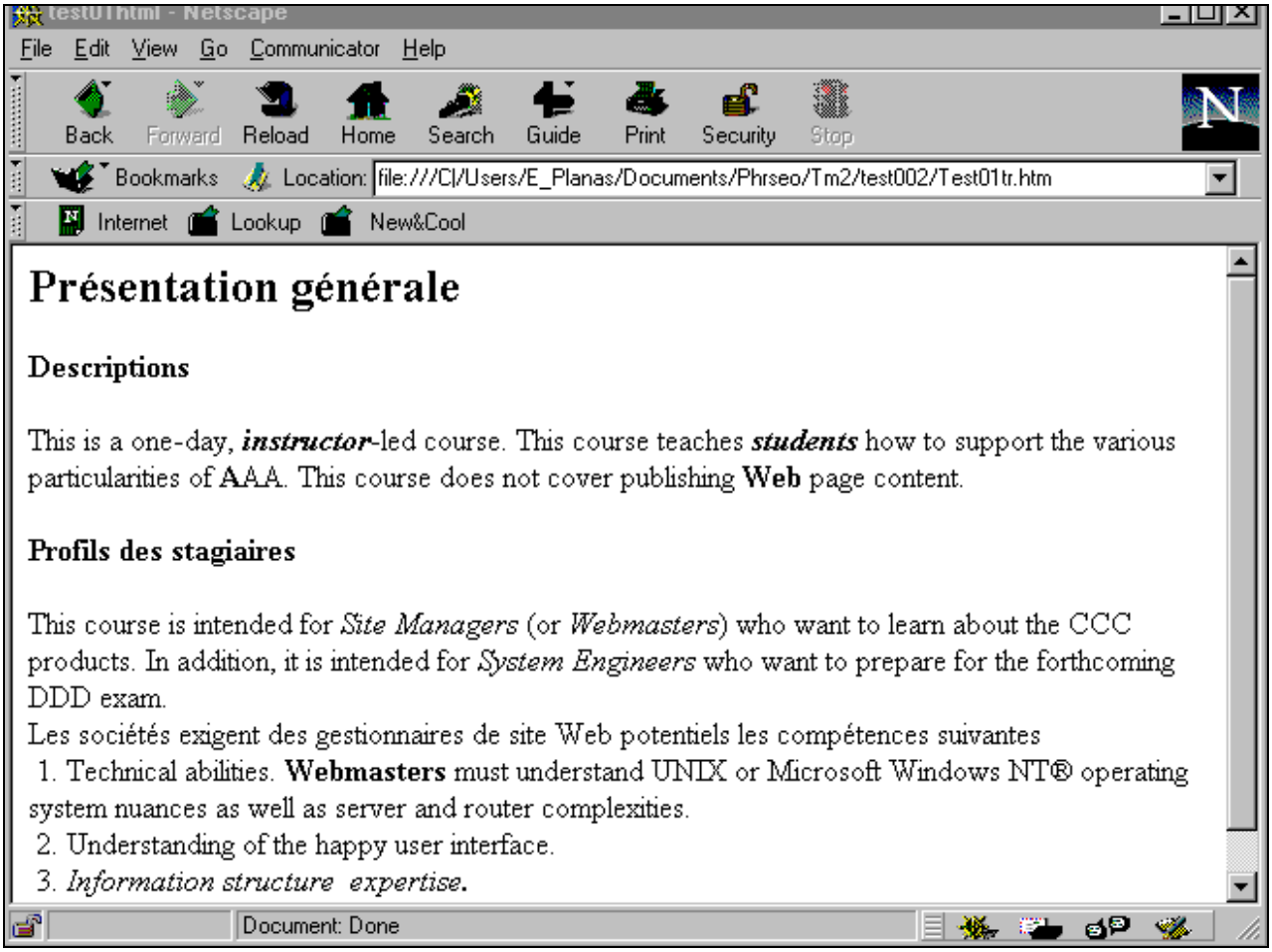


Figure 18 : Le même texte sous HTML

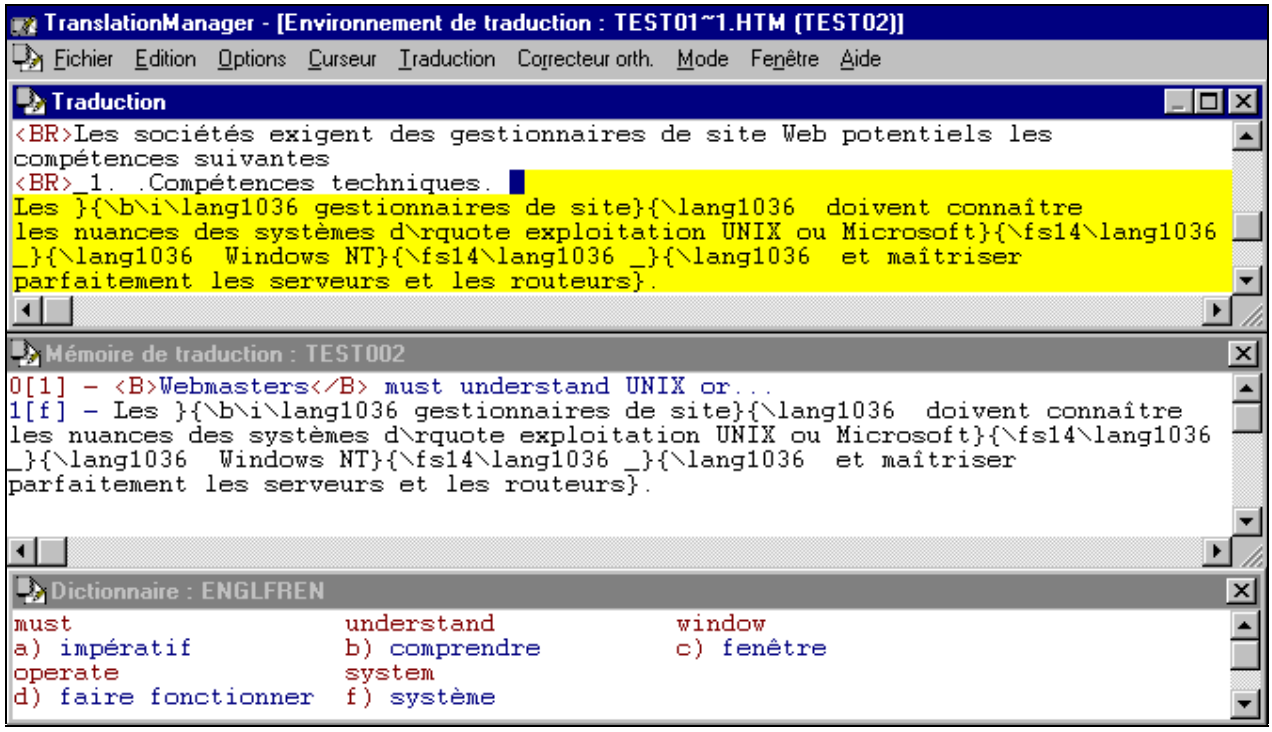


Figure 19 : Utilisation d'une mémoire RTF pour un fichier HTML

2.4 L'alignement automatisé : Eqfitm.exe

Un logiciel d'alignement{xe "alignement (Translation Manager)"}, Eqfitm.exe, accompagne Translation Manager, et permet de créer une mémoire de traduction à partir de deux ou plusieurs fichiers parallèles.

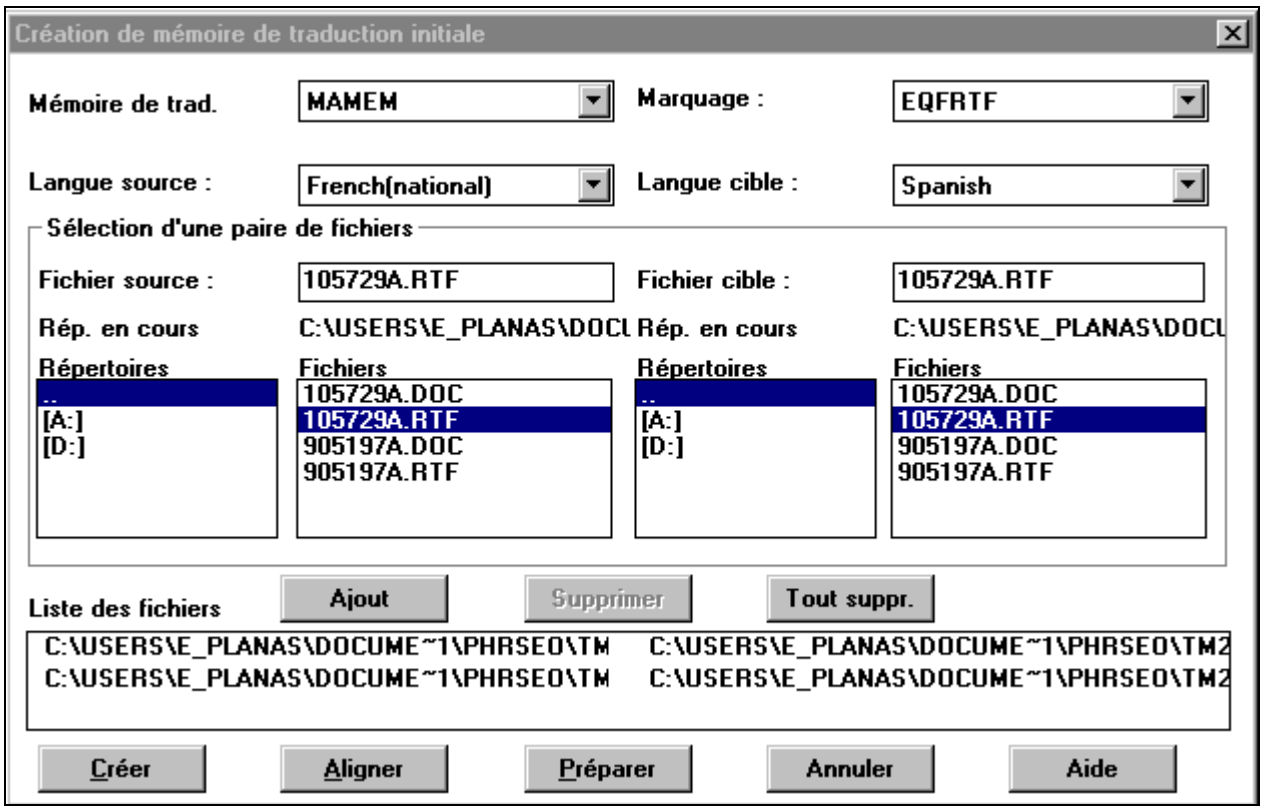


Figure 20 : fenêtre d'initialisation d'un alignement sous TM

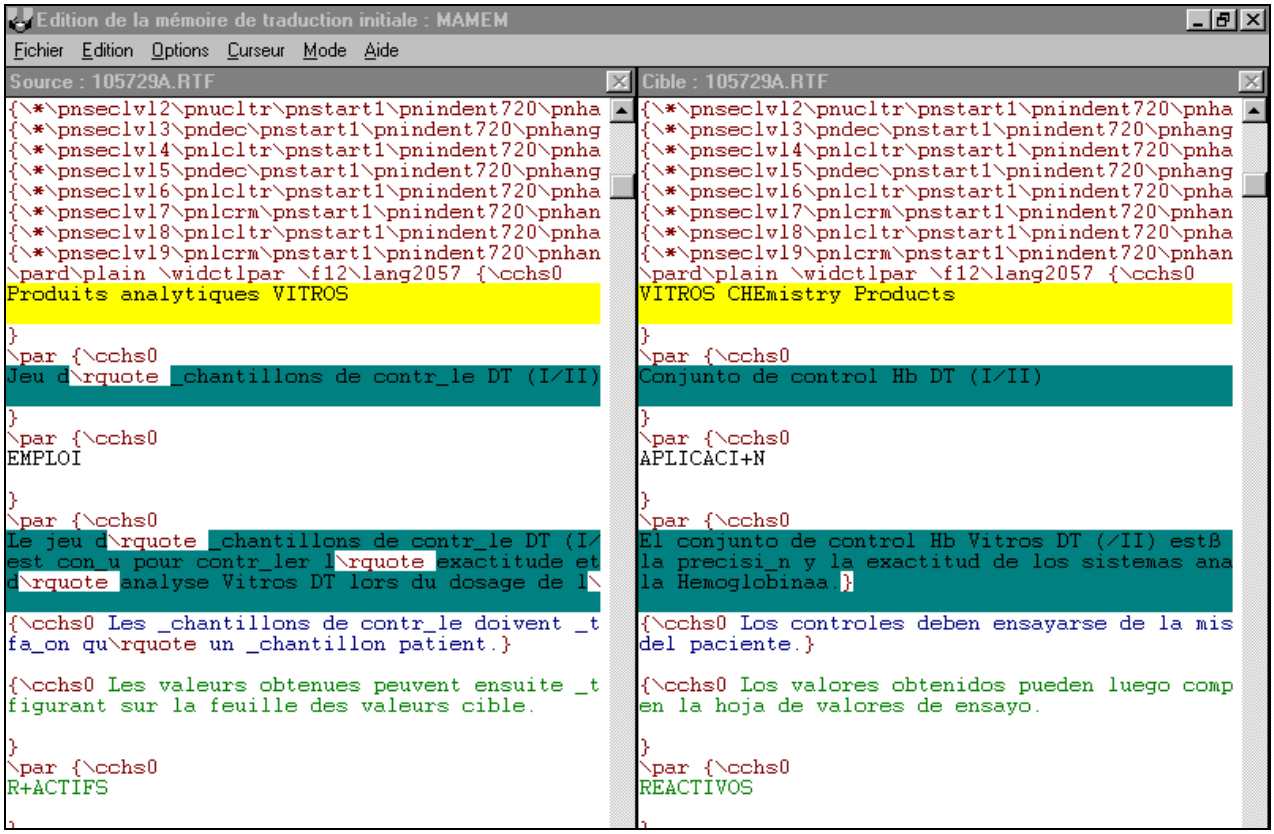


Figure 21 : Éditeur pour l'alignement sous Translation Manager (anglais vers espagnol).

L'interface de réglage des paramètres est claire, et permet de régler les langues source et cible, d'entrée des fichiers, et la mémoire de Translation Manager (IBM) dans laquelle les alignements seront intégrés. Une fois les paramètres et les fichiers choisis, le bouton "créer" permet de lancer l'alignement, et ouvre la fenêtre de l'éditeur (Figure 21) qui permet de modifier les alignements faux.

2.5 La Terminologie sous Translation Manager

La gestion des dictionnaires**{**xe "dictionnaires (Translation Manager)"**}** est assuré sous Translation Manager même, contrairement à Workbench et Transit qui se servent eux d'applications séparées (respectivement Multiterm et TermStar). Un dictionnaire est un fichier de type ".asd" accompagné d'un fichier d'index ".isd". De nouveaux dictionnaires peuvent être créés et remplis par l'utilisateur. La Figure 22 montre la fenêtre d'édition d'un terme d'un dictionnaire **{**xe "terminologie (Translation Manager)"**}** personnalisé.

2.6 Une fonctionnalité originale : "Analyse et création de listes utiles"

La fonction "Analyse**{**xe "pré-analyse"**}**", qui permet de traiter un lot de fichiers (cf. plus haut), permet de créer des listes intéressantes, dont en voici quelques-unes.

2.6.1 Création d'une liste de segments fréquents non traduits

Comme dans Workbench, il est possible de créer la liste des segments non présents dans la mémoire. On ne peut malheureusement pas choisir le seuil à partir duquel ces segments doivent être retenus. Ce sont donc tous les segments non traduisibles par la mémoire qui sont retenus, et pas seulement les plus fréquents.

2.6.2 Création d'une liste des mots nouveaux

Une fonctionnalité originale consiste à créer la liste des termes inconnus, et non plus des segments. Cela peut être très utile dans la préparation d'un dossier de traduction, d'une part pour

assurer l'homogénéité de traduction en proposant la terminologie adéquate, et d'autre part pour permettre aux terminologues d'anticiper les questions des traducteurs.

2.6.3 Liste des termes connus

Une liste des termes connus peut être constituée, par exemple pour être donnée comme support aux traducteurs qui ne possèdent pas TM.

2.6.4 Liste des termes connus d'un dictionnaire particulier

La même liste peut être créée relativement à un dictionnaire donné.

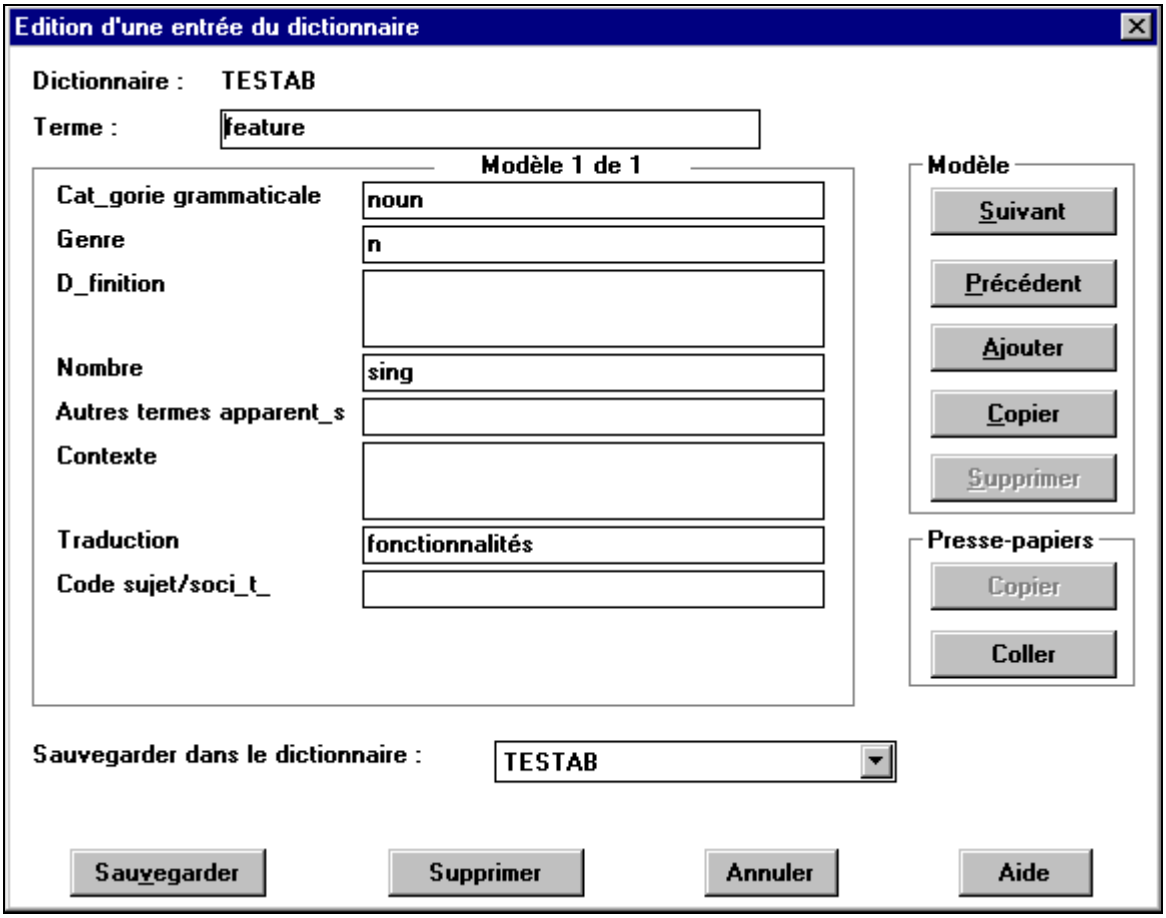


Figure 22 : Edition d'un terme de dictionnaire sous Translation Manager

3. Transit de STAR[4]

3.1 Généralités

Cet OTFM1g, appelé Transit{xe "Transit"}, est couplé avec un outil de gestion terminologique indépendant : TermStar{xe "TermStar"}. La logique architecturale est donc la même que pour les outils de Trados. L'outil fonctionne sous Windows 95{xe "Windows 95"}. Nous avons testé la version 2.6. La philosophie du produit est avant tout d'être une interface unique pour la traduction de documents d'origines diverses. Ainsi l'éditeur propriétaire utilisé pour la traduction pouvant convertir depuis et vers des fichiers de différents types, il peut être aussi utilisé simplement en tant qu'éditeur{xe "éditeur cmmun"} commun. Cela permet de pouvoir éditer un fichier de type Interleaf sans posséder Interleaf, ou un fichier texte provenant d'un Macintosh,{xe "Macintosh"} sous un PC. Les fonctionnalités d'import et d'export sont raffinées comme cela est décrit ci-après. Une des particularités de Transit est qu'il permet de traiter également les langues asiatiques, en source comme en cible. Ces langues comprennent le chinois, {xe "chinois"}le japonais{xe "japonais"}, le coréen{xe "coréen"} et le thai{xe "thai"}.

3.2 Le processus de traduction sous Transit

3.2.1 Création d'un "Projet"

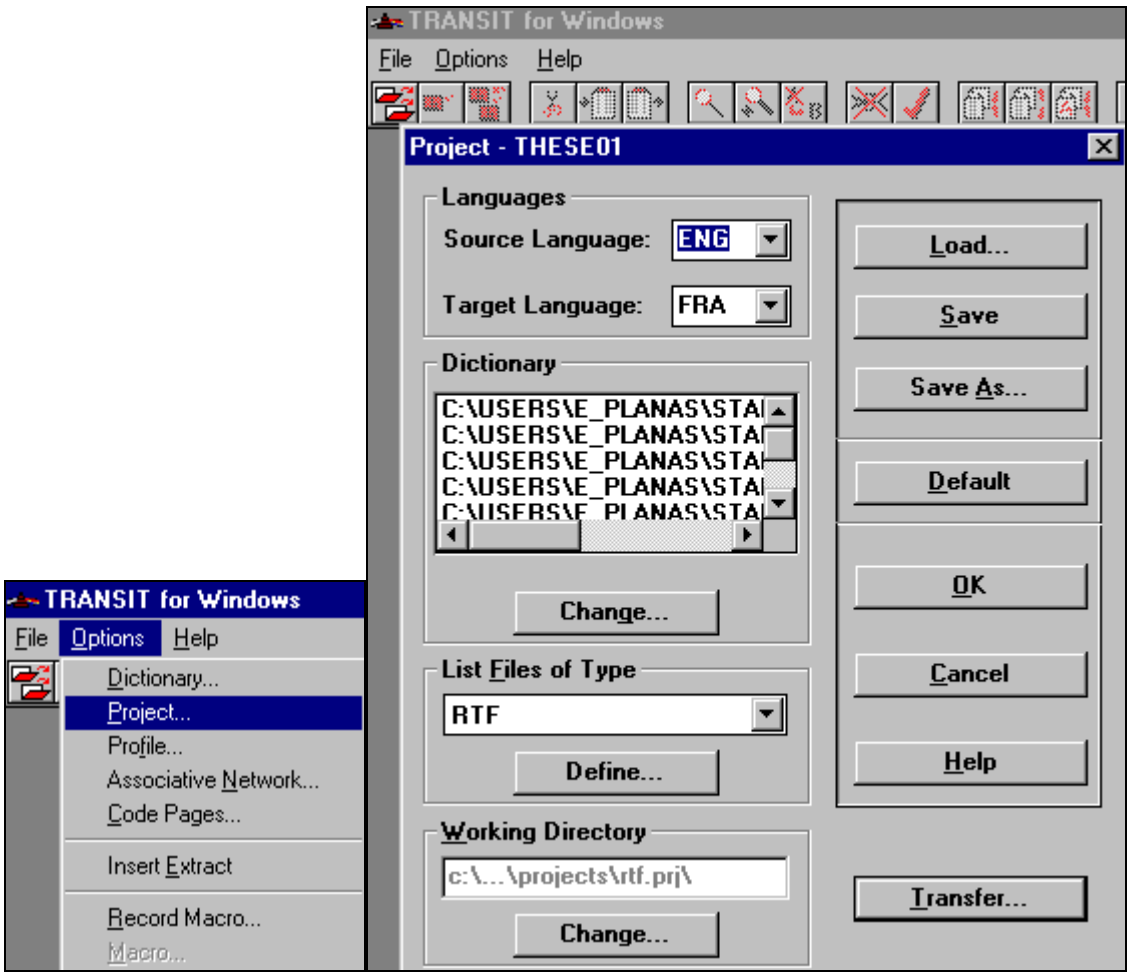


Figure 23 : Création d'un projet de traduction avec STAR Transit

[4] STAR™ Japan a aimablement contribué à cette thèse en fournissant une version gratuite de Transit™ et TermStar™. Je tiens particulièrement à remercier ici Madame Michiko Kagami, et Monsieur Yasuyuki Ueda pour leur constant support. STAR Japan Co. Ltd., Matsuda Bldg. 4F, 3-14-13 Shiba, Minato-ku, Tokyo 10 (http://www.star-ag.ch/).

Avant toute manipulation de fichier, un “projet” doit être créé. La fenêtre de création d’un projet{xe "projet de traduction (Transit)"} est représentée en Figure 23. Ceci consiste à fixer les paramètres nécessaires au travail de traduction. Les cinq paramètres principaux sont :

- les langues sources et cibles
- l’ensemble des dictionnaires que l’on veut utiliser
- le répertoire de travail sur l’ordinateur
- le type de fichiers qu’il faut traiter
- les paramètres d’import et d’export.

Lors de la création du projet, on peut contrôler exactement les traitements qui seront effectués à l’importation et à l’exportation des fichiers, dans une fenêtre spéciale (Figure 24) appelée par le bouton “Transfert” de la fenêtre de création d’un projet (Figure 23).

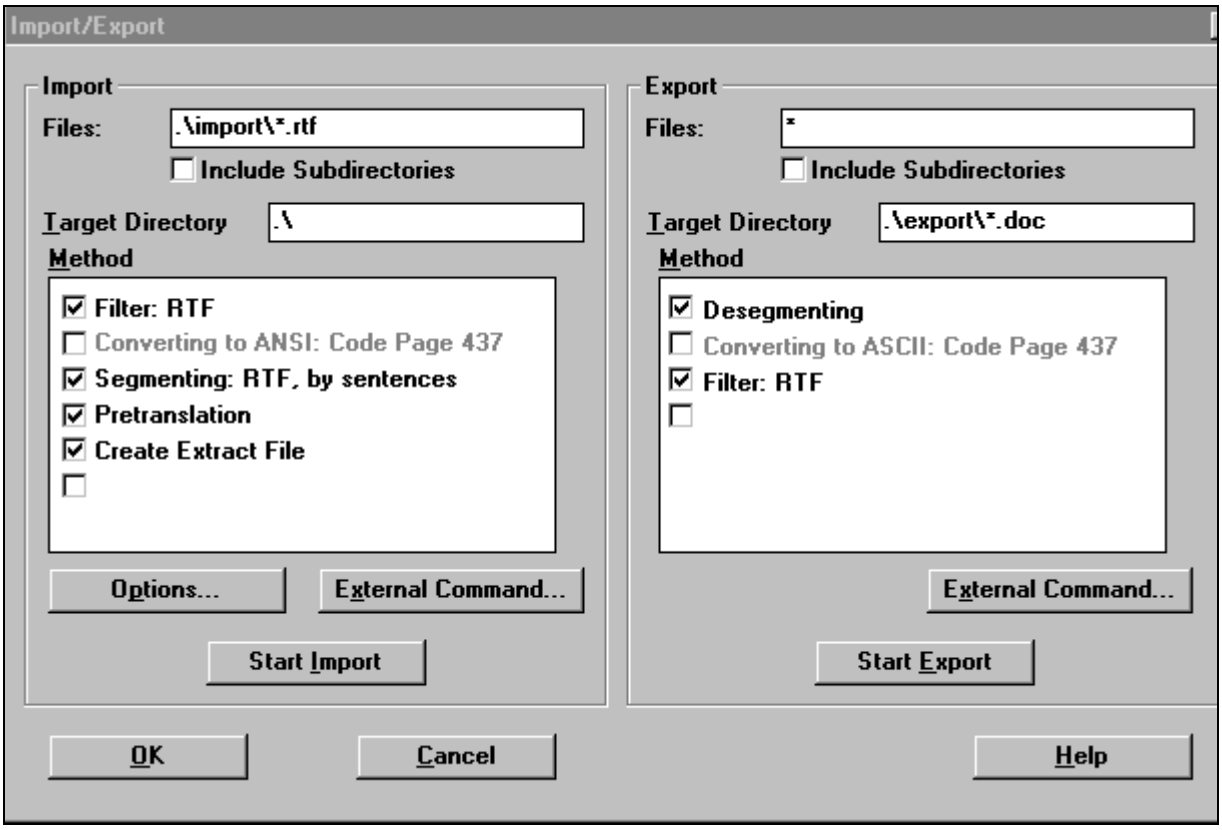


Figure 24 : Configuration des actions à l’import et à l’export

Ces traitements sont de six types :

- **interprétation** des fichiers pour **filtrer{**xe "filtres (Transit)"**} le format** du fichier à traiter vers un langage de représentation interne à Transit**{**xe "langage de représentation interne (Transit)"**}** (“Filter”). Dans cette phase, la structure du fichier original (RTF dans la figure) est analysée selon des règles stockées dans les filtres de Transit. Des règles supplémentaires d’analyse peuvent être spécifiées grâce à un langage de règles appelé “expressions régulières” (voir plus loin).
- éventuelle **conversion de table de caractères**. Cela permet de travailler sous Windows, sur des fichiers provenant d’autres systèmes, ou écrits avec d’autres tables de caractères (Macintosh, ASCII du DOS, ..).

- **segmentation{**xe "segments (Transit)"**}** des textes en unités de traduction (segments). Cela se fait selon des règles prédéfinies et modifiables. La Figure 25 donne un exemple de définition de règle de segmentation.
- **pré-traduction{**xe "pré-traduction (Transit)"**}** : cela permet d'insérer les traductions trouvées pour les segments déjà traduits que le "Réseau Associatif**{**xe "Réseau Associatif (Transit)"**}**" (la mémoire en fait, voir plus loin) a pu trouver. Cela est utile si le projet comporte une forte redondance interne (cf. B.3.4.1.1.).
- Création de fichiers d'**extraction de phraséologie** : une fois le fichier à traduire segmenté, l'ensemble des segments, ou (au choix) seulement ceux qui ne sont pas traduisibles par la mémoire sont regroupés dans un fichier, qui peut être ensuite traduit séparément pour créer une mémoire relative à ce projet.
- Une particularité de la programmation de Transit est que l'on peut spécifier des **commandes externes{**xe "commandes externes"**}** (commandes DOS ou Start Windows) à exécuter entre ou après l'une des étapes précédentes. Les caractères jokers et d'autres commandes sont disponibles, et des commandes comme celle qui suit peuvent être spécifiées :

Exemple : "copy {cfp}{cfn}.{cfe} c:¥backup", où {cfp}, {cfn}, et {cfe} sont le répertoire, le nom du fichier et l'extension du fichier courant. Cette commande peut être utile pour créer une copie de sauvegarde des fichiers traités après une étape, par exemple.

On notera qu'une certaine cohérence de traitement est conservée entre les fichiers à traduire et les fichiers à aligner. En effet les paramètres réglés ci-dessus le sont pour tous les fichiers, qu'ils soient à aligner ou à traduire. Cela garantit un traitement cohérent, en particulier pour l'interprétation des formats et la segmentation en unités de traduction.

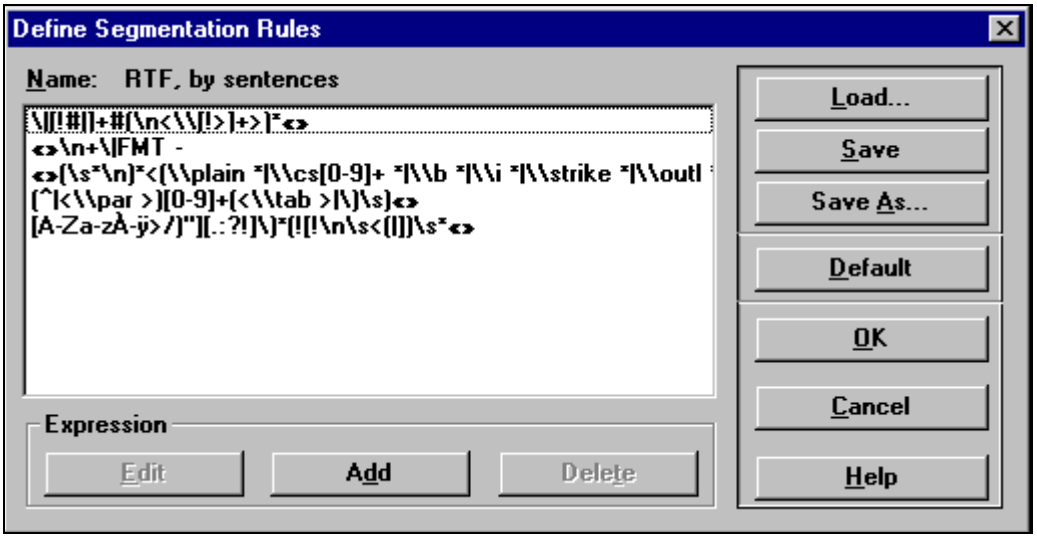


Figure 25 : Définition d'une règle de segmentation à l'aide d'une "expression régulière"

On pourra se reporter à la Figure 29 pour une explication de cette notation.

3.2.2 Ouverture d'une "paire de langues" et édition

Une fois les paramètres fixés par la création d'un projet, il est possible d'éditer le texte à traduire dans un environnement adapté à la traduction. Dans la terminologie de Transit, cette opération s'appelle l'ouverture d'une "paire de langues**{**xe "paire de langues (Transit)"**}**". Elle consiste à ouvrir (importer en fait) un fichier dans un contexte de traduction donné. L'ouverture d'un fichier à traduire provoque l'apparition de quatre fenêtres d'aide à la traduction affichant :

- le texte source

- le texte cible (un fichier cible est créé à l'ouverture, d'où le nom de "paire de langues")
- les entrées du dictionnaires
- les notes (équivalent d'un calepin de notes pour la phase de traduction)

Une cinquième fenêtre d'affichage des correspondances avec la mémoire de traduction peut être ajoutée :

- le réseau associatif (voir plus loin)

L'ordre et le choix de l'affichage des fenêtres peuvent être personnalisés. Les fenêtres source et cible peuvent être synchronisées. Le texte peut être affiché avec les balises qui représentent le code du format relatif au type de fichier (RTF{xe "RTF"}, Interleaf{xe "Interleaf"},..) ou sans ces balises. Il existe deux types de balises{xe "balises"} : les balises de format, et les balises d'attributs de texte. Le texte est présenté par segment de traduction. Un segment peut être soit une balise, soit un segment de texte à traduire (cf. Figure 26).

La traduction du texte s'effectue donc successivement de segment en segment{xe "segment"} de texte, dans la fenêtre cible. Il est possible de naviguer de segments en segments via les différents items du menu "Cursor", par raccourcis claviers, ou avec la souris. Un système de synchronisation des segments sources et cible permet une édition aisée. Les codes de formatage sont éditables ou peuvent être protégés, mais sont indiqués de façon claire et différente du texte. Différents détails d'affichage dans chacune des fenêtres sont réglables par les fonctions du menu "View".

Il est dommage qu'il ne soit pas possible de distinguer a priori le texte concernant les informations relatives au fichier du corps du texte lui même. Ainsi, lors de l'édition d'un fichier RTF, l'auteur du fichier, qui est une information cachée sous Word, est pourtant éditable sous Transit. Il existe un moyen de "masquer" les segments à ne pas traduire, mais cela demande un effort supplémentaire.

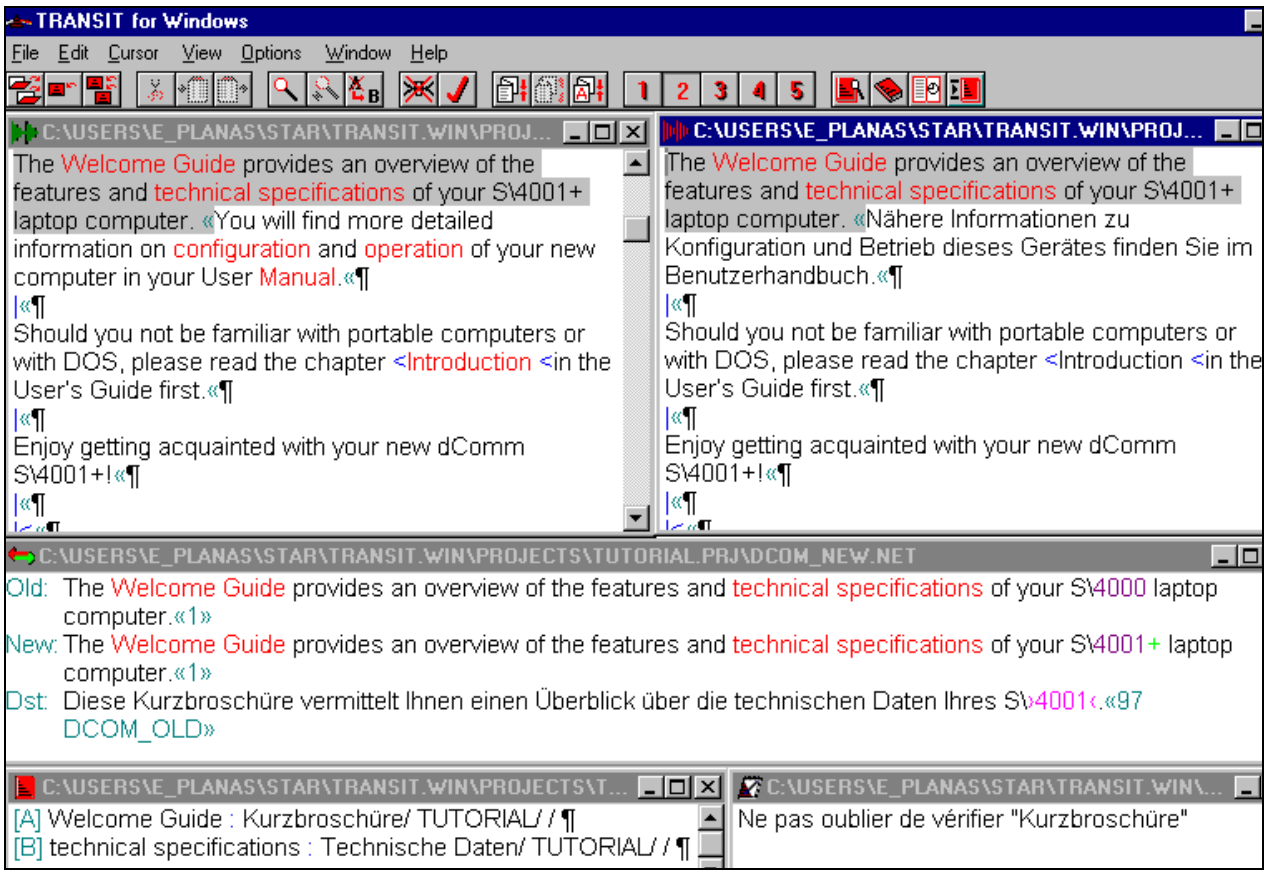


Figure 26 : Edition de la traduction

3.2.3 Pré-traduction

La traduction peut bénéficier de traductions précédentes. Ces traductions peuvent soit provenir des traductions interactives précédentes, soit être constituées à partir d'un alignement de fichiers correspondants sources et cibles, qu'il faut alors préciser à la création du projet comme "fichiers de référence". Les traductions du segment peuvent alors être insérées après l'analyse du fichier dans une phase de pré-traduction**{**xe "pré-traduction (Transit)"**}**. Le choix de l'application ou de la pré-traduction se fait lors de la configuration du projet (cf. Figure 24).

3.2.4 Fonctionnalités de gestion de projet

Différentes fonctionnalités de gestion du projet sont disponibles. En voici une liste :

- Vérificateur orthographique
- Vérificateur de la cohérence terminologique (avec des dictionnaires TermStar)
- Décompte des mots et des caractères
- Vérification des codes de format dans les balises éditées
- Décompte et vérification des segments
- Création de Terminologie nouvelle
- Création de Phraséologie nouvelle
- Alignement de textes sources et cibles

Certaines de ces fonctionnalités sont reprises dans la suite.

3.3 Alignement, Mémoires de Traduction et pré-traduction sous Transit

3.3.1 Alignement de fichiers sources et cibles existants et mémoire de traduction

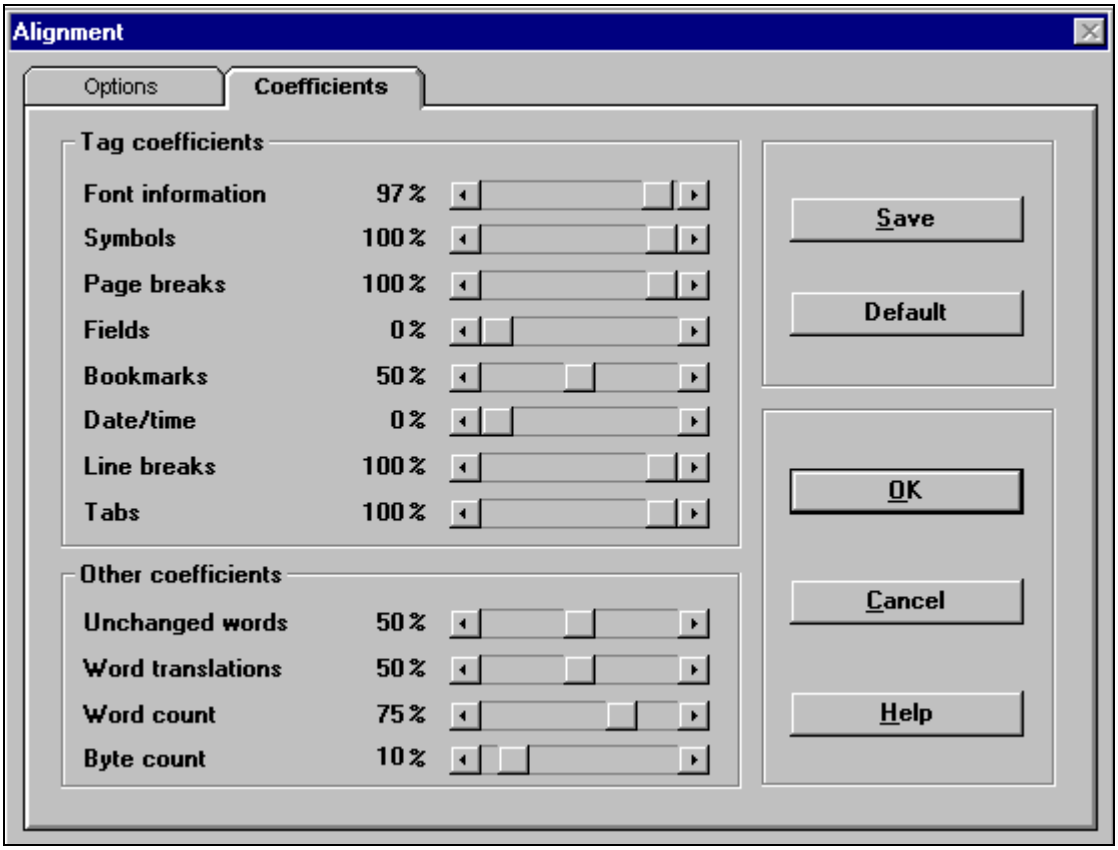


Figure 27 : Réglage des paramètres d'alignement

Transit permet d'aligner deux fichiers traduction l'un de l'autre pour constituer une mémoire (virtuelle) pour les traductions suivantes. Ceci se fait avec la fonction "Options/Alignement". Pour pouvoir aligner{xe "alignement (Transit)"} deux fichiers (tous deux au même format - l'un des formats acceptés par Transit), il faut auparavant les importer pour les transformer au format Transit, et renommer les fichiers ainsi créés avant de les importer à nouveau. On voit que la procédure n'est pas des plus simples !

L'alignement consiste en la recherche de segments équivalents entre les deux fichiers. Cette recherche se fait selon différents paramètres. La Figure 27 montre les paramètres réglables.

En fait il n'existe pas de mémoire au sens où un fichier unique contient la paire de segments source et cible correspondants. La mémoire existe plutôt comme correspondance de numéros de segments. Lors de l'alignement en effet, les segments sont comparés, et un numéro d'occurrence leur est attribué, dans le fichiers source et dans le fichier cible. Les segments ayant le même numéro sont alors traduction l'un de l'autre. Une mémoire sous Transit est donc constituée par un fichier source, son fichier cible, et les numéros correspondants (dans le format "Transit").

3.3.2 Réseau Associatif

Le "Réseau Associatif{xe "Réseau Associatif"}" (Associative Network) est la fonction de recherche de similarité entre le segment nouveau à traduire et les segments du fichier source de référence. Les résultats sont alors présentés dans une fenêtre spéciale (cf. Figure 26) lors de l'édition interactive d'une traduction. Pour utiliser cette fonction, il faut avoir spécifié auparavant au moins une paire de fichiers{xe "fichiers de référence"} source et cible, précédemment alignés, qui tiennent lieu de mémoire de traduction ; ces fichiers mémoires sont appelés "références" dans le jargon de Transit. La Figure 28 montre le réglage du seuil accepté pour les propositions de traduction.

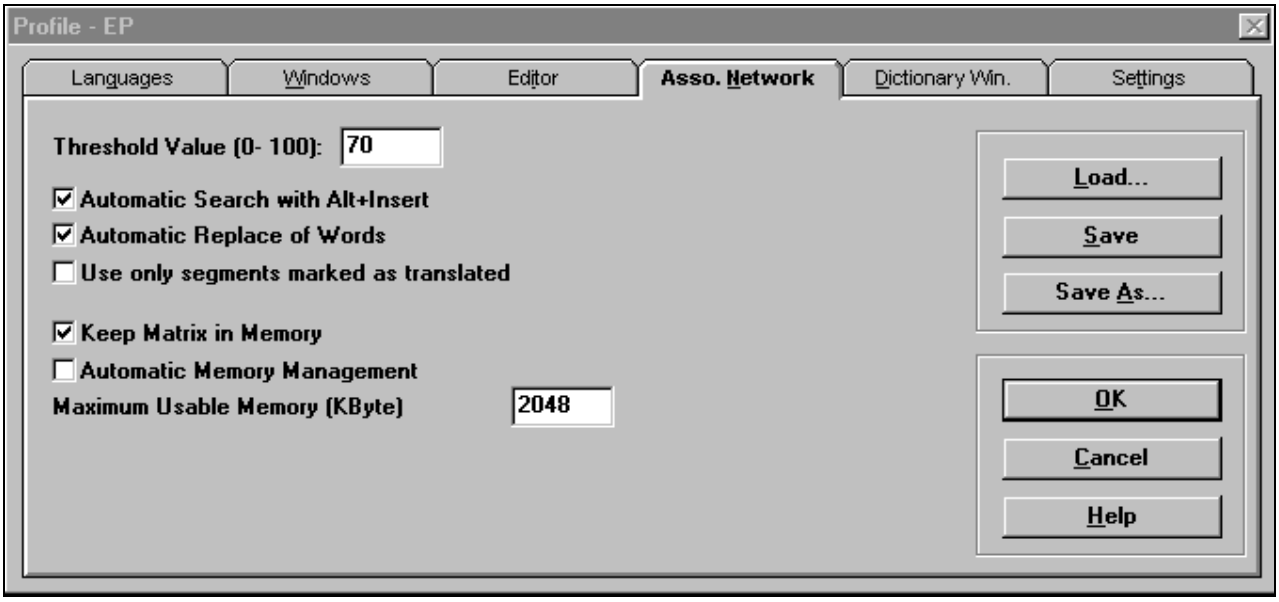


Figure 28 : Réglage des paramètres réseau associatif

Lors de l'édition d'un segment, l'affichage montre l'ancien segment{xe "segment"} similaire provenant de la mémoire, le nouveau à traduire, et la proposition du réseau associatif. Cette proposition peut être copiée par raccourci clavier pour remplacer le segment à traduire. Les différences entre les segments ancien et nouveau sont mises en relief par couleurs.

Dans la recherche de similarité entre les deux segments, une lemmatisation basique permet de reconnaître les pluriels et autres flexions{xe "flexions"}. La génération n'est malheureusement pas capable de donner la bonne forme. On notera cependant qu'il est possible de demander le remplacement automatique des termes trouvés dans le dictionnaire. Ceci se fait par simple substitution du mot source par le mot cible ; le mot cible est donc souvent mal placé (comme dans le cas de la substitution d'un adjectif de l'anglais au français).

3.3.3 Pré-traduction

La fonction de "pré-traduction{xe "pré-traduction"}" se base sur les mêmes principes que l'utilisation du réseau associatif en mode intéractif. A cette différence près qu'elle s'effectue automatiquement lors de l'import d'un fichier, par rapport aux fichiers références formant la mémoire de traduction. Les réglages des paramètres du réseau associatif s'appliquent à la pré-traduction. Seuls les segments exacts sont substitués.

Il est cependant possible de définir des éléments du document dont il ne faudra pas tenir compte lors de la comparaison du segment à traduire avec les segments du fichier source de référence. On utilise alors la notion "d'expression régulière" pour définir la syntaxe des éléments du document à ignorer. Il est ainsi possible d'ignorer le code spécifiant les polices, les index, etc. Mais cela demande un effort supplémentaire non négligeable de définition de ces expressions. La Figure 29 montre la définition d'une exception de pré-traduction pour les polices des documents RTF.

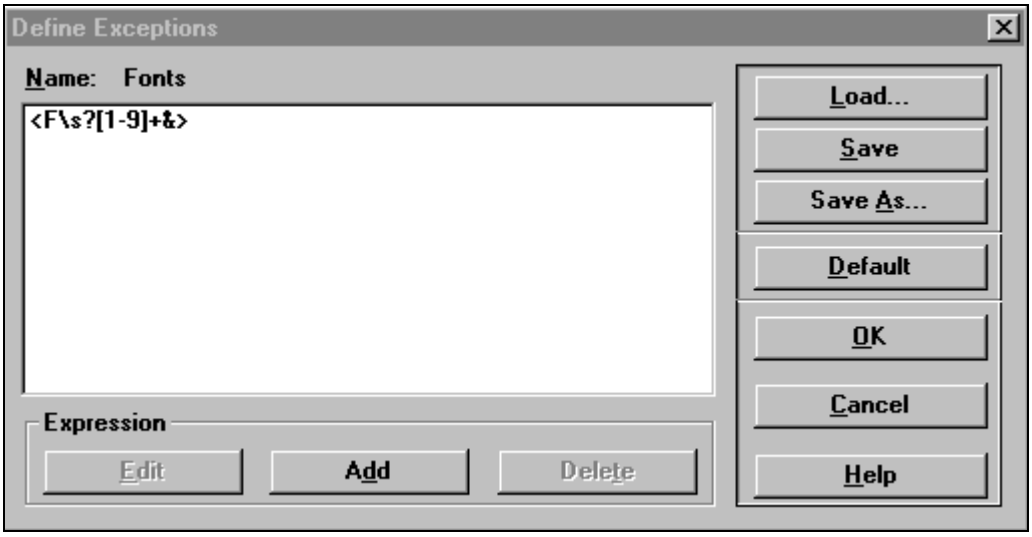


Figure 29 : Définition d'un exception de pré-traduction à l'aide d'une expression régulière

Comme seules les correspondances exactes sont acceptées, la pré-traduction ne fournit en général qu'un faible pourcentage de phrases pré-traduites (cf. le test suivant).

Cette règle permet d'exprimer que la chaîne correspondant au codage d'une police dans un fichier RTF, commence par "<F", puis un espace, puis éventuellement ("?") une nombre, c'est à dire un ou plusieurs ("+") caractères entre 1 et 9, et enfin un ">".

3.4 Structure de la représentation interne

3.4.1 Introduction

Nous nous y sommes référés précédemment sous la dénomination "format Transit". La structure de représentation interne**{**xe "structure de représentation interne (Transit)"**}** garde la séquence des codes d'origine, mais les transforme en "balises". La structure de ces balises suit cependant la représentation du fichier brut d'origine (comme il est vu lors de l'ouverture avec un éditeur de base comme WordPad sous Windows par exemple). Cela permet donc de fournir un langage de niveau supérieur au code des fichiers à traiter grâce à l'interprétation que représentent ces balises, mais en même temps, l'interprétation reste proche de la représentation d'origine du fichier (cf. Figure 26). Cela permet à un opérateur connaissant bien les structures des fichiers d'origine de les manipuler en connaissance de cause.

3.4.2 Procédure d'interprétation et format Transit

Lors de l'import d'un fichier, le filtre parcourt le fichier au format d'origine, et transforme le codage en balises et texte. Les balises**{**xe "balises"**}** représentent les formatages (gras, italique, langue) et objets non textuels (marques index, marques de note de bas de page,..). Les balises sont numérotées. Le fichier Transit ainsi créé prend l'extension "NomOriginal.LAN", où "LAN" représente la langue source (ENG pour anglais par exemple). La Figure 30 montre un extrait de fichier de type Transit.

```
|FMT - Text#
<¥plain ¥s4¥sb80¥sa40¥sl-340¥slmult0¥widctlpar ¥b¥f18¥fs30¥lang1036
>_54_Audience_55_
|FMT - Text#
<¥plain ¥widctlpar ¥f4¥lang1036 >_56_This course_57_
|FMT - Text#_58_
 is intended for <{><¥i >Site Managers<}> (or <{><¥i >Webmasters<}>)
who want to learn about the CCC products. _59_In addition, it is intended
```

```
for <{><¥i >System Engineers<}> who want to prepare for the forthcoming
DDD exam._60_
```

Figure 30 : Représentation interne des données par un fichier de type Transit

Parallèlement, un fichier ".COD" est créé, et contient le code réel du fichier d'origine correspondant à la balise d'interprétation. La Figure 31 ci-dessous donne une idée de la correspondance entre les balises d'un fichier Transit, et le code réel d'un fichier RTF :

```
¥par ¥pard|FMT - Text#
¥par ¥pard|FMT - Text#{¥pard¥plain ¥widctlpar ¥v¥f4¥lang1036 {¥xe
{course}}}|FMT - Text#
¥par ¥pard|FMT - Text#
¥par ¥pard|FMT - Text#{¥pard¥plain ¥widctlpar ¥v¥f4¥lang1036 {¥xe
{course}}}|FMT - Text#
```

Figure 31 : Fichier complémentaire de correspondance entre le fichier Transit et le code source dont est issu le fichier Transit

Un fichier pour la langue cible, similaire au fichier ".LAN", donc au format Transit est créé. C'est ce fichier qui est modifié lors de la pré-traduction et de l'édition, et qui devra être exporté pour donner le fichier traduit au format original.

3.5 La terminologie : TermStar

TermStar**{**xe "TermStar"**}** est un gestionnaire de terminologie**{**xe "terminologie (Transit)"**}** classique. Cette application permet de créer et gérer des dictionnaires**{**xe "dictionnaires"**}**, et de rechercher les termes selon différents critères.

Des fonctions d'import et d'export permettent de créer des extraits du dictionnaire sous table ASCII ou ANSI. L'application ouverte se présente comme dans la Figure 32. Des interfaces permettent d'utiliser TermStar sous la version pour Windows**{**xe "Windows"**}** de Word**{**xe "Word"**}**, WordPerfect**{**xe "WordPerfect"**}** et Amipro**{**xe "Amipro"**}**. TermStar est aussi le logiciel appelé par Transit pour gérer la terminologie lors de la traduction sous Transit. Les bases de données (les dictionnaires) de TermStar que l'on veut utiliser lors de la traduction sous Transit, d'extension ".tbd" sont spécifiables lors de la définition d'un projet (cf. Figure 23), et sont alors ouvertes sous TermStar pour Transit lors de l'appel d'un projet.

Les dictionnaires sont incrémentables depuis la fenêtre d'édition d'un fichier sous Transit. De même il existe une fonction pour créer toutes les entrées qui n'existent pas dans les dictionnaires de départ de la traduction.

Sous la fenêtre d'édition de Transit, les termes sources reconnus dans les dictionnaires sont indiqués d'une couleur définissable par l'utilisateur (cf. Figure 26). Par raccourci clavier, il est possible d'introduire à la position du curseur le terme correspondant.

Une fonction d'acceptation systématique du premier terme proposé pour les entrées trouvées dans le dictionnaire est disponible. Dans ce cas, dans la fenêtre cible, les termes sont insérés automatiquement à la place du terme source, selon bien sûr la syntaxe de la langue source (ce qui peut être gênant non seulement pour un couple de langues de structures opposées (ex : français et japonais), mais aussi dans le cas ou les structures sont les mêmes car il faut alors "sauter" de gauche à droite du mot, et l'écriture de la traduction n'est plus simple.

Une autre fonction permet de vérifier que la terminologie employée dans la traduction est cohérente avec celle des dictionnaires du projet.

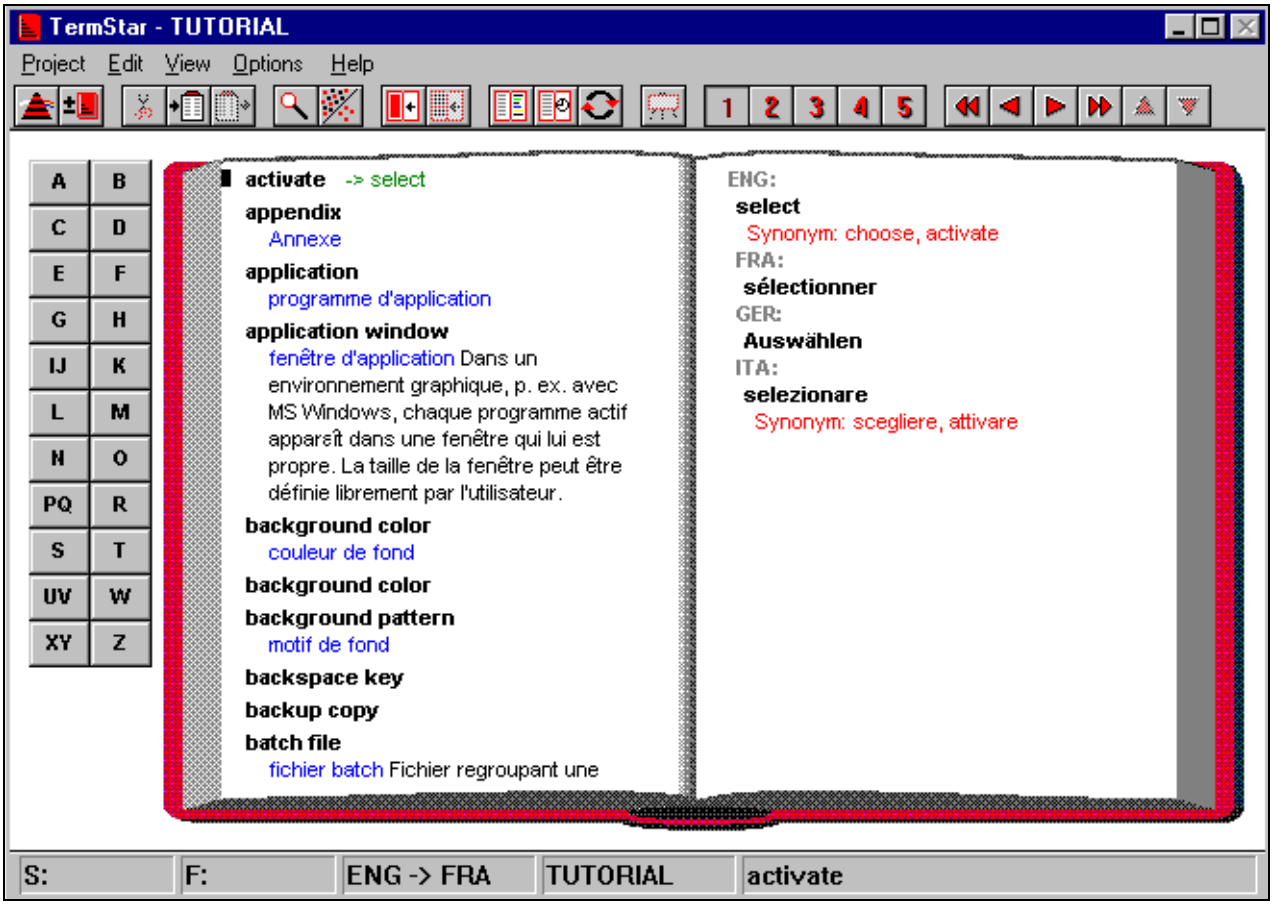


Figure 32 : Fenêtre principale de TermStar

3.6 Coopération avec Logos

Il est possible d'ajouter à Transit une interface de communication avec le logiciel de traduction par règles Logos{xe "Logos"}. Dans ce cas, au niveau de la pré-traduction des fichiers, l'interface envoie à Logos l'ensemble des segments qui n'ont pas pu être traduits par consultation du réseau associatif. Il est possible de choisir d'intégrer automatiquement ces phrases au niveau de la pré-traduction, ou simplement de les proposer en mode interactif.

3.7 Originalités

3.7.1 Correcteur d'orthographe intégré.

Le correcteur orthographique intégré est un module différent du gestionnaire de terminologie TermStar. Il est possible d'ajouter ces modules au fur et à mesure des besoins moyennant finance (seuls les lexiques anglais et allemands sont fournis par défaut). Le passage de ce module sur le texte traduit permet une correction orthographique de base. Un dictionnaire supplémentaire aux dictionnaires de base est incrémentable par l'utilisateur.

3.7.2 Remplacement terminologique dans un segment approché de la mémoire

Lorsque des segments similaires au segment à traduire sont retrouvés par le "réseau associatif", ceux-ci diffèrent parfois de certains mots. Si ces mots se trouvent dans les bases TermStar{xe "terminologie"}, il est alors possible de les réintégrer automatiquement dans la proposition de traduction qu'offre le "réseau associatif".

3.7.3 Vérification de la cohérence terminologique

Cette vérification consiste à rechercher dans les bases terminologiques TermStar si l'équivalent cible de chaque mot source trouvé dans ces bases existe bien dans le fichier cible qui est en train d'être traduit. Des messages aident à retrouver les équivalents cibles n'ayant pas été trouvés dans le texte cible. Transit est capable de retrouver les formes des mots de base (déclinaisons, pluriels, conjugaisons).

3.7.4 Macro Commandes

Un ensemble d'opérations de base de Transit peut être enregistré et mémorisé dans une "macro**{**xe "macros (Transit)"**}**". Un raccourci clavier ayant été attribué à cette macro, la pression des touches correspondantes au raccourci permet d'enchaîner les tâches enregistrées, sans devoir repasser par l'ensemble des commandes. Il s'agit de la même idée que celle que l'on peut trouver dans Excel ou Nisus, appliquée à Transit.

Conclusion

Ce chapitre a montré que les outils de traduction fondés sur la mémoire de première génération reposent sur les mêmes principes, et proposent des fonctionnalités similaires. Ils se distinguent toutefois par quelques points faisant l'originalité de chacun. La représentation des données en interne consiste toujours en une juxtaposition des données en un flot linéaire.

Chapitre 3

Faiblesses et espoirs des outils de première génération

Contenu du chapitre

Introduction

1. Test comparatif sur un fichier .RTF

1.1 Introduction

Le principe de ce test{xe "test"} consiste à appliquer une mémoire phraséologique construite à partir de fichiers RTF, sur des phrases tests, et cela pour les trois logiciels vus aux chapitres précédents.

Pour cela, on aligne deux fichiers, la source et sa traduction (cible), pour obtenir une mémoire phraséologique. Les moyens d'alignement peuvent être divers, et éventuellement manuels, ce n'est pas cela qui est testé. Le but est simplement d'avoir exactement les mêmes phrases sources et cibles dans chacune des mémoires (une par logiciel), avec les mêmes attributs de format. Les éléments que nous avons particulièrement testés sont :

- l'exactitude de la traduction proposée par le logiciel
- la traduction des formes fléchies, comme les pluriels et les conjugaisons
- la capacité à remplacer un mot de la phrase trouvée dans la mémoire par un mot du dictionnaire
- la capacité à transférer le formatage des phrases, des mots, et des lettres, depuis la phrase source sur la phrase cible
- la gestion des objets imbriqués (marques d'index et de note de bas de page)
- la qualité du formatage de restitution du document final

1.2 Fichiers parallèles de départ

Voici les fichiers tests de départ. Ils sont dérivés (banalisés) d'un fichier anglais et de sa traduction d'un vrai cours en ligne portant sur un produit d'édition. Ces fichiers sont enregistrés au format RTF.

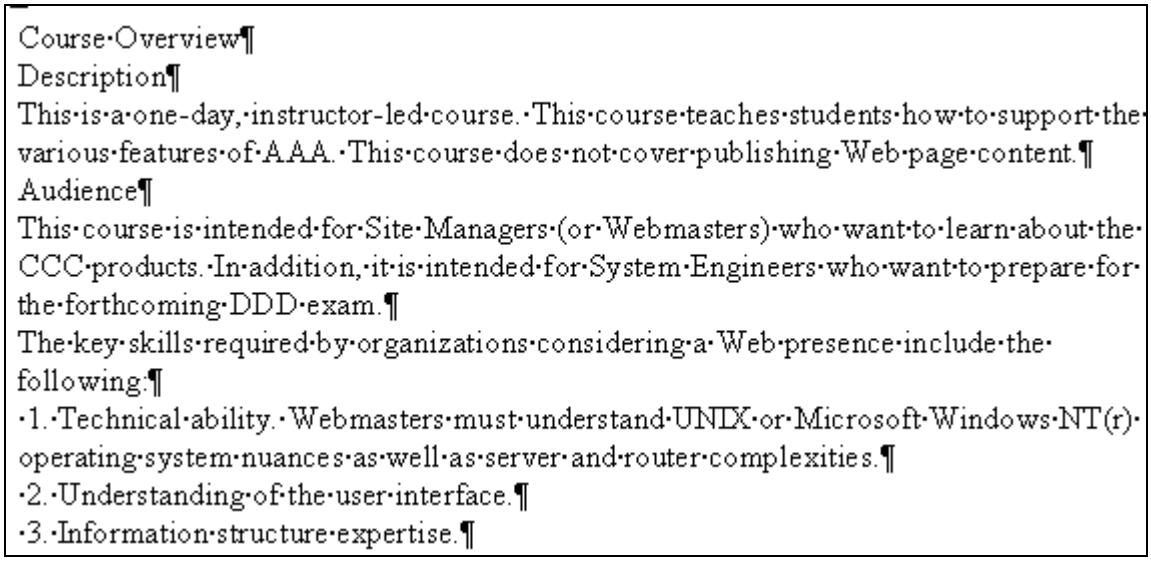

Course·Overview¶
Description¶
This·is·a·one-day,·instructor-led·course.·This·course·teaches·students·how·to·support·the·various·features·of·AAA.·This·course·does·not·cover·publishing·Web·page·content.¶
Audience¶
This·course·is·intended·for·Site·Managers·(or·Webmasters)·who·want·to·learn·about·the·CCC·products.·In·addition,·it·is·intended·for·System·Engineers·who·want·to·prepare·for·the·forthcoming·DDD·exam.¶
The·key·skills·required·by·organizations·considering·a·Web·presence·include·the·following:¶
·1.·Technical·ability.·Webmasters·must·understand·UNIX·or·Microsoft·Windows·NT(r)·operating·system·nuances·as·well·as·server·and·router·complexities.¶
·2.·Understanding·of·the·user·interface.¶
·3.·Information·structure·expertise.¶

Figure 1 : fichier RTF de départ (anglais)

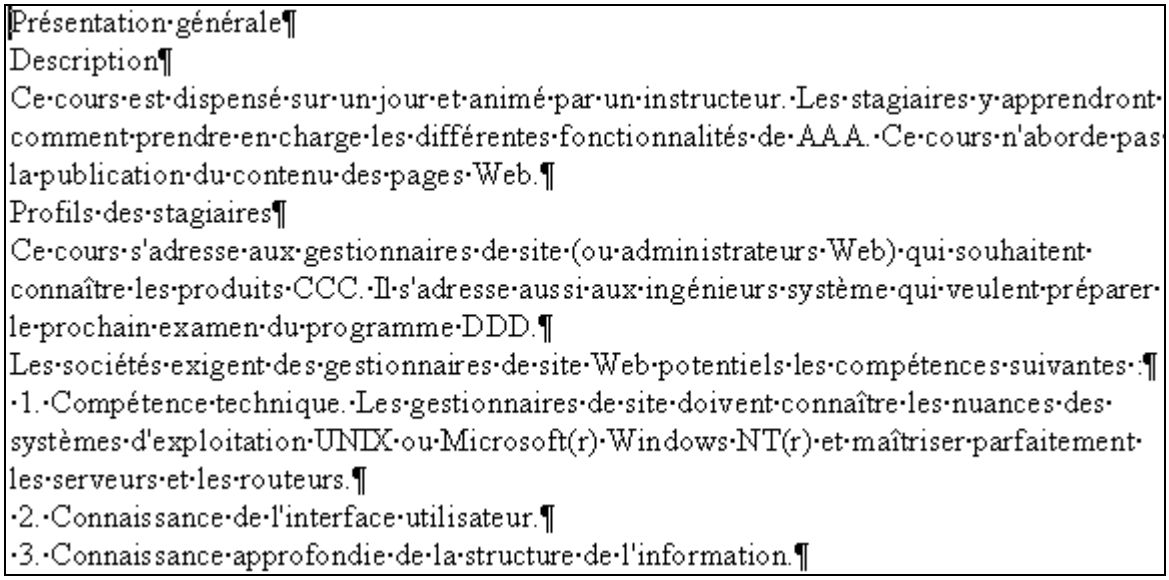

Présentation·générale¶
Description¶
Ce·cours·est·dispensé·sur·un·jour·et·animé·par·un·instructeur.·Les·stagiaires·y·apprendront·comment·prendre·en·charge·les·différentes·fonctionnalités·de·AAA.·Ce·cours·n'aborde·pas·la·publication·du·contenu·des·pages·Web.¶
Profils·des·stagiaires¶
Ce·cours·s'adresse·aux·gestionnaires·de·site·(ou·administrateurs·Web)·qui·souhaitent·connaître·les·produits·CCC.·Il·s'adresse·aussi·aux·ingénieurs·système·qui·veulent·préparer·le·prochain·examen·du·programme·DDD.¶
Les·sociétés·exigent·des·gestionnaires·de·site·Web·potentiels·les·compétences·suivantes·:¶
·1.·Compétence·technique.·Les·gestionnaires·de·site·doivent·connaître·les·nuances·des·systèmes·d'exploitation·UNIX·ou·Microsoft(r)·Windows·NT(r)·et·maîtriser·parfaitement·les·serveurs·et·les·routeurs.¶
·2.·Connaissance·de·l'interface·utilisateur.¶
·3.·Connaissance·approfondie·de·la·structure·de·l'information.¶

Figure 2 : fichier RTF de départ (français)

Aucun formatage de texte, aucun style particulier n'a été retenu, sinon les indentations du bas du texte.

1.3 Création de mémoires strictement identiques

1.3.1 Workbench

La mémoire**{**xe "mémoire"**}** a été créée manuellement, bien qu'elle puisse l'être automatiquement par le logiciel TAlign ou WinAlign**{**xe "TAlign"**}**, car ce dernier n'était pas disponible au moment des tests. La procédure a consisté à créer un fichier texte ayant une structure importable dans la mémoire vide créée sous Workbench. Le contenu de la mémoire

peut être visualisé sur le fichier correspondant à l'export de la mémoire, et disponible en Annexe 1.1.

1.3.2 Translation Manager

Le procédé est similaire à celui appliqué pour Workbench. La mémoire{xe "Translation Manager"} exportée est visible en Annexe 1.2.

1.3.3 Transit

La mémoire est créée selon la procédure d'alignement décrite plus haut (cf. C.3.3.1). Il a fallu une intervention manuelle importante. Il a bien été vérifié avant la pré-traduction, que chaque segment se corresponde bien. En effet, il n'existe pas de fichier représentant la mémoire à proprement parler sous Transit{xe "Transit"}. C'est la correspondance des numéros des segments du fichier source et du fichier cible qui représente la mémoire. Le fichier mémoire ne peut donc pas être montré ici. On peut cependant comparer les fichiers extraits des deux documents de référence (source et cible) qui forment la mémoire par la correspondance de leurs segments (cf. Annexe 1.3).

1.4 Les fichiers test

Ces fichiers reprennent les mêmes phrases que celles entrées dans la mémoire, avec cependant quelques différences au niveau linguistique, du formatage et des objets imbriqués. Le fichier{xe "fichier test"} à traduire est montré dans la Figure 3. La version française telle qu'elle devrait être traduite se trouve dans la Figure 4.

Course·Overview¶

Descriptions¶

This·is·a·one-day,·*instructor*-led·course{xe·"course"}.·This·course·teaches·*students*·how·to·support·the·various·particularities·of·AAA.·This·course·does·not·cover·publishing·**Web**·page·content.¶

Audience¶

This·course{xe·"course"}·is·intended·for·*Site·Managers*·(or·*Webmasters*)·who·want·to·learn·about·the·CCC·products.·In·addition,·it·is·intended·for·*System·Engineers*·who·want·to·prepare·for·the·forthcoming·DDD·exam.¶

The·key·skills·required·by·organizations·considering·a·Web·presence·include·the·following:¶

1. Technical·abilities.·**Webmasters**·must·understand·UNIX·or·Microsoft·Windows°NT·operating·system·nuances·as·well·as·server·and·router·complexities.¶
2. Understanding·of·the·happy·user·interface.¶
3. *Information·structure*{xe·*expertise*.¶

Figure 3 : Version anglaise du fichier test

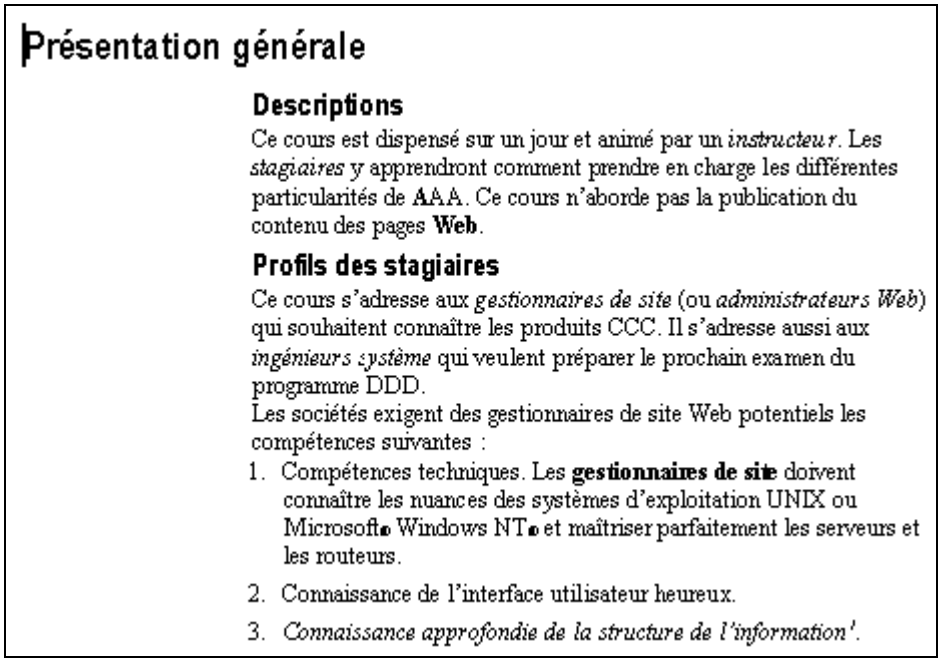


Figure 4 : Version française du fichier test

Voici les différences entre les fichiers de départ servant de mémoire et les fichiers à traduire des figures précédentes, classées par type :

- Linguistique
  a) le pluriel sur “descriptions”
  b) “features” est remplacé par “particularities”, dans la deuxième phrase
  c) le pluriel est appliqué au groupe “Technical abilities”.
  d) l’adjectif “happy” est ajouté dans l’avant-dernière phrase

- Format
  e) le style “heading 2” est appliqué à “Course Overview”
  f) le style “heading 4” est appliqué à “Description” et “Audience”
  g) le formatage “gras” est appliqué aux mots suivants : Web, Webmaster, et à la première lettre “A” de AAA.
  h) le style “italique” est appliqué aux mots : Site Managers, Webmasters, System Engineers
  i) les styles “gras” et “italique” sont appliqués sur : instructor, students
  j) la dernière phrase est totalement en italiques

- Objets imbriqués
  k) une balise d’index ({XE: “course”}) est insérée après “course” dans les premières phrases des deux premiers chapitres
  l) une marque de note pied de page est insérée après “Information structure” dans la dernière phrase.

1.5 Terminologie

Les termes{xe "terminologie"} qui ont été changés et les nouveaux termes sont incorporés dans les dictionnaires relatifs, sous leur forme de base (lemme). Les informations grammaticales qui sont entrées sont la catégorie grammaticale (POS : “part of speech”), le genre, et le nombre. Ces termes sont :

- description (noun, s), description (noun, f, s)

- feature (noun, s), fonctionnalité (noun, f, s)
- particularity (noun, s), particularité (noun, f, s)
- technical (adj, ∅), technique (adj, m, s)
- ability (noun, s), compétence (noun, f, s)
- happy (adj, ∅), heureux (adj, m, s)

Ainsi une nouvelle base terminologique est créée pour Workbench sous Multiterm, pour Transit sous TermStar, et un nouveau dictionnaire est créé sous Translation Manager. Voici par exemple son aspect sous TermStar (figure) :

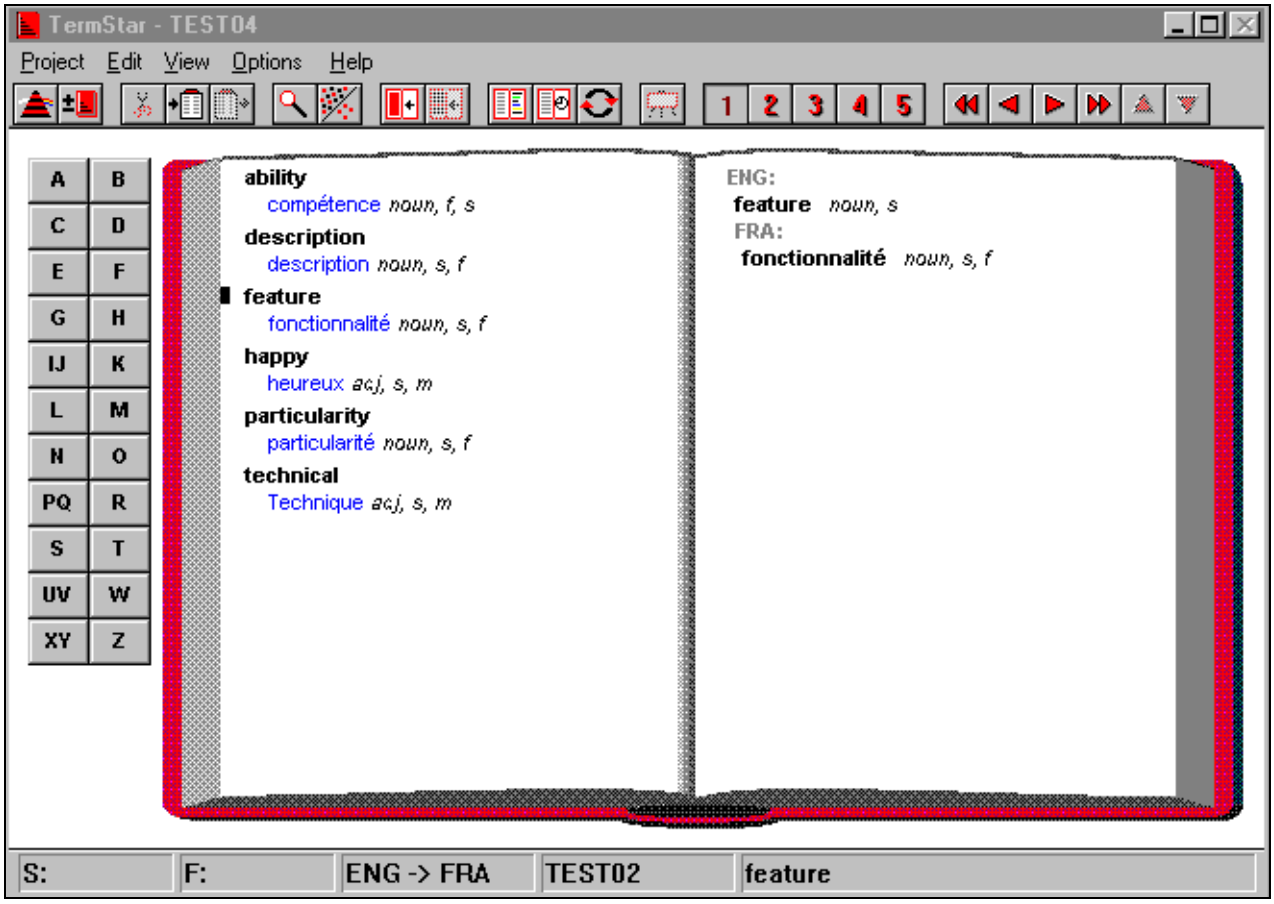


Figure 5 : Dictionnaire du test sous Transit

1.6 Le test de traduction

1.6.1 Workbench

La version 1.14 de Workbench**{**xe "Workbench"**}** a été utilisée pour ce test**{**xe "test"**}**. Les paramètres de traduction sont réglés comme suit :

- pénalités pour différences entre la phrase anglaise de la mémoire et la phrase à traduire ; on notera que les coûts unitaires sont des nombres premiers, cela nous permet, par une décomposition en facteurs premiers, d'analyser exactement le pourcentage résultat :
  ⇒ coût d'une différence de format : 3%
  ⇒ coût d'une différence d'autres attributs : 2%
  ⇒ coût d'une différence sur les objets imbriqués : 5%
  ⇒ Autres coûts : 0%

- seuil de similarité accepté : 70%

Le résultat de la traduction par Workbench est présenté dans la figure suivante :

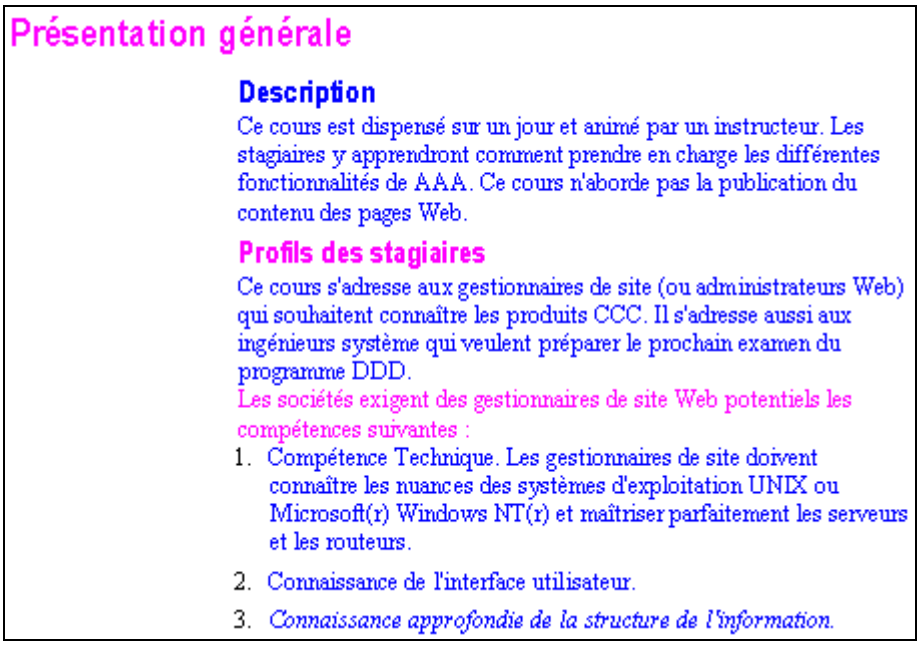


Figure 6 : Traduction automatique du fichier test par Workbench

Les observations principales sont les suivantes :

- La couleur magenta est appliquée aux phrases “traduites à 100%” ; le bleu aux traductions approchées (entre 70% et 99%).
- Les styles et formats appliqués à des phrases entières (en fait à des “paragraphes Word”) sont conservés (e, f, j).
- Les formats appliqués à des lettres, mots ou groupes de mots différents d’une phrase entière, ne sont pas transmis. On notera que, si les styles avaient été introduits en mémoire (pour la source et la cible), ils auraient été transmis, même si leurs attributs changent (ex : gras vers italique), pourvu qu’ils soient appliqués au même endroit.
- Les nouveaux mots introduits n’ont pas été traduits, bien qu’ils soient dans la base terminologique. Ils sont cependant reconnus, et disponibles dans une fenêtre de Workbench, si l’on traduit en mode interactif.
- En mode interactif, les marques d’index, ainsi que la marque note de pied de page sont reconnues comme des objets imbriqués, et leur placement est proposé. Elles ne sont cependant pas placées automatiquement au bon endroit.

1.6.2 Translation Manager

La version{xe "Translation Manager"} 2.0.7, a été utilisée pour ce test. Le résultat de la traduction, affiché dans Word, est montré par la figure suivante :

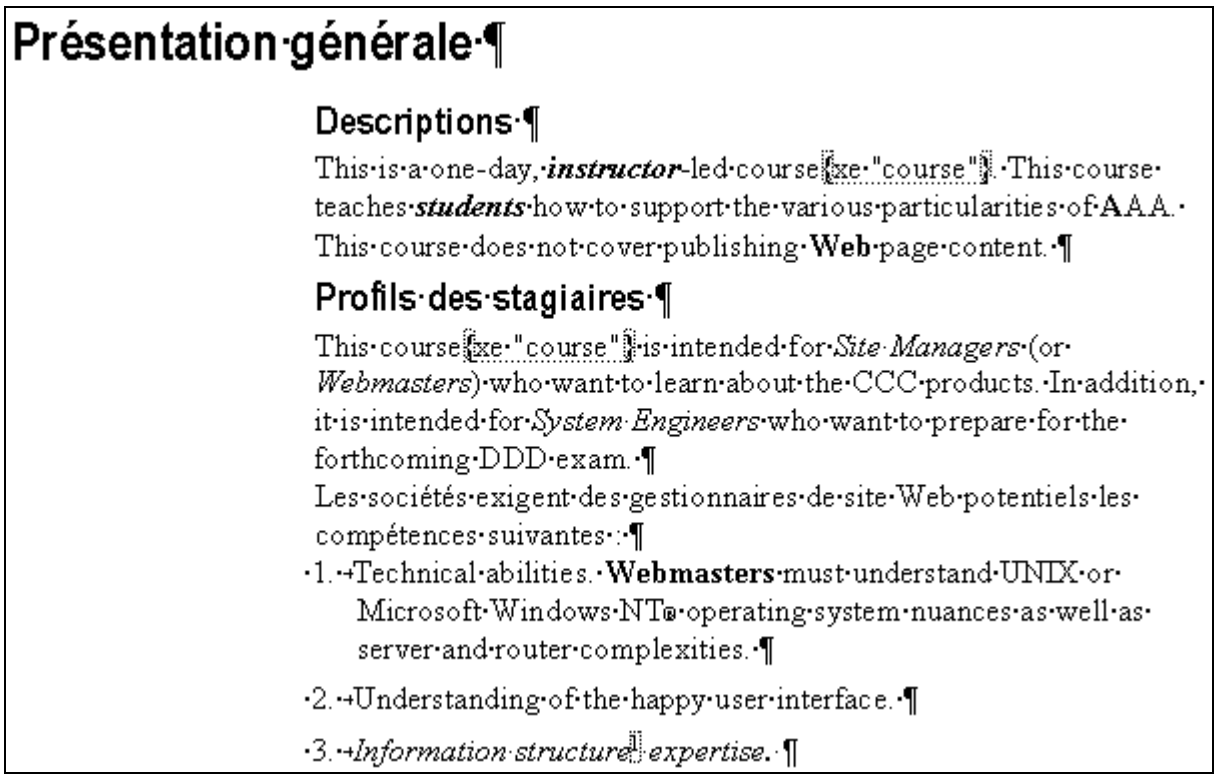


Figure 7 : Résultat de la traduction automatique du fichier test sous Translation Manager

En fait les résultats sont meilleurs si l'on procède à la traduction interactive, comme le montre la fenêtre de Translation Manager illustrée par la Figure 8. Voici les principaux commentaires :

- En mode automatique, les traductions approchées ne sont pas substituées. Il faut pour cela le demander explicitement dans le mode interactif, pour chaque phrase.
- Le pourcentage de correspondance n'est pas fourni par TM.
- En mode interactif, les balises d'index et la marque de note pied de page sont reconnues.
- Souvent une erreur de remplacement du code, dans le fichier cible édité sous Translation Manager, cause une impossibilité d'ouverture du fichier exporté sous Word.

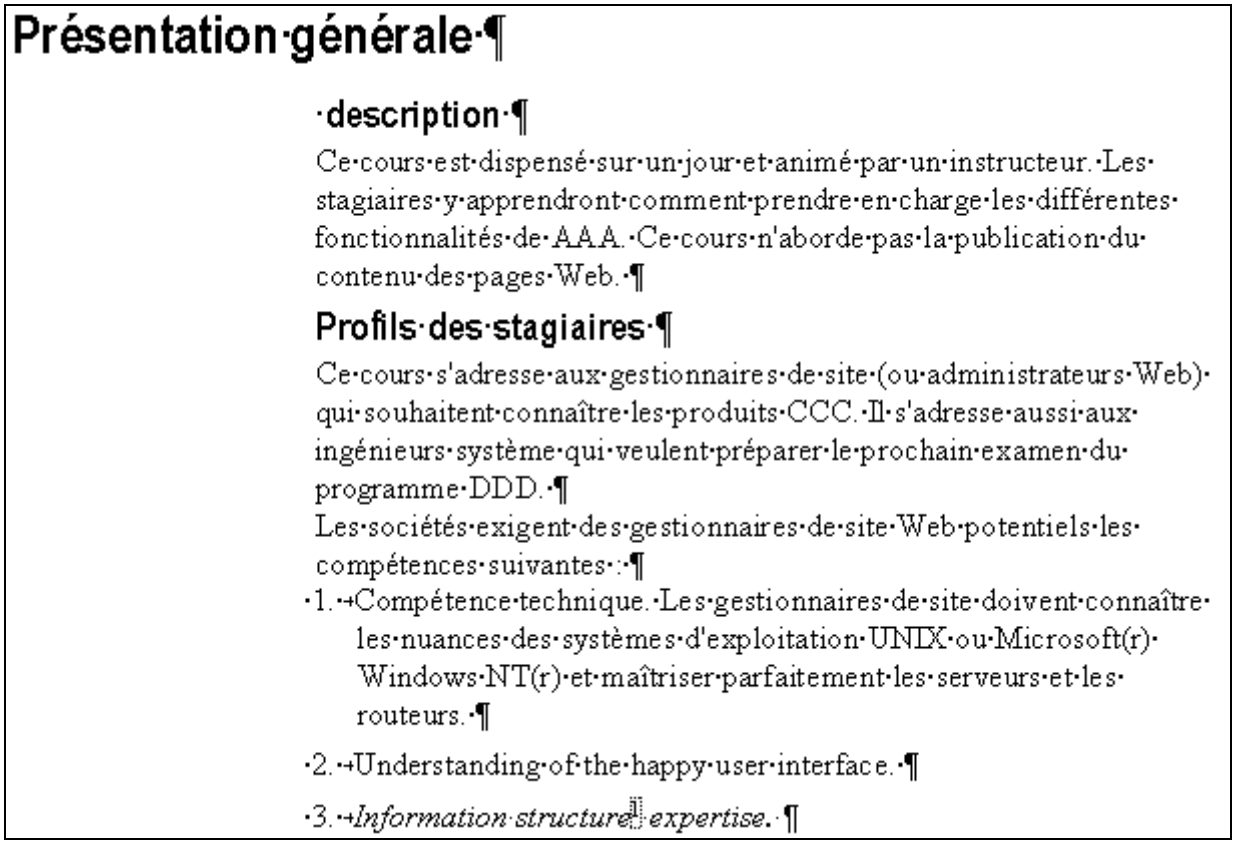


Figure 8 : Traduction du fichier test sous Translation Manager (intéractif)

1.6.3 Transit

La version utilisée est Transit{xe "Transit"} 2.6. La pré-traduction du fichier, c'est-à-dire le mode automatique ne remplace que les phrases identiques, aux "exceptions" près (cf. chapitre 2).

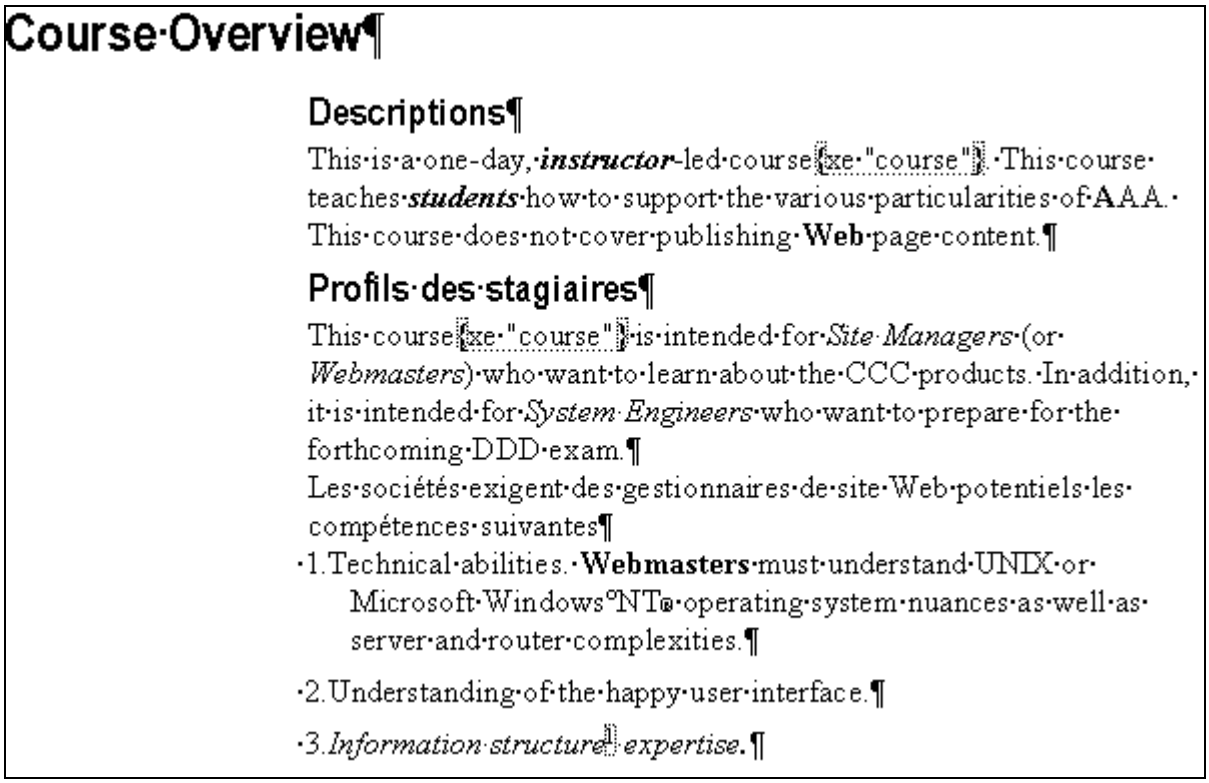


Figure 9 : Résultat de la pré-traduction sous Transit

Cela ne donne qu'un fichier partiellement traduit comme le montre la Figure 9. Transit reconnaît cependant plus que ces segments, et la consultation du réseau associatif (cf. Figure 28 du chapitre 2) lors d'une édition manuelle donne le résultat de la Figure 10 (pour un seuil de similarité de70%). Pour ce test les réglages des paramètres du réseau associatif étaient :

- la balise représentant le code des polices a été considérée comme non influente sur la recherche de similarité, cela s'appelle une "exception" de pré-traduction dans le langage de Transit.
- L'ensemble des paramètres fins de pré-traduction a été réglé comme la figure 4 du chapitre 2 l'indique. Le seuil d'acceptation a été fixé à 70%.

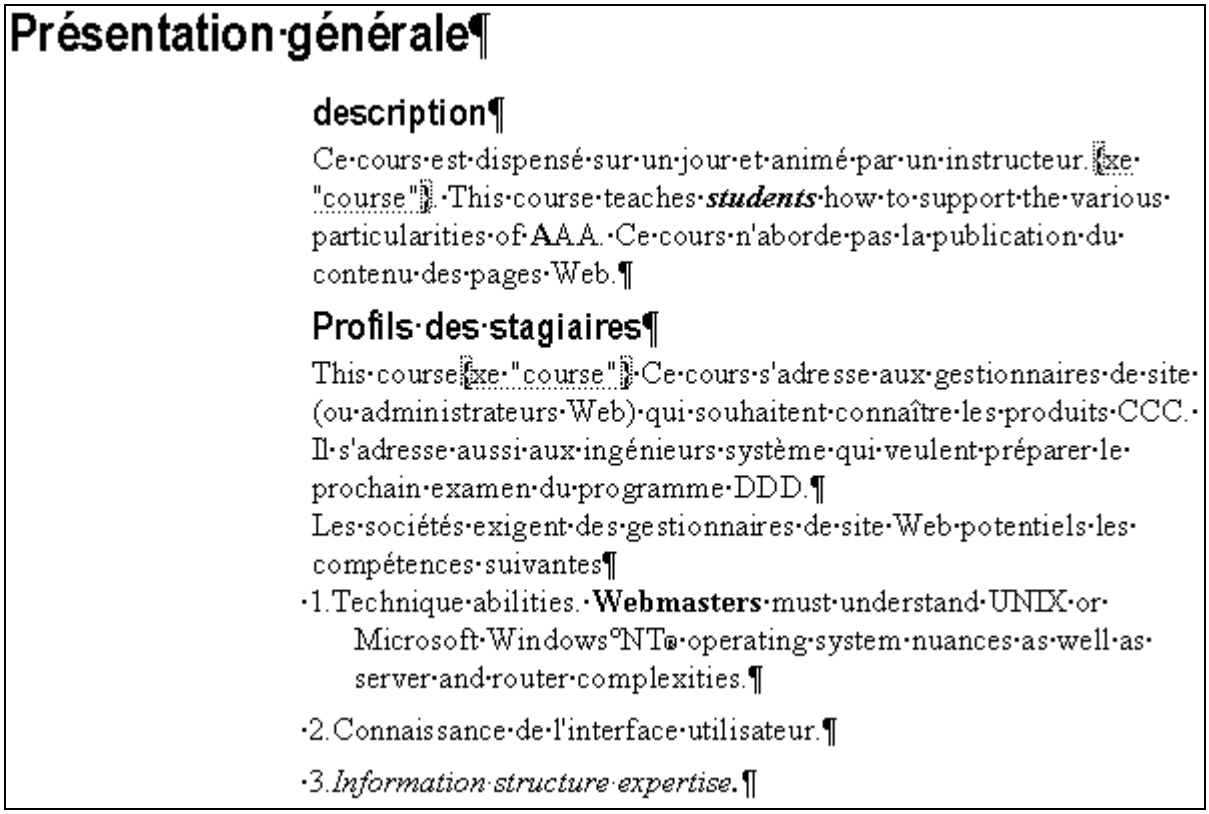


Figure 10 : Résultat de l'acceptation systématique des traductions proposées par le réseau associatif, sous Transit

Le problème de la gestion des objets non textuels est bien illustré ici (par une marque index sur le mot anglais "course"). Outre le problème de la traduction de la référence elle-même à l'intérieur de la marque d'index ("course" devient "cours"), cette marque n'est pas replacée au bon endroit. En fait il y a un effet secondaire supplémentaire : la segmentation**{**xe "segments"**}** du texte a arrêté le segment à l'index, alors que la logique de la phrase aurait voulu qu'elle s'arrêtât au point suivant. Cela crée un problème de recherche de segment équivalent dans la mémoire, qui n'est pas trouvé. La partie de la phrase située avant l'index est donc laissée en anglais. Par la suite, lors de la traduction de la seconde partie de la phrase, et avec un seuil de traduction à 70%, l'ancienne phrase **complète** de la mémoire est proposée. C'est pour cela qu'elle figure en entier. Cette segmentation est visible sur la Figure 11 qui représente le fichier test en phase d'édition sous Transit.

Il faut remarquer ici que les mots "description" et "technique" proviennent d'un remplacement automatique du logiciel. Cependant "heureux" n'a pas été remplacé. Lorsqu'il y a remplacement, il n'y a ni génération de la bonne forme ("Descriptions" et pas "description"), ni placement au bon endroit ("Compétences techniques", et pas "Technique abilities").

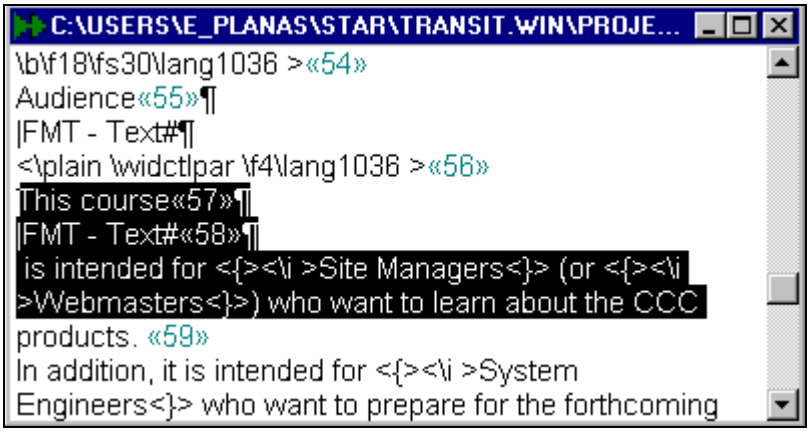


Figure 11 : Problème de segmentation sous Transit, dû à une marque d'index

1.7 Résultats

1.7.1 Score de traductions (même approximatives) trouvées en mode interactif

sent	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*T*
WB	1	1	1	1	1	1	1	1	1	1	1	1	1	13/13
TM	1	1	1	1	1	1	1	1	1	1	1	0	0	11/13
TR	1	1	1	0	1	1	1	1	1	0	0	1	0	9/13

Les résultats ci-dessus montrent quelles phrases on pu être traduites parmi les treize phrases du fichier test, pour un seuil de traduction approchée de 70%, en mode interactif. On notera que si l'on baisse le seuil, on retrouve progressivement toutes les traductions. Il faut prendre en considération que les seuils ont différentes significations selon l'OTAM1g. Le pouvoir potentiel de retrouver une traduction dans la mémoire est donc à peu près équivalent et correct pour tous ces OTAM1g. On remarquera que les phrases à traduire étaient littéralement les mêmes que celles de la mémoire. Les différences proviennent seulement des formats ou des objets non textuels.

1.7.2 Score de traductions (même approximatives) trouvées en mode automatique (pré-traduction)

sent	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*T*
WB	1	1	1	1	1	1	1	1	1	1	1	1	1	13/13
TM	1	0	0	0	0	1	0	0	1	0	0	0	0	3/13
TR	0	0	0	0	0	1	0	0	1	0	0	0	0	2/13

Le remplacement automatique des segment à traduire avec les solutions fournies par la mémoire est beaucoup plus problématique. La différence entre Workbench et les autres OTAM1g provient du fait que seul Workbench autorise une traduction automatique approchée en pré-traduction. Les autres n'autorisent que des traduction "exactes" selon leurs critères.

1.7.3 Test des transfert de format et de traduction augmentée

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*Total*
WB	0	0	0	0	1	1	0	0	0	1	0	0	3/12
TM	0	0	0	n/a	1	1	0	0	0	n/a	0	n/a	2/12
TRA	0	n/a	n/a	0	1	1	0	0	0	n/a	0	n/a	2/12

Dans ce tableau, chaque colonne représente un des points à tester de la liste proposée au paragraphe 1.4. Le score "1" est attribué si la gestion de ce point a été correctement effectuée, 0 sinon ; "n/a" signale que le test n'a pas donné de résultat car la phrase n'a pas été traduite.

De façon globale donc les OTAM1g ne sont pas capables de gérer correctement le transfert du format. Cependant :

- Quand le formatage de mots particuliers est contenu dans la mémoire, il peut être retrouvé pour ces mots.
- La qualité du transfert de format de la source vers la cible, et dans une moindre mesure celle de la traduction, dépend fortement de la capacité du logiciel à appréhender le formatage du document à traduire. Cette capacité change d'un logiciel à l'autre. Translation Manager, par exemple, s'accommode mieux de documents FrameMaker que les autres, et Workbench donne ses meilleures performances pour les documents RTF.

1.8 Conclusions

L'étude précédente couvre trois des principaux OTAM1g du marché occidental de l'année 1998. En ce sens elle est couvrante. Le fichier test utilisé rassemble des points précis qui font à la fois les forces et les faiblesses des OTAM1g et provient d'un document réel adapté ; en ce sens l'étude précédente est aussi représentative des performances de ces outils. Elle ne l'est pas pour les raisons suivantes :

- le test est limité à un fichier de format RTF. D'autres fichiers du même format ou d'autres formats (Interleaf, FrameMaker,..) pourraient soulever de nouveaux problèmes ou montrer quelques avantages des OTAM1g
- toutes les possibilités fines de ces outils n'ont pas été testées, comme par exemple :
  ⇒ l'adaptation de la segmentation du texte
  ⇒ l'adaptation optimale des paramètres de recherche de similarité
  ⇒ la préparation extérieure des fichiers pour une adaptation précise à l'outil testé

Cette étude a aussi permis de faire ressortir les fonctionnalités principales des OTAM1g, qui peuvent être résumées par cette liste :

- Pré-traduction
- Traduction interactive (édition de la traduction sous un environnement adéquat).
- Gestion des mémoires de traduction
- Gestion du dictionnaire
- Module d'alignement
- Extraction de phraséologie redondante au sein du document
- *Extraction de listes de vocabulaire*
- *Pré-analyse*
- *Mise à jour des documents*
- *Connexion à un module de traduction automatique fondée sur les règles*

Les éléments en italique indiquent des fonctionnalités importantes mais non disponibles dans l'ensemble des OTAM1g vus ci-dessus.

Au niveau de l'édition, deux approches différentes ont été vues. Dans la première, le logiciel de TFM utilise son propre éditeur. C'est le cas de Translation Manager et de Transit. Dans ce cas, les fichiers doivent être convertis depuis leur format vers un format interne, puis restitués au format d'origine après leur traduction. Dans la seconde, le logiciel utilise un éditeur externe du marché, et communique par DLL interposées. C'est le cas de Workbench qui utilise Word ou WordPerfect, et d'Eurolang Optimizer qui peut utiliser Word ou FrameMaker. Dans ce cas, le format utilisé est celui de l'éditeur hôte (Word).

Nous n'avons pas traité dans cette étude la comparaison des aptitudes à la gestion de projets de traduction, comme le comportement devant un groupe de fichier à traduire et les fonctionnalités relatives. Dans les cas réels, la gestion de projet est un facteur clé pour ces outils de traduction.

Les fonctionnalités de traitement par lots et de pré-analyse des fichiers à traiter (Workbench et Translation Manager) sont essentielles à la bonne gestion d'un projet. Comme il a été dit plus haut, la rentabilité maximale provient de la traduction par lot de fichiers, pas du traitement interactif.

Ceci étant, le fait de posséder un logiciel donnant un environnement de traduction adapté (dictionnaires, traductions approchées) est un plus, et l'ensemble des OTAM1g assurent cette fonctionnalité.

On trouvera en Annexe 2 différents tableaux récapitulant les caractéristiques de quatre OTAM1g. Ces tableaux couvrent les thèmes suivants :

- Aligneur
- Langues traitées
- Traitement des formats
- Fonctionnalités particulières
- Caractéristiques des mémoires de traduction
- Caractéristiques des bases terminologiques
- Echange de données
- Connexion à un module de traduction automatique
- Comparatif des prix

Cette étude montre les limites de ces outils en particulier en ce qui concerne les possibilités d'adaptation linguistique de la solution et le transfert du format du fichier source vers le fichier cible. Ainsi, les performances de ces OTAM1g peuvent être divisées en quatre groupes :

- **Recherche et recopie d'un segment identique en mémoire** :
  Les OTAM1g traitent correctement cette fonction lorsque l'on reste au sein de documents de même format d'origine. Les résultats sont plus hasardeux lorsque l'on utilise des documents de formats différents. Entre documents de même formats d'origine (RTF par exemple), les OTAM1g ont cependant des difficultés à reconnaître la même phrase formatée autrement (gras, italiques). Ils interprètent alors la correspondance comme "approchée".
- **Recherche et traduction de phrases linguistiquement proches** : C'est une fonction mal gérée par les OTAM1g, en ce sens qu'ils ne sont pas capables de générer les différences linguistiques comme les pluriels, les conjugaisons, ou les simples substitutions de mots.

- **Transfert de format de la phrase source à la phrase cible** : Cette fonctionnalité n'est globalement pas assurée sauf dans deux cas : soit le format est déjà contenu en mémoire et est le même à appliquer dans la phrase à traduire, ou au pire est différent mais placé au même endroit ; soit le format s'applique à tout le segment à traduire.
- **Transfert des objets non linguistiques** : C'est ce qui fonctionne le plus mal. Dans le meilleur des cas l'objet est reconnu et placé à un endroit fixe dans la phrase source, ou est proposé pour être placé en mode interactif. Ce dernier point est crucial pour la localisation**{**xe "localisation"**}** (traduction de logiciels). Notre test n'a utilisé que des objets non textuels de documents. Dans la cas de ressources de logiciels, ces objets incluent par exemple des variables. La prise en compte et la gestion automatique de ces variables dans la traduction de ces ressources ferait gagner énormément de temps.

Les trois derniers points sont repris dans le chapitre suivant. Ces points sont en effet détaillés car ils constituent la motivation de la présente étude.

2. Les outils de TFM de première génération ont des faiblesses certaines

2.1 Introduction

Les résultats du test précédent sont explicites. Les Outils de Traduction Aidée par la Mémoire de première génération (OTAM1g) n'ont pas vraiment de problèmes pour proposer une solution exacte ou approchée au niveau de la similarité de cette solution. Ils ne sont cependant pas globalement capables de remplacer un mot automatiquement via leur dictionnaire, ou de donner sa flexion convenable. Leur faiblesse**{**xe "faiblesses"**}** est importante au niveau des formats, qu'ils peuvent transférer depuis la phrase source sur la phrase traduite, seulement dans le cas où la phrase entière porte sur ce format. Ils ne sont pas capables de formater correctement un groupe de mots qui ne porte aucun format en mémoire. Au niveau des objets imbriqués, la plupart des outils du marché reconnaissent ces objets (parfois implicitement, juste par leur "code"), proposent de les traiter en interaction, mais ne sont pas capables de les placer de façon automatique. Nous allons voir quelques-uns de ces problèmes en détail ci-après.

Pour ce chapitre, les tests pratiques ont été conduits sous le logiciel Workbench de la société Trados. Les résultats sont cependant comparables, si on se base sur un des trois autres produits. Ce qui peut changer est le pourcentage de similarité.

2.2 La solution proposée n'est pas reconstruite linguistiquement

2.2.1 Utilisation du dictionnaire

Test

Dictionnaire

source	click	on	the	right	button	left
cible	cliquer	sur	le	droit	bouton	gauche

Résultat 1

	source	cible
mémoire	Click on the right button.	Cliquer sur le bouton droit.
trad. attendue	Click on the left button.	Cliquer sur le bouton gauche.
trad. obtenue	Click on the right button.	Cliquer sur le bouton droit. (fuzzy: 85%)

L'adjectif "left" (gauche) n'est donc pas remplacé.

Explications

Il n'y a apparemment pas de prise en compte du lexique dans le processus de construction de la solution. En effet, ce processus consiste à retrouver la chaîne source la plus proche parmi les unités de traduction de la mémoire, et à proposer comme traduction la chaîne cible correspondante. Il ne comporte donc pas de vrai traitement linguistique.

Cette attitude a bien sûr des fondements. On ne peut remplacer à la volée un mot par son équivalent d'une langue à l'autre, cela demande une connaissance linguistique**{**xe "faiblesses linguistiques"**}** de la structure du texte traité, qui n'existe pas dans les outils de première génération.

Exemple :
Considérons le problème suivant :

	source	cible
mémoire	You must touch the button	Vous devez toucher le bouton
trad. attendue	You must not touch the button.	Vous ne devez pas toucher le bouton

Remplacer “not” par “ne pas” nécessite de savoir que “devez” est le verbe auquel il faut appliquer la négation, qui malheureusement en français se compose de deux parties qui doivent entourer le verbe, contrairement à l’anglais.

2.2.2 Mise au pluriel d’un nom au sein d’une phrase

Test

Dictionnaire

source	click	on	the	right	button	left
cible	cliquer	sur	le	droit	bouton	ø

Résultat 2

	source	cible
mémoire	Click on the right button.	Cliquer sur le bouton droit.
trad. attendue	Click on the right buttons.	Cliquer sur les boutons droits.
trad. obtenue	Click on the right button.	Cliquer sur le bouton droit. (fuzzy: 93%)

Explications

Pour effectuer cette mise au pluriel, deux connaissances sont nécessaires. Une lemmatisation doit d’abord donner la forme de base des mots “buttons”, et “boutons” (puis de “the”, “right”, “les” et “droits”). Puis une analyse de la phrase doit montrer que “the right buttons” et “les boutons droits” sont d’une part des groupes nominaux, et d’autre part qu’ils se correspondent. Une fois cela acquis, et pourvu que l’on possède des règles d’accord des groupes nominaux, on peut procéder à l’opération. Or aucune de ces connaissances, sauf peut-être dans certains cas la lemmatisation, n’est disponible dans les outils de première génération.

2.3 Gestion restreinte des formats de texte

Test

Dictionnaire

source	click	on	the	right	button
cible	cliquer	sur	le	droit	bouton

Résultat 3.1

	source	cible
mémoire	Click on the right button.	Cliquer sur le bouton droit.
trad. attendue	Click on the right **button**.	Cliquer sur le **bouton** droit.
trad. obtenue	Click on the right button.	Cliquer sur le bouton droit. (fuzzy: 97%)

Le logiciel n’est donc pas capable d’appliquer un nouveau format sur un mot à l'intérieur d'une phrase. Il est par contre capable d’appliquer un format sur une phrase entière.

Résultat 3.2

	source	cible
mémoire	Click on the right **button**.	Cliquer sur le **bouton** droit.
trad. attendue	Click on the right *button .*	Cliquer sur le *bouton* droit.
trad. obtenue	Click on the right *button .*	Cliquer sur le *bouton* droit.

Si un format à transférer existe déjà en mémoire (dans les parties sources et cibles), le logiciel est capable de récupérer pour la même phrase un format appliqué au même mot, mais d'attributs différents (ici l'italique remplace le gras).

Explication

Dans le cas où les équivalents sont dans le lexique, et où le formatage**{**xe "faiblesses de transfert du format"**}** enregistré en mémoire provient d'un type de document similaire à celui qui est en train d'être traduit, il n'y a pas de raison qu'un mécanisme ne puisse transférer le formatage correctement sur un mot donné : l'explication la plus simple est que cette fonctionnalité n'a pas été implémentée.

Cependant, le transfert de format ne peut se faire que si l'on a une interprétation du formatage, ce qui suppose une standardisation du formatage, ce n'est pas le cas dans les outils de première génération.

En effet les formats sont indiqués de la fa çon suivante :

```
Click on the right {¥i button}.
```

2.4 Pas de réelle gestion des objets imbriqués dans le texte du document.

Test

Dictionnaire

source	click	on	the	right	button
cible	cliquer	sur	le	droit	bouton

Résultat 4

	source	cible
mémoire	Click on the right button.	Cliquer sur le bouton droit.
trad. attendue	Click {hyperlink}on the right button.	Cliquer sur le bouton droit.
trad. obtenue	Click on the right button.	Cliquer sur le bouton droit. (fuzzy 70%, embedded object to placed)

Ainsi l'outil reconnaît l'objet, mais n'est pas capable de le placer correctement.

Explication

Le placement d'un objet non linguistique**{**xe "faiblesses de gestion des objets non linguistiques"**}** demande à la fois de comprendre qu'il s'agit d'un tel objet, donc une interprétation, mais aussi de maîtriser la structure de la phrase et des correspondances entre les structures source et cible, pour savoir où placer cet objet dans la structure cible. Le cas ci-dessus ne présente pas de difficultés concernant la structure, mais le cas suivant, où l'on doit interpréter le lien sur le sous-groupe nominal "right button", pose problème :

	source	cible
mémoire	Click on the right button.	Cliquer sur le bouton droit.

trad. attendue	Click on the right {hyperlink} button.	Cliquer sur les boutons droits {hyperlink}.
trad. obtenue	Click on the right button.	Cliquer sur le bouton droit. (embedded object to placed)

2.5 Niveau structurel de la redondance traitée et problème de la segmentation des phrases

Les outils de première génération segmentent les unités de traduction selon la ponctuation, la marques de structure (tabulations, retours chariots), et selon les objets non textuels. Cela signifie que, la plupart du temps, les unités correspondent à des phrases. Or les répétitions concernent souvent de petits groupes, et donc la traduction par analogie gagnerait à pouvoir traiter ces groupes plutôt que des phrases entières.

Mais cela n'est possible que si l'on sait reconstruire les phrases en combinant ces groupes. Cela demande d'en savoir plus sur la structure linguistique de la phrase à traduire.

2.6 Connectivité et coopération avec d'autres outils linguistiques

Les OTAM1g du marché sont souvent connectés à des outils de traduction automatique (fondés sur les règles) du commerce. Ainsi, Eurolang Optimizer est connectable à LanTmaster**{**xe "LantMark"**}**, Transit à Logos**{**xe "Logos"**}**, Workbench à Transend**{**xe "Transend"**}** (d'Intergraph), et Translation Manager à LMT.

Ces logiciels de traduction automatique permettent de traduire les phrases pour lesquelles aucune unité de traduction similaire n'a pu être trouvée dans la mémoire par le logiciel de TFM, mais en général avec une qualité médiocre.

Dans ce cas, l'échange entre les deux types de logiciel est limité aux phrases entières, et la loi du tout ou rien est appliquée. Il n'y a pas de combinaison possible des traductions provenant de la TFM avec celles provenant de ces outils de traduction fondé sur les règles. Souvent, les dictionnaires ne sont pas partagés. Toutefois, dans le cas de Trados Workbench et Transend, une base de données terminologique gérée par Multiterm**{**xe "Multiterm"**}** peut être partagée.

3. Pourquoi ces faiblesses proviennent de la conception de la structure de la représentation interne, et premières idées d'amélioration des performances

3.1 Structures de représentation des données

3.1.1 Introduction

Les données nécessaires à la traduction fondée sur la mémoire sont présentes à deux niveaux. Le premier est la structure interne de représentation des données (SIRD), nécessaire aux traitements de traduction. Le second concerne les mémoires de traduction elles-mêmes, les données externes (SERD), c'est-à-dire le stockage de l'information par unités de traduction, donnant la correspondance entre une expression source et une expression cible. La SIRD utilise aussi les informations provenant de la SERD.

3.1.2 La représentation interne des données au niveau des traitements

Principe

L'utilisation des mémoires de traduction se fait au moment du processus de traduction. Une structure interne**{**xe "structure de représentation interne pour les traitements"**}** de représentation (SIRD**{**xe "SRDI"**}**) est alors utilisée. Cette structure utilise et interprète les données provenant du document à traduire, et aussi de la mémoire de traduction, pour procéder à la traduction du document en cours. Cela sous-entend bien sûr de disposer de transducteurs depuis les formats de documents que l'on veut traiter vers la structure qui sera stockée en mémoire. De même les transducteurs inverses sont nécessaires. Transit, Translation Manager, et Optimizer utilisent une telle interprétation. Cependant leur représentation, qui retranscrit tout ce qui est trouvé dans le document d'origine, juxtapose le codage de la mise en forme (qui est transformé en balises) avec celui du texte, dans une expression linéaire.

Plusieurs types de données sont représentées en interne. Dans Workbench, par exemple, il existe en plus de la SIRD une matrice d'occurrence des caractères qui permet, sous forme de réseau neuronal, de calculer les similarités entre chaînes. Ce type de données est spécifique à une technique particulière. Nous n'en parlerons pas ici. Nous nous concentrons plutôt sur la représentation interne du document.

Si la SIRD des OTAM1g est tenue secrète par les éditeurs de ces logiciels, les résultats de traduction laissent cependant supposer que l'analyse linguistique et les structures nécessaires n'y sont pratiquement pas présentes, puisque ces outils de première génération ne sont pas capables de substituer un mot, ni de changer la flexion d'un mot pour produire une traduction exacte.

En fait la SIRD est proche de celle utilisée pour la mémoire (c'est la même dans le cas de Transit) et unidimensionnelle. Dans le flot linéaire de données, ces outils concentrent l'ensemble des informations provenant du document et des mémoires. La stratégie est essentiellement de recourir à des "balises" pour marquer les informations de mise en page et les objets non textuels. Ces balises sont insérées au sein du texte lui-même, comme dans le schéma suivant :

<balise1> mot1 mot2 <balise2> mot3 <balise4><balise5>….

Figure 12 : Principe des représentations de données actuelles pour les OTAM1g

Ces représentations sont souvent basées sur SGML, comme celle d'Eurolang Optimizer (ELDIF**{**xe "ELDIF"**}**), celle de Transit ou encore celle de Translation Manager. Si l'idée semble être bonne pour séparer le texte du "codage" du document, elle n'est pas suffisante.

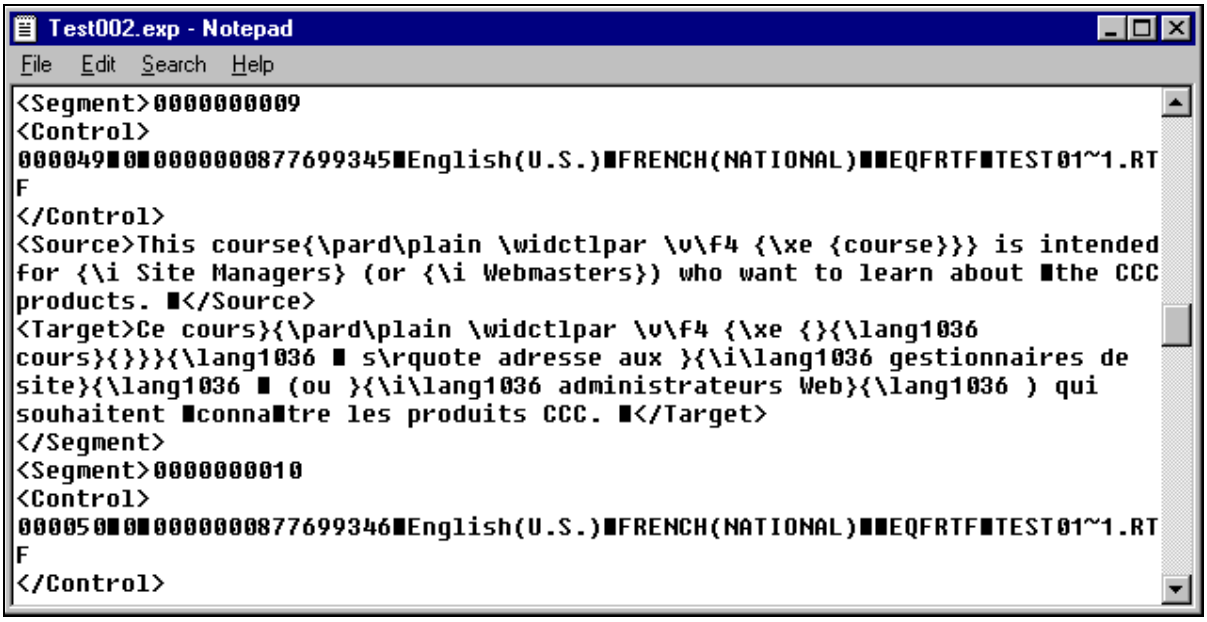


Figure 13 : Structure de représentation interne de Translation Manager.

Les SIRD des OTAM1g sont responsables des erreurs de segmentation et de mauvaise correspondance entre segments.

Comme les OTAM1g se basent sur de telles structures pour comparer les chaînes à traduire et provenant de la mémoire, cela mène à des erreurs comme le montrent ces deux points :

- de mauvaises traductions ou des traductions incomplètes sont proposées lorsque la mise en forme du segment provenant de la mémoire et du segment à traduire sont différentes, comme dans l'exemple suivant :

  Exemple :

  Mémoire : This course is intended for Site Managers (or Webmasters) who want to learn about the CCC products.

  Document : This course **{¥pard¥plain ¥widctlpar ¥v¥f4 {¥xe{course}}}** is intended for **{¥i** Site Managers**}** (or **{¥i** Webmasters**}**) who want to learn about the CCC products.

  Aucune solution n'est alors proposée pour cette phrase, alors qu'elle existe en mémoire (cf. Figure 10).
- Des erreurs au niveau de la segmentation. Dans l'exemple ci-dessus **{¥xe{course}}** représente un index provenant d'un fichier RTF (cf. Figure 11). Lorsque Transit essaie de segmenter cette phrase, l'index joue le rôle de séparateur, et deux segments sont créés

  segment 1 : "This course"

  segment 2 : "is intented for Site Managers (or Webmasters) who want to learn about the CCC products"

  Alors la recherche de segments similaires dans la mémoire ne donne rien pour le premier puisqu'aucun ne correspond, et donne une solution "approchée" pour le second car seul le segment entier est disponible (cf. Figure 10).

Remarquons qu'il est possible de parer à cet inconvénient en définissant une nouvelle "expression régulière" sous Transit qui permettra de ne plus tenir compte de ce genre de balises d'index. Mais l'opération devra être effectuée pour toutes les balises qui pourraient gêner la segmentation, et ceci n'est donc pas une solution générale.

Quelle analyse linguistique ?

Cette représentation ne permet pas non plus de traiter correctement le texte au niveau linguistique. Un niveau d'analyse supérieur est nécessaire, et il devrait être représenté dans la structure de données interne de façon "séparée", tout en restant lié au texte de départ; ceci permettrait des traitements linguistiques spécifiques.

Jusqu'où doit aller l'analyse linguistique nécessaire ? En particulier, quelle part de cette analyse doit être stockée en mémoire, et quelle part doit être construite dynamiquement lors de la traduction ? Des questions de temps de traitement se posent. En effet, l'étape de recherche directe sur toutes les chaînes est très coûteuse.

Par exemple, on peut penser qu'il serait judicieux de stocker en mémoire un ensemble beaucoup plus petit de correspondances de patrons de traduction, comportant l'analyse structurale (préalablement effectuée, puis stockée) sous forme de correspondance d'arbres. Lors de la pré-traduction, on n'effectuerait alors qu'une analyse élémentaire et on établirait une correspondance en appliquant ces patrons, plutôt que de procéder à une analyse complète (lexicale, syntaxique, sémantique).

Une autre idée serait de garder une mémoire "pauvre" d'équivalences entres chaînes, et de procéder à une analyse profonde à chaque traitement. Cela suppose alors de posséder des analyseurs efficaces auxquels on puisse faire appel dans la phase de traduction. Là encore se pose la question de savoir jusqu'où l'on peut aller avec tel type d'analyseur (morphologique, lexical, syntaxique, sémantique, conceptuel). En particulier, on doit être vigilant pour ne pas reconstruire un traducteur fondé sur les règles.

Robustesse et interprétation

Certains logiciels de TFM comme Workbench n'acceptent que les fichiers d'un format donné (par exemple "Microsoft RTF"). C'est alors le format RTF qui joue le rôle de langage de représentation interne. Dans ce cas, si l'on veut traiter un document enregistré sous un autre format, il faut utiliser les convertisseurs propriétaires (ceux de Microsoft Word, par exemple). Cela a l'avantage ne pas devoir mettre à jour les tables de conversion à chaque changement de spécification de formats (qui sont fréquents) ; le travail est astucieusement délégué aux techniciens de Microsoft, société qui de toutes façons a intérêt à ce que ses convertisseurs soient à jour des dernières modifications. Cela a par contre le désavantage dont on a parlé plus haut de faire dépendre l'analyse du texte de ce format RTF, et aussi d'être complètement dépendant d'un éditeur.

On peut donc se poser la question suivante : comment adopter une attitude analogue, qui fasse que le travail perpétuel d'adaptation des filtres aux formats d'édition du marché soit allégé, et en même temps qui permette une interprétation standard ? Nous verrons au chapitre 2 comment cela peut être résolu.

3.1.3 Représentation au niveau externe des mémoires de traduction

L'aspect de la représentation externe**{**xe "structure de représentation externe"**}** (SERD**{**xe "SRDE"**}**) de la mémoire est donnée par l'Annexe 1.1. pour Workbench, et l'Annexe 1.2. pour Translation Manager. Ce qui est stocké est une copie de la chaîne du texte et des attributs éventuels, qui ont été trouvés dans les documents source et cible d'origine. Aucune analyse n'a donc été menée, il s'agit simplement d'une retranscription pauvre.

Standardisation de la représentation du texte lui-même

Il est bien entendu nécessaire de pouvoir utiliser un support correct des langues en mémoire. Cela est assuré par la plupart des outils actuels, et doit être conservé. Il s'agit donc d'utiliser par exemple les recommandations du groupe "Text Encoding Initiative**{**xe "Text Encoding Initiative"**}**" (TEI), et de veiller en particulier à prendre un codage du texte qui puisse supporter le

traitement du plus grand nombre de langues possibles, comme Unicode{xe "Unicode"}. Mais ce texte doit être libre de tous autres éléments du document, de telle façon que les traitements habituels concernant les chaînes de caractères de texte puissent s'appliquer directement.

Standardisation de la représentation des éléments non linguistiques

La mémoire doit pouvoir servir à traduire un document indépendamment de son format d'origine. Aussi, **si toutefois on juge nécessaire de garder les informations de mise en page au niveau des mémoires (c'est une question à résoudre)**, il nécessaire de procéder à une standardisation de la représentation des éléments non linguistiques tels que le formatage et les objets imbriqués dans le texte de la mémoire. Cela demande donc la construction d'un module d'interprétation capable non seulement de dire si un mot est formaté en gras, mais aussi si une balise d'index y est appliquée. Il faut alors garder toute l'information concernant cette balise d'index, information qui peut rester au format propriétaire pour partie. Il n'est d'ailleurs pas nécessaire de connaître exactement cet objet non linguistique, mais simplement sa nature et le traitement qui doit lui être appliqué. Cela peut aider à ne pas devoir interpréter toute information provenant d'un document possédant un format propriétaire, et maintenir la robustesse de l'interprétation de ce document. Ce problème est alors similaire à celui de la représentation des SIRD.

Chaînes de caractères, ou structures plus informatives

Les unités de traduction des mémoires actuelles sont composées de "bi-textes" ne comportant que la chaîne source et la chaîne cible.

Si ce type d'information est simple à stocker, et favorise en particulier l'échange des mémoires de traduction d'un système à l'autre (bien que cet échange n'existe actuellement presque pas), la connaissance linguistique n'y est pas présente, à part le fait que l'une des phrases est la traduction de l'autre.

Il serait plus intéressant de posséder d'autres informations linguistiques, comme par exemple :

- un alignement au niveau des mots, et pas seulement au niveau du segment,
- les lemmes des mots des textes source et cible,
- leur catégorie grammaticale, et d'autres traits linguistiques comme le genre ou le nombre.

Une structure arborescente plutôt que linéaire (la suite des mots ou des lemmes et leurs propriétés) permet de véhiculer beaucoup plus d'information. D'autre part, il paraît intéressant d'étudier la possibilité de stocker des schémas de traduction liant les structures syntaxiques plutôt que les chaînes ; pour cela aussi, des structures arborescentes paraissent a priori mieux adaptées

3.2 A propos de quelques procédures

3.2.1 Calcul de la similarité des chaînes

Dans les systèmes industriels actuels (1998), la similarité des chaînes{xe "similarité des chaînes"} est calculée non seulement sur la correspondance strictement linguistique des deux chaînes à comparer, mais aussi selon le formatage. Ce calcul de similarité peut s'effectuer sur la base d'informations complémentaires comme l'auteur ou la date de l'alignement. Si inclure de telles informations peut a priori être une idée excellente au niveau de la gestion de projets de traduction industrielle, cela cache quand même l'incapacité pour ces systèmes à gérer séparément le formatage, comme cela vient d'être vu plus haut. La stratégie consiste alors à compenser cette mauvaise gestion du formatage en appelant l'attention du traducteur. On peut par exemple afficher le taux de traduction inférieur à 100%. Cela est justifié par le fait qu'une phrase, pour être traduite complètement, doit aussi posséder le bon formatage de son texte.

Lorsqu'un traducteur humain traduit une phrase d'un document formaté, le cœur de son travail, et ce qui fait sa spécificité, consiste a trouver l'équivalent dans la langue cible. Il applique le formatage immédiatement après cette étape de traduction, et l'effectue de manière très naturelle par similarité avec la source, aux conventions culturelles près. Cette opération de mise en page peut prendre autant de temps au traducteur que la traduction elle-même (cette dernière se fait très rapidement chez les professionnels). Cette opération devrait être transparente dans les systèmes automatisés de traduction et permettrait ainsi de gagner beaucoup de temps. Si elle ne l'est pas dans les systèmes actuels, c'est simplement parce qu'ils n'ont pas la capacité de traiter automatiquement le formatage du texte traduit. Ceci est assez contraignant pour les traducteurs qui doivent reprendre les phrases sur lesquelles la traduction a été effectuée, mais pas la bonne mise en forme. Le traducteur se transforme en "metteur en forme", alors que sa spécialité est de traduire.

Pour revenir au calcul de similarité entre les chaînes de deux phrases d'un document formaté, il semble donc préférable de ne pas inclure des considérations sur le formatage dans le calcul de la similarité des phrases dans la phase de comparaison entre le segment à traduire et les segments candidats de la mémoire.

Cela n'empêche pas d'inclure ces informations de différence de formatage au niveau de la présentation de la traduction pour le traducteur, ou pour le gestionnaire du projet de traduction, bien sûr. Il devrait ainsi exister deux calculs de similarité différents : l'un pour la traduction proprement dite par la machine, l'autre pour l'information du traducteur.

3.2.2 Segmentation et format de texte

Les OTAM1g se basent sur la structure propriétaire de format du texte à traiter pour effectuer la segmentation**{**xe "segments"**}** des unités de traduction. De savantes règles de découpage du texte sont souvent proposées pour arriver à des unités cohérentes. On peut ainsi spécifier qu'une unité de traduction commence par une majuscule, et se termine par un point (simple, double, d'exclamation, d'interrogation, point virgule, etc.). Des règles correctrices pour, par exemple, ne pas considérer les points après les abréviations ou les chiffres comme des fins de phrases sont alors nécessaires.

Or ces règles sont souvent insuffisantes et prises en défaut, par exemple par les codes relatifs aux objets imbriqués. Cela ne se produirait pas si avant la segmentation du texte, une interprétation était appliquée, comme expliqué plus haut.

Conclusion

Nous avons pu voir l'importance de l'ensemble des éléments d'un document, et pas seulement des chaînes de caractères, dans la traduction automatique de documents réels.

L'industrie de la traduction montre que les gains de traduction les plus significatifs sont apportés par des outils de traduction automatique autonomes, avec lesquels le traducteur humain n'a que peu ou pas d'interaction. Or les OTAM1g actuels demandent un haut niveau d'interaction en particulier pour corriger des variations linguistiques simple d'un segment provenant de la mémoire (flexions, substitution de mots). Ils présupposent aussi que le réviseur humain place correctement la mise en forme et les objets non textuels car ils ne n'assument que très mal cette tâche.

Les OTAM1g se basent sur une structure de représentation interne linéaire et hétérogène. Dans ce type de structure, les balises de codage sont insérées séquentiellement dans le flot de texte du document. La structure est alors homomorphe, parfois isomorphe à celle utilisée dans le codage des éditeurs du marché.

Cette structure ne permet pas un traitement séparé de chaînes de caractères (du texte nu), et des éléments non textuels. Ainsi par exemple le calcul de similarité entre deux segments est biaisé par la présence de code ne représentant pas des mots. De même la segmentation du texte en unités de traduction, opération essentielle pour la bonne marche du processus de traduction, est souvent perturbée par la présence de code supplémentaire à celui des chaînes de caractères des mots.

Il semble donc qu'il faille s'orienter vers une structure de représentation des données interne aux OTAM1g qui permette une séparation des dimensions des données (dimension texte, dimension non texte).

Chapitre 4

Fondements de la structure

Contenu du chapitre

Introduction

Le *critère de compatibilité* avec les outils actuels implique que l'Outil de Traduction Fondée sur la Mémoire (TFM) dispose d'une représentation interne des données qui puisse communiquer à la fois avec les documents d'édition du marché, mais aussi avec les standards d'échange linguistique, en particulier d'échange de mémoires de traduction. On n'oubliera pas, de plus, de respecter le *critère de compatibilité avec Internet*. Ces contraintes se traduisent en particulier par les conditions suivantes :

- Le format des documents traduits doit pouvoir être interprété et représenté par la Structure Interne de Représentation des Données (SIRD) que l'on cherche
- De la même façon, il doit être possible de générer en retour ces formats sur les documents traduits à partir de la SIRD
- La SIRD doit être proche des standards d'échange de documents électroniques

Un des enseignements de la première partie de cette étude est que la Structure Interne de Représentation des Données des OTFM doit pouvoir représenter les différentes dimensions de l'information contenue dans un segment de document. Il est néanmoins nécessaire de pouvoir conserver l'information brute sous un format standard et transportable, en particulier via Internet, pour assurer l'échange standard des documents à traduire.

Dans la suite, une étude des différents types de documents et de leur format physique est proposée au paragraphe I. Il en est tiré un ensemble de données qu'il faut représenter dans la SIRD et une étude des formats logiques et des formats internes des outils actuels montre quelles sont les possibilités de choix pour la structure de la SIRD. Le paragraphe II montre alors que les fichiers codés selon XML conviennent pour la forme primaire de la SIRD. Le paragraphe III explore les possibilités de représentation des données analysées de la SIRD, et propose d'éclater les données en autant de treillis (les étages) que de types de données présentes dans le segment. Enfin la conclusion fixe l'aspect général de la structure complète.

1. Types de documents envisagés pour notre étude et variété des formats

Introduction

Nous reprenons ici quelques documents provenant de projets de traduction réels, afin de dégager une liste d'éléments de base que doivent pouvoir gérer les logiciels de TFM. Les types de documents traités sont obligatoirement des documents sous forme électronique bien sûr.

1.1 Documents concernés

1.1.1 Types de documents à traduire

Manuels techniques

Des exemples classiques de manuels techniques sont donnés par les manuels de maintenance des engins (voitures, engins militaires, outils industriels, engins de travaux publics). Les rapports d'analyses ou descriptions techniques font appel aux mêmes éléments documentaires.

Nous voyons le type de document "manuel technique" essentiellement comme un document comportant du *texte*, des figures avec *légendes*, et des *tableaux*. La présence de légendes dans les figures est une donnée essentielle qu'il faut pouvoir traduire. La gestion des légendes du tableau ne sera pas du ressort de cet exposé (rappelons que nous nous restreignons à la gestion de l'intérieur d'unités de traduction). Nous considérons cependant que chaque légende correspond à une unité de traduction, ou à une suite de telles unités.

Manuels d'aide en ligne

La figure suivante représente un fragment de manuel en ligne sous Microsoft Word. Ce manuel est un hypertexte contenant des *hyperliens* comme " ex "{aa¥¥ AX016.DOC-1001"}" qui ne sont normalement pas visibles par le lecteur. Des *images* ou *icônes* peuvent s'intercaler.

Write a Business Letter

IMAGE

Don't know how to get started writing a letter, or maybe you're working under a tight deadline? The Letter Wizard or the letter templates provide a fast and easy way to create a business or personal letter and matching envelope. You can choose from three professional designs, print on letterhead or plain paper, and even select a prewritten business letter, such as a request for payment.

{ex "aa¥¥ AX016.DOC-1001"}

Figure 1 : Un manuel en ligne sous Microsoft Word.

Fichiers électroniques de cours

Les cours électroniques font appel pour leur édition à la fois aux possibilités de mise en forme classique, et à des objets non-textuels, comme les *références* sous forme d'hyperliens (comme les marques d'index), ou des *objets* plus ou moins complexes (les champs de Word

par exemple). Nous présentons ici le fichier représentant la mémoire de traduction d'IBM Translation Manager associée à la traduction de ce document de l'anglais vers le français et quelques problèmes linguistiques liés à cette traduction. Ce type de documents et les problèmes que leur représentation posent, ont déjà été vus dans la première partie (Chapitre 3).

Cette représentation prend en compte le texte lui-même, mais aussi les différents morceaux de code RTF qui servent à gérer l'aspect du document sous un éditeur de fichiers RTF. Translation Manager ajoute ses balises (de type SGML) pour représenter ce document, mais garde entre ces balises l'information sous sa forme d'origine.

Transparents

Il existe des logiciels comme Microsoft Power Point pour éditer les transparents pour les présentations. Ces documents sont souvent formés de chaînes de caractères isolées, qui assemblées forment de la prose ou des légendes de schémas. Les chaînes ne contiennent pas vraiment de nouveaux types d'éléments.

Sources de programmes et fichiers ressources de programmes

Fichers sources de programmes

Voici un extrait de programme JAVA montrant la déclaration d'une classe. Il est intéressant de noter que pour localiser ce fichier, ce qui est à traduire est un sous-ensemble de l'ensemble des *chaînes* entre guillemets.

```
public final class HexCalc extends Applet {
private Display display;
public void
init()
{
        KeyPad keypad;
        Font f = new Font("Helvetica", Font.BOLD, 20);
        display = new Display();
        keypad = new KeyPad(display);
        display.setKeyPad(keypad);
        keypad.setFont(f);
        setLayout(new BorderLayout());
        add("North", display);
        add("Center", keypad);
        display.onScreen("Nothing");
}
```

Figure 2 : Fichier source en JAVA

Dans cet exemple, " Helvetica " n'est pas à traduire, alors que " Nothing " l'est. Ainsi la décision de traduction dépend de la syntaxe du programme. Idéalement, un outil de traduction devrait pouvoir distinguer ce genre de nuance. Cela suppose d'une part que l'information concernant la syntaxe d'un tel programme puisse être indiquée au logiciel de traduction, et d'autre part qu'il est capable d'interpréter cette information pour agir en conséquence. La structure interne du logiciel doit donc être capable de représenter cette information.

Chaque chaîne à traduire représente dans la structure à venir une unité de traduction (ou segment de traduction). Cette structure gère donc l'information au sein de cette unité de traduction, et non celle du segment par rapport au reste du document. Nous n'aborderons donc pas la question précise de savoir comment interpréter la structure d'un tel document, mais

plutôt comment représenter et interpréter l'intérieur de chaque segment. Nous supposons pour la suite de ce chapitre que l'extraction des segments de traduction a été faite, et que nous disposons de l'ensemble des segments à traduire.

Dans un programme en C, par exemple, il se peut que la chaîne à traduire se finisse par "¥n", comme dans : "Entrez un chiffre¥n". Ces chaînes comportent donc des *instructions* accolées, qui faut aussi pouvoir traiter lors de la traduction.

Ressources de programmes

Les ressources de programmes, qui contiennent des collections d'objets appelés par les programmes (des expressions de menus, de boutons, de messages d'aide ou d'erreur, des images, des sons, etc.) contiennent souvent des chaînes de caractères qui doivent être traduites. Ce sont ces types de messages qui apparaissent par exemple dans les fenêtres d'avertissement de l'écran d'un ordinateur, ou dans les messages d'une imprimante. Ces fichiers de ressources ne posent pas de difficultés au niveau de la mise en page en général (pourvu que l'on garde tous les caractères, y compris les blancs, voir explication ci-dessous), mais demandent par contre de pouvoir traiter l'expression textuelle des *variables* comme un "*objet textue*l**{**xe "objet textuel"**}**". Dans cet exemple, la variable "%s1", exprimée en caractères, se comporte comme un mot de la phrase car elle remplace le nom d'un objet cité dans cette phrase (une DLL dans ce cas). Il est nécessaire de pouvoir représenter cet élément de façon adéquate.

On notera que les espaces du second message (matérialisés par des points, entre "LINES" et "Coud not.." sont à conserver car ils déterminent la place du message dans la fenêtre qui les affiche. En outre ce genre de document fait appel à une base de données linguistiques qui n'est pas un dictionnaire classique, et qui peut être gérée par une base de données terminologiques (comme TermStar ou Multiterm). Il est important de pouvoir utiliser cette base de données en combinaison avec un dictionnaire classique, et donc de pouvoir prendre en compte ces *entrées terminologiques* dans la représentation des données internes.

MSG_COULD_NOT_LOAD_DLL."Could not load the DLL (%s1)! Load Library returned: %s2"
MSG_COULD_NOT_GET_INFO_ABOUT_LOCKED_LINES
……."Could not get information about Locked Lines from the database!"
CAP_GNVFILED_SAVE_AS…."Save As"

Figure 3 : Extrait de fichier de ressources

Pages de sites Interne ("pages Web")

```
<HEAD>
<TITLE>NTT Home Page</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<IMG ALT="" SRC="/ntt/header.gif" WIDTH=557 HEIGHT=92>
<H1>Welcome to the NTT Home Page!</H1>
Japanese version is <A href="index-j.html"><b>here</b></a>.
Classic version is <A href="index.old.html">here</a>.<p>
This is the Nippon Telegraph and Telephone Corporation grass-roots
WWW home page, in JAPAN.<br>
```

```
Official WWW home page is
<A HREF="http://info.ntt.co.jp/index.html"> <b> NTT DYNAMIC LOOP INFORMATION </b></a>.
<HR>
<STRONG>
We finished
<A HREF="/WHATSNEW/ntt.co.jp.html">changing the domain name of NTT</A>.
 Thank you for your cooperation.
.....
```

Figure 4 : Extrait de page HTML (page d'accueil du site Internet de NTT)

Les pages WEB sont le type de document dont le volume de traduction augmente de plus en plus. On veillera à ne pas confondre "pages Web" et "format HTML". En fait le format HTML est aussi utilisé de plus en plus pour d'autres types de documents. Par exemple Microsoft a entrepris de créer ses nouvelles aides en lignes au format HTML.

Le domaine de ces pages est souvent d'ordre général, et elles forment en fait un type de document bien particulier. Un page Web comporte des éléments classiques qui sont les *objets multimédia* (icônes, images, son, vidéo), des *hyperliens*. Le codage HTML comprenant des balises qui peuvent contenir des chaînes (le nom des fichiers dans les exemples ci-dessus) éventuellement à traduire, on n'oubliera donc pas ces "*chaînes imbriquées*".

Documents Multimédia

Les pages Web vues précédemment en sont un exemple, mais il en existe d'autres comme les CD multimédias. Ces documents comportent du texte, des hyperliens, et des *données multimédias* comme des icônes, des images, des sons ou des vidéos. On peut être amené à localiser ces données [Semmar 1999 (à paraître)], mais ce ne sera pas l'objet de notre propos. Il reste à noter que certaines icônes, images, ou vidéo comportent des *chaînes imbriquées* de texte à traduire.

Contrats

Les contrats sont des documents extrêmement simples dans leur structure, et très intéressants de part leur caractère répétitif. Ils n'apportent aucun élément documentaire supplémentaire mais restent un type de document important.

Documents qui ne seront pas traités

Dans les documents envisagés ne sont pas incluses les œuvres littéraires comme les romans, les pièces de théâtre, les essais, les poèmes, ou les livrets d'opéra. Les discours politiques, affiches électorales et publicités ne seront pas pris en considération ici. La version électronique de ce genre de documents ne pose en général pas de problèmes de représentation. C'est plutôt leur contenu qui en pose ici : il est trop riche et non répétitif aussi bien dans sa structure que dans son vocabulaire pour pouvoir être traduit efficacement par des logiciels fondés sur la mémoire.

1.1.2 Résumé des types d'éléments à représenter

Dans la description de quelques types de documents que l'on vient de voir, un certain nombre d'éléments documentaires (signalés en italique) sont apparus et doivent être pris en compte car ils peuvent apparaître au sein d'une unité de traduction. Nous les listons ci-dessous, en ajoutant ceux qui ont été omis :

- chaînes textuelles

⇒ mots
⇒ instructions de programmes
⇒ variables
⇒ chaînes imbriquées dans des scripts ou hyperliens
⇒ codes de procédures imbriquées (script JAVA en HTML par exemple)
⇒ légendes de schémas
⇒ contenus de cellules de tableaux

- objets non linguistiques
  ⇒ mise en forme locale sur caractères, mots, et paragraphes
  ⇒ champs
  ⇒ images,
  ⇒ sons
  ⇒ boutons
  ⇒ hyperliens
- références
  ⇒ index
  ⇒ marques de tables des matières
  ⇒ instructions incluses dans le texte en caractères
- marques de révisions

1.1.3 Formats propriétaires

Adobe Frame Maker

C'est un des logiciels d'édition structurés les plus utilisés. La page 7 du chapitre 1 donne une illustration d'un texte sous Frame Maker.

Adobe MIF

MIF est un format d'échange propriétaire pour les documents Frame Maker. La figure ci-dessous représente un extrait du codage d'un document FrameMaker (son aspect sous l'éditeur Frame Maker peut être vu à la page 7 du premier chapitre).

```
_D“t Œÿð çûç__þ_ _'ô__B _› _è ^^_D“t Œÿð çûç__þ___ -d
_ _§^ààÑÑ²à _þ __¦ü _ _ _ _ _$
Ò_m ¦ 2ost tropical forests form a_green band around the_ _ _ __
Ò_m¦¦,earthÕs center extending roughly 10 degrees¦, (_ _ __
Ò_m¬^ .north and south of the Equator. Amazing as it__8 _ _ __
Ò_mà- /may seem, these forests occupy less than eight {^¦ H _ _ __
Ò_m|¦ 2percent of the EarthÕs entire land mass, yet they _X _ _ __
Ò_m 0account for nearly half of all the growing wood ¦u h _ _ __
Ò_m,è 1on the planet and are the home for two fifths of - x _ _ _D
Ò_mà¬ #EarthÕs animal and plant species.
§ { ‘ _ _ _ _l
Ô_z,è ___¼_Tropical Rain Forests
¤ _ _ _ _$
Õ_m{û 2At least a_quarter of all pharmaceutical products ´ _ _ __
Õ_m¬- 0are derived from tropical rain forests. An even ÿë Ä _ _ __
Õ_m'ó -more impressive statistic, when you consider __Ô _ _ __
```

Figure 5 : Codage MIF

Le codage consiste en une série de balises (soulignées ici) entourant les chaînes de texte. Il est volontairement crypté avec une série de caractères non ASCII 7 bits, et ce document n'est pas auto-explicatif.

Microsoft RTF

Le format RTF est en fait un format d'échange pour les documents textuels de Microsoft. Un document au format RTF se présente comme une chaîne de texte entrecoupée de balises et de parenthèses. Cela donne des documents dont le codage ressemble à celui-ci :

```
....{¥*¥pnseclvl7¥pnlcrm¥pnstart1¥pnindent720¥pnhang{¥pntxtb (}{¥pntxta )}}
{¥*¥pnseclvl8¥pnlcltr¥pnstart1¥pnindent720¥pnhang{¥pntxtb (}{¥pntxta )}}
{¥*¥pnseclvl9¥pnlcrm¥pnstart1¥pnindent720¥pnhang{¥pntxtb
(}{¥pntxta )}}¥pard¥plain ¥widctlpar ¥f4¥lang1036
Cette phrase comporte un mot en {¥b gras}, en {¥i italique}, deux en {¥b¥i italique et
gras}, et une balise d'index{¥pard¥plain ¥widctlpar ¥v¥f4¥lang1036 {¥xe {balise
d'index}}}.
¥par }....
```

Figure 6 : Fichier RTF

Microsoft Word

Ce traitement de texte qui est devenu un standard, base la représentation de l'information contenue dans ses documents sur une structure multilinéaire synchronisée. Il s'agit de bandes parallèles contenant l'une le texte, l'autre les informations de mise en page. Un système de correspondance permet de lier un formatage aux mots formatés.

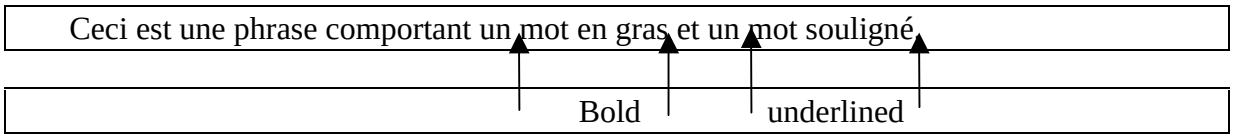


Figure 7 : Fichier Word

Interleaf

Interleaf est un exemple de traitement de texte de structure "ensembliste" utilisé dans l'industrie surtout pour les manuels techniques. "L'éditeur d'Interleaf considère un document comme un ensemble ordonné de composants typés...un rapport peut utiliser les types titre, auteur, résumé, titre de section, titre de sous section, et paragraphe" [André, Furuta et Quint 1989]. L'utilisateur peut définir lui-même ses composants. Même les illustrations peuvent contenir des "sous-composants". Chaque type de composant possède des propriétés dont certains attributs peuvent être modifiés au niveau de l'instance du composant utilisée dans le document.

Editeurs d'HTML

Il existe un nombre très important d'éditeurs de HTML. De plus les éditeurs communs (Frame Maker, Word, Interleaf, Claris Works) possèdent maintenant tous des filtres depuis et vers HTML (la version traitée peut cependant varier). Ce format logique utilisé en tant que format physique par ces éditeurs a l'avantage de devenir de plus en plus universel. Il est une instance de la famille des documents SGML que nous décrivons au paragraphe suivant.

Editeurs de SGML

Un certain nombre de sociétés spécialisées proposent des modules d'édition de documents respectant les spécifications SGML (Arbor Text, Metal, etc.). Le format logique SGML est décrit plus loin.

Autres types

Les protocoles de représentations des documents physiques dernièrement cités se basent généralement sur un "langage de marques" [Joloboff, p86]. Interleaf échappe à cette règle comme expliqué ci-dessus. Il existe des représentations plus structurées telles que Griff, qui ne sont pas vraiment utilisées dans les documents courants. Il existe un autre type de langages : les langages déclaratifs comme Scribe, Latex et GML (IBM) qui a donné la famille SGML, le plus connu aujourd'hui, dont nous parlons ci-dessous.

1.2 Formats d'échange

1.2.1 La famille SGML

Généralités

Le "Standard Generalized Mark-up Language" (SGML) est un metalangage fournissant un standard d'édition et d'échange de documents reconnu par l'International Organization for Standardization (ISO), qui est apparu au début des années quatre-vingts (sur la base d'un format d'échange mis au point par le Dr Goldfarb d'IBM : GML [Goldfard 1990]). Il s'est stabilisé entre 1986 et 1988, et est resté stable depuis. Cette stabilité reste exemplaire dans l'histoire de l'édition électronique, vu les possibilités qu'offre SGML [Marshall 1996].

L'idée qui le guide est de coder de façon déclarative un document par sa structure logique, son architecture. Ainsi le document peut être divisé par exemple en groupes de documents, documents, chapitres, sections, groupes de paragraphes, paragraphes, groupes de phrases, phrases, groupes de mots, mots, caractères.

Une fois la structure fixée, la mise en forme du document consiste alors à appliquer tel ou tel style sur telle ou telle structure logique du document via une feuille de style. Cela implique donc qu'il n'est pas possible, ou du moins pas naturel dans ce formalisme d'appliquer la même mise en forme à cheval sur deux parties structurales distinctes (un gras ne peut commencer sur la deuxième moitié d'un paragraphe, et finir sur la première moitié du paragraphe suivant). Les spécialistes de SGML argumenteront dans ce cas que le document est mal structuré. Cela impose donc une réelle restriction au niveau de la façon de décrire un document, mais c'est pour le meilleur car le traitement de l'information dans le document s'en trouve simplifié, sans que le pouvoir d'expression en soit réellement atteint car la mise en forme suit presque exclusivement la structure du document.

La structure du document est décrite à l'aide d'un langage exprimé par une grammaire, la Définition du Type de Document (DTD), qui peut être complètement définie par le rédacteur. L'architecture du document n'est donc pas imposée par SGML, seule la façon d'écrire une DTD est imposée (le metalangage, pas le langage). Il en découle que les procédures peuvent être écrites un seule fois pour traiter l'ensemble des documents de la famille SGML. Cela permet une souplesse extrême qui a mené pour ne citer qu'un exemple d'application, au langage HTML utilisé pour écrire les pages Internet.

L'application du langage se traduit par l'introduction de marques génériques au sein du document, décrivant sa structure. Un document SGML contient donc à la fois l'information que véhicule le document (des mots, des images, du son, par exemple), et des marques qui le structurent.

L'intérêt majeur de SGML est sa généricité, sa généralité et sa portabilité. En particulier, un système de codage des caractères permet leur lisibilité sur la plupart des ordinateurs.

Différence avec un traitement de texte classique

Les traitements de textes actuels utilisent aussi, pour le codage des documents qu'ils manipulent, des marques génériques (en association avec des feuilles de styles par exemple). Ils font preuve d'une double évolution depuis l'origine des documents électroniques, en ce sens que ces marques ne caractérisent pas la mise en forme directement (où l'auteur formate directement le document), ou ne font pas appel à une mise en forme procédurale (l'auteur introduit des marques qui seront interprétées par le périphérique d'impression). Ils vont plus loin en permettant l'utilisation de "macros commandes" d'édition, gérées par les marques introduites au sein du document, et suivant (fréquemment) la structure de ce document. Ces marques restent cependant spécifiques au traitement de texte employé (ou à un groupe de

traitements de textes), et ne peuvent être redéfinies. SGML va plus loin en proposant des marques génériques et redéfinissables, basées strictement sur la structure du document [Goldfarb 1990].

Un exemple de document SGML

Voici un exemple de document SGML très simple, donné en illustration dans [Marshall 1997] :

```
<!DOCTYPE mail SYSTEM "c:¥sgml¥dtd¥mail.dtd">
                    <mail>
                    <head>
                    <to>Club Secretary
                    <fr>Oldest Member
                    <sb>What About SGML?
                    <body>
                    <p>Dear Secretary
                    <p>P. G. Wodehouse dedicated The Heart of a
                    Goof <cite>To my daughter Leonora without
                    whose never-failing sympathy and
                    encouragement this book would have been
                    finished in half the time.</cite> Do you
                    think SGML would have done some good?
                    <p>Best regards
                    <p>Oldest Member
                    </mail>
```

Figure 8 : Document SGML

Les différentes parties constituant la structure de cette lettre sont "marquées" par des "balises" entre signes inférieur et supérieur (<balise>). On peut remarquer que seule la structure est indiquée par ces balises. L'apparence à donner à ce texte est indiquée par une "feuille de style" qui peut être définie selon les besoins, et qui associe à chaque type de balise une apparence. L'application de ces styles au document SGML donne par exemple ceci :

Mail: Oldest Member, What About SGML?

To: Club Secretary
From: Oldest Member
Subject: What About SGML?

Dear Secretary

P. G. Wodehouse dedicated The Heart of a Goof "*To my daughter Leonora without whose never-failing sympathy and encouragement this book would have been finished in half the time.*" Do you think SGML would have done some good?

Best regards

Oldest Member

Figure 9 : Document SGML mise en page

Dans ce cas, la feuille de style a par exemple associé à l'en-tête (balise <head>) la mise en forme en gras, et au corps du texte un retrait à droite. La DTD définissant le langage de description des balises de ce document est donnée ci-dessous :

```
<!--Golf Club's Mail DTD-->
                    <!ELEMENT mail - - (head,body)>
                    <!ELEMENT head - O ((to & fr) & sb?)>
                    <!ELEMENT body - O (p*)>
                    <!ELEMENT to - O (#PCDATA)>
                    <!ELEMENT fr - O (#PCDATA)>
                    <!ELEMENT sb - O (#PCDATA)>
                    <!ELEMENT p - O ((#PCDATA|cite)*)>
                    <!ELEMENT cite - - (#PCDATA)>
```

Figure 10 : Déclaration SGML

Chaque type de balise y est décrit par un formalisme de grammaires hors-contexte avec attributs. Les caractères "<!" introduisent une déclaration, comme celle d'une d'une balise. La deuxième ligne indique par exemple que la structure "mail", délimitée par les balises <mail> et </mail> comporte deux parties obligatoires successives, "head" et "body" qui sont à leur tour définies. Les deux tirets "- -" indiquent que les deux balises de début et de fin sont obligatoires (alors que celle de fin pour "head" peut être omise sans créer d'ambiguïté d'interprétation). On pourra se reporter à [Goldfarb 1990] pour plus d'informations sur la description des DTD.

Ainsi, chaque entreprise ou individu peut déclarer le type de document qu'il souhaite en construisant sa DTD : rapport, poèmes, affiche électorale, manuel, devis, fichier d'aide en ligne, etc.. Comme ils sont exprimés selon le formalisme SGML, toute application conçue pour SGML pourra lire le document, pourvu qu'on lui fournisse sa DTD.

DSSSL, CALS, OS, FOSI

La description SGML d'un document ne spécifiant que sa structure, les organismes qui utilisent SGML se basaient souvent sur des méthodes propriétaires pour définir les feuilles de style associées à leurs documents.

Pour homogénéiser aussi la description des feuilles de styles, un sous-groupe de l'ISO a adopté en 1996 la "Document Style Semantics and Specification Language" (DSSSL), qui est un métalangage de définition des feuilles de style. En suivant la DSSSL, il est donc possible de définir une (puissante) feuille de style associant un style à chaque balise.

Parallèlement, le département de la défense du gouvernement des Etats-Unis (DoD) a produit, dans le cadre du projet CALS, une version (plus avancée) de la DSSSL nommée "Output Specification" (OS), qui permet de produire des feuilles de styles nommées "Formatting Output Specification Instance" (FOSI).

Le comité ISO travaille à l'assimilation de OS comme un sous-ensemble de DSSSL. JADE, une implémentation de DSSSL réalisée par James Clark [Clark 1997], permet de produire par exemple des documents RTF ou TeX.

Text Encoding Initiative (TEI)

La Text Encoding Initiative est le résultat d'une réflexion de cinq ans menée par des spécialistes de l'édition et de nombreux chercheurs en "humanités computationnelles", qui a pour but de donner des *recommandations* pour l'encodage de données lisibles par la machine,

visant à l'échange de ces données. Les principes de la TEI vérifient en particulier les clauses suivantes [Burnard L. 1992] :

- fournir un format standard pour l'échange de données de recherches en "humanités"
- suggérer les principes pour le codage de textes du même format
- les recommandations doivent :
  ⇒ définir une syntaxe recommandée pour le format
  ⇒ définir un métalangage pour la description des schémas d'encodage de texte
  ⇒ décrire le nouveau format et des schémas représentatifs à la fois par le métalangage, et en prose.
- proposer des ensembles de conventions pour coder les nouveaux textes selon le format recommandé.
- la compatibilité avec les standards existants doit être maintenue, autant que faire se peut.
- encourager les entités ou organisations de financement à supporter la création d'outils d'échange.
- ne pas imposer l'inclusion d'autre information que celle qui est déjà codée dans les textes.

Pour résumer, les deux principaux buts de la TEI sont de montrer ce que l'on peut coder (quoi), et comment le coder (comment). Le premier résultat est une documentation de 1400 pages, précisant comment 400 entités textuelles doivent être codées (quoi), sous forme d'éléments et attributs SGML (comment). Celles-ci sont regroupées en "groupes de balises". Il existe plusieurs groupes dont un "cœur" qui comporte les balises les plus communes.

Plusieurs idées sont avancées pour donner des éléments les plus globaux possibles, telles que "le système des classes", ou que les mécanismes de spécification comme le "renommage". Pour notre propos nous retiendrons par exemple la définition de la notion d'unité de texte ou segment par deux balises :

- <s> et </s> qui permet la segmentation de haut niveau en unités textuelles, la phrase en général.
- <seg> et </seg> qui permet une segmentation plus fine, en groupes de mots ayant une logique linguistique par exemple.

Le détail des recommandations de la TEI n'est pas analysé ici. Un des points importants que nous soulignons est que ce groupe d'étude a opté pour une représentation de données et documents basée sur SGML.

Arbre de la famille SGML

Si SGML est générique, il n'en reste pas moins un choix restrictif en absolu (langage déclaratif basé sur un marquage structurel). Les documents tels que la TEI les décrit, tout en restant génériques, ont un degré de généricité moindre que SGML. Une restriction supplémentaire est imposée pour la partie DEI de TEI qui décrit les dictionnaires. Le schéma suivant représente une partie des instances de SGML. Le degré de généricité, qui décroît à chaque création d'instance, est indiqué par la présence d'un certain nombre de côtés du cadre qui entoure ces éléments. L'étoile indique que ce ne sont pas des familles de documents qui sont directement définies mais des recommandations engendrant ces familles.

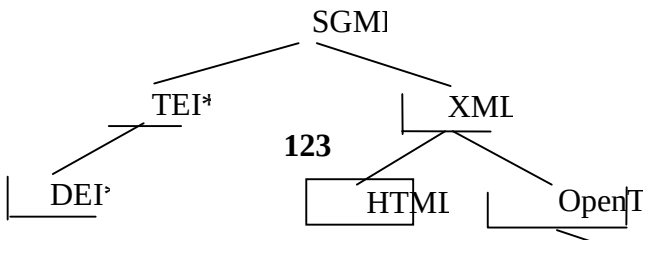


Figure 11 : La famille SGML et ses degrés de généricité

La partie droite de l'arbre représente des familles de documents, en ce sens que les DTD de ces documents sont restreintes par des "meta DTD". OpenTag est un peu ouvert (malgré son nom), alors que HTML et TMX sont fermés en ce sens que leur DTD est fixée.

Une évolution de SGML pour les applications Internet : XML

Introduction

XML est le résultat des efforts d'un groupe de travail du "World Wide Web Consortium" (W3C). Un des buts est de remédier au caractère figé de HTML, d'ailleurs lisible par les analyseurs XML, qui est actuellement résolu par exemple par l'utilisation d'applications JAVA [Bosak 1997], et de permettre comme dans les documents SGML une définition de leur structure par une grammaire intérieure aux documents.

Parmi les caractéristiques de XML, la conformité et l'utilisation de DSSSL en font une représentation intéressante pour le codage des documents contenant une mise en page importante. Ainsi l'auteur d'un document peut en définir sa structure avec une DTD (ou la spécialisation d'une DTD), et sa mise en forme à l'aide d'une feuille XSL de style.

Description sommaire des éléments de la grammaire XML

Un document XML est un ensemble d'unités de stockage que nous appellerons "éléments". Un élément est constitué de texte ou de données binaires. Un "texte" est une suite de caractères ; il peut représenter les données textuelles du document, ou un "marquage". Le marquage code la structure logique du document.

Nous classons les différents types d'éléments par leur syntaxe. En fait, cette syntaxe correspond souvent à une sémantique propre, comme on peut le vérifier dans la liste ci-dessous, sauf dans le cas des "balises" qui restent génériques. Voici les différents types d'éléments :

- données textuelles
  représentant réellement du texte (1)
  notationnelles :
  → références (2)
    - à des caractères : &#…; (2a)
    - à des entités : &…; (2b)
    - à des paramètres : %…; (2c)

  → balises (3)
    - de début : <balise> (3a)
    - de fin : </balise> (3b)
    - balises vides : <balise/> (3c)

  → commentaires : '<!--…….-->' (4)

  → sections de données CDATA : '<![CDATA[….]]> (5)

→ instructions pour procédures (7) dont la déclaration de type de document est un cas particulier (6) : <?XML version=”1.0”, encoding=”UTF-8” ?>

- des données binaires (8)

Pour trouver plus d'informations, et en particulier sur XML on peut se reporter à [XML 1997]. Pour une illustration de ces éléments, on peut se reporter à l'exemple de la Figure 13 qui montre un document TMX, spécialisation de XML.

1.2.2 Formats d'échanges de données textuelles

Introduction

La plupart des lignes qui suivent proviennent de l'analyse de [Thurmair 1997].

Office Document Architecture (ODA, 1989) / ISO 8613

ODA est une norme intéressante qui a été conçue pour un échange de documents multimédias tel que le destinataire ne soit pas obligé de connaître l'expéditeur. [Marcou 1994©] nous résume ses caractéristiques :

"La structure informatique de support dans ODA n'est pas la séquence de caractères, mais plutôt une structure orientée objet (sans les "méthodes"). La notion de balisage n'existe donc pas directement, mais ODA est comparable à un format utilisant des codes, en ce que les documents ODA ne sont pas directement lisibles par l'humain. ODA est donc, au premier abord, plus éloigné de l'univers traditionnel des documents que SGML .

Un document ODA est constitué d'une structure logique et d'une structure physique, ainsi que d'éléments de contenu partagés par les deux structures. Une feuille de style détermine la correspondance entre les deux structures. Les structures logique et physique sont les deux seules structures prévues dans la norme. Ceci rendrait donc a priori difficile l'utilisation de ODA pour implanter un type de structuration autre que la structuration logique, par exemple, la structuration par contenu ou selon la structure argumentationnelle (qui sont possibles avec SGML)."

Eurotra Document Interchange Format (EDIF, 1991)

Ce standard, fondé sur SGML et conçu lors du projet Eurotra, s'attache à coder essentiellement les phénomènes linguistiques (Morphèmes, mots,..). Il ne considère pas en particulier les problèmes de codage de la mise en forme des documents. Nous ne nous y attardons donc pas.

Simplified English and Correction of text (SECC, 1994)

C'est un format résultant des travaux du projet européen de ce même nom. Le projet utilise des marques pour qualifier les alternatives de correction. Il utilise aussi des structures de traits pour garder l'information relative à ces alternatives. Il ne concerne donc pas directement la prise en compte de la mise en page du document.

EURAMIS Pivot Format (EPF, 1997)

Ce projet a défini l'intégration de composants de traduction du service de traduction de la Communauté Européenne. Ce format se base sur HTML, et utilise Unicode. Il consiste en un marquage du texte de départ. Ce projet intéressant permet à la fois de coder les documents HTML eux-mêmes, mais aussi d'autres documents, comme les RTF, à l'aide d'un ensemble de marques particulières. Il définit les passerelles depuis et vers les formats communs.

Il comporte cependant toute sorte d'information spécifique au projet, et ne permet pas une interprétation facile par les systèmes extérieurs.

MULTEXT

Le but de ce projet est de développer à la fois un ensemble de marques utiles pour annoter les corpus de textes multilingues, et des outils relatifs aux manipulations de base de tels corpus. [MULTEXT 1998]. Ces notations se fondent sur SGML, les recommandations de la TEI, et les résultats des projets EAGLES [Ide & Véronis 1993, 1995].

Ce projet complet offre une notation englobant notamment les problèmes de représentation des caractères et glyphes des différentes langues, l'import et l'exportation de documents sous différents formats, et en particulier les notations linguistiques.

OpenTag (1997)
OpenTag est une instance de XML. Ce standard ouvert mais propriétaire (!) a été créé et est développé par la société ILE[1]. C'est un travail intéressant de développement d'un langage standard de description des documents textuels du commerce. Il doit permettre l'extraction des données de ces documents, leur traitement, et leur réinsertion. Plusieurs filtres sont actuellement en développement vers les principaux formats physiques du marché.

Les différents éléments XML spécifiques à OpenTag sont syntaxiquement classés comme suit (cf. spécifications en annexe ou à http://www.opentag.org/otspecs.htm) :

- texte brut
- références d'entités de caractères : &#x00E0 (“à” français codé sous Unicode)
  éléments à contenu (balises doubles) : <elem1 attr=”valeur”> contenu </elem1>
- éléments sans contenu (monobalises) : <elem2 attr=”valeur” />

Chaque balise peut contenir en son intérieur des “attributs”. Les exemples précédents montrent des attributs d'identification. Voici un exemple d'attribut spécifiant quelle est la langue à utiliser pour le contenu de la balise double :

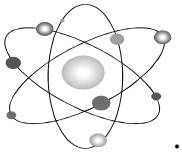
Exemple : <p lang[2]=”EN-EN”> ……

Sémantiquement, ILE classe ces éléments comme suit :

  instructions pour les traitements : <?XML version=”1.0”encoding=”iso-8859-1” ?>
  éléments structuraux et d'information : … , pour un paragraphe
  éléments en ligne : <fn/> : marque de note de bas de page, <g>…</g> : groupe
- éléments délimiteurs : <s> phrase </s>

Voici un exemple de document OpenTag, tiré de ses spécifications [OpenTag 1997], représentant respectivement les deux textes suivants :

&Activité

This is a text in 01 and ALL CAPS RED Second paragraph with graphic  .

_ ILE : Boulder, Colorado (http://www.ile.com/). Nous remercions ici M. Yves Savourel, de la société ILE, qui a répondu nos questions, ainsi qu'à tous les inscrits la liste de discussion sur Opentag. Merci aussi à M. Nicolai Puntikov de la société STAR, qui a contribué significativement cette thèse par des remarques précises

_ L'ancien attribut "lc" a été remplacé par "lang" dans la spécification de XML, fin 1997

```
<?XML VERSION="1.0" ?>
<OPENTAG version="1.0">
 <!-- First file, from a Java property file. It contains several locales. -->
 <FILE
  lc="EN-US"
        tool="Java_OTF:1.01-004:Java"
        dataattrib ="Java"
        original="//brazil/recife/devile/data/app.pro">
        <CSA lc="FR-FR" cs="iso-8859-1"/>
        <GRP rid="ID_DLG_STATUS" attrib ="label">
                <GRP id="IDC_ACTIVITY" coord="8;72;54;10">
                        <P lc="EN-US">&amp;Activity</P>
                        <!-- Tools specific data, e.g. in this case leverage information --
>
                        <P                                                    lc="FR-FR"
ts="100%,Gandalf3.tm">&amp;Activit&x00e9;</P>
                </GRP>
        </GRP>
        <!-- Example of a note generated by a filter -->
        <NOTE>Extraction word count = 1</NOTE>
 </FILE>
 <!-- Second file, this time from RTF. It contains only the text of the source language. --
>
 <FILE
        lc="EN-US"
        tool="Borneo 1.00-017"
        dataattrib ="RTF"
        original="//brazil/recife/devile/data/help.rtf">
        <!-- Definition for two user-defined characters. -->
        <CSDEF name="Latin1Cirth" base="ISO-8859-1">
                <NOTE>For more information about the Cirth see
                        the Web page http://www.indigo.ie/egt/standards/csur/cirth.html
                </NOTE>
                <MAP code="130" ucode="&#xE0D5;" ent="noldorian_o"/>
                <MAP code="#83" ucode="57558" ent="noldorian_oo"/>
        </CSDEF>
        <P id="1">This is a text in <G attrib ="01">01</G> and <G id="1">all caps
        red</G></P>
        <P id="2">Second paragraph with graphic <X id="1"/>.</P>
 </FILE>
</OPENTAG>
```

Figure 12 : Document OpenTag

Otelo Text Handling Interchange Format (OTHIF, 1996-98)

Les projets européens OTELO et ADVENTINUS visent à proposer un environnement commun aux outils de traduction, et les standards d'échange de données associés, au niveau des données (OTELO) et logiciel (ADVENTINUS). Le format d'échange de documents OTHIF est en particulier proposé (dans les premières versions, il s'appelait OTEXT).

On notera que OTHIF est compatible avec XML [OTELO 1998]. Ce format définit clairement (dans la mesure de son avancée, car le projet est en cours) une représentation du document par une DTD. Plusieurs aspects sont très intéressants comme par exemple la définition de segments à différents niveaux ("String, Phrase, Literal, Link, Font, Data). La gestion de la mise en forme est prise en compte.

1.2.3 Format d'échange entre systèmes de traduction

Metal Document Interchange Format (MDIF)
C'est un format d'échange qui permet de gérer les phrases, ou dans sa dernière extension les paragraphes. Il prend en compte toute une série d'éléments non-textuels comme les liens hypertextes, des polices ou des marques de titres. MDIF possède des filtres aller-retour vers les formats FrameMaker, Interleaf, ViewPoint, WordPerfect et ODIF [Kugler 1992].

Logos Exchange Format (LEF)
Ce format est proche de MDIF.

Une instance de XML pour l'échange de mémoires de traduction : TMX de LISA.
TMX est un format commun d'échange de mémoires de traduction proposé par l'organisation LISA (Localization Industry and Software Association) dont les membres sont représentatifs du milieu de l'édition et de la traduction de logiciels (environ quarante sociétés éditrices cinquante fournisseurs de services de traduction). Pour plus d'informations sur TMX, on pourra se référer au site Internet suivant (http://www.lisa.org/tmx/tmx.htm). Voici un exemple de mémoire au format TMX provenant de sa spécification:
Exemple :

Il s'agit ici du codage de deux unités de traduction (délimitée par les balises <TU> et </TU>) dont la première représente une phrase en deux langues : l'anglais et le français canadien, et la seconde un mot en trois langues : l'anglais, le français canadien et le français européen. Les unités sont précédées par un long en-tête.

```
<?XML VERSION="1.0" ENCODING="ISO-646" ?>
<TMX VERSION="1.0" >
  <HEADER
        CREATIONTOOL="XYZTool v1.01-023",
        DATATYPE="Text",
        SEGTYPE="sentence",
        O-TMF="ABCTransMem",
        CREATIONDATE="19970101T163812",
        CREATIONID="ThomasJ",
        CHANGEDATE="19970314T023401",
        CHANGEID="AlbertA",
        ADMINLANG="EN",
        SRCLANG="EN" >
        <NOTE>This is a note at document level.</NOTE>
        <META
              NAME="ExternalData",
              REF="data.txt" />
              <PROP NAME="RTFPreamble">{¥rtf1¥ansi¥tag etc...
              {¥fonttbl} </PROP>
```

```
            <UDE NAME="MacRoman">
                  <MAP UNICODE="#xF8FF" CODE="#xF0"
                   ENT="Apple_logo" SUBST="[Apple]"/>
            </UDE>
      </HEADER>
      <BODY>
            <TU
                  ID="0001",
                  DATATYPE="Text",
                  USAGECOUNTER="2",
                  LASTUSAGEDATE="19970314T023401" >
                  <NOTE>Text of a note at the TU level.</NOTE>
                  <PROP NAME="Domain">Computing</PROP>
                  <PROP NAME="Project">P&#x00E6;gasus</PROP>
                  <TUV
                        LANG="EN",
                        CREATIONDATE="19970212T153400",
                        CREATIONID="EP" >
                        <s>data (with a non-standard character:
&#xF8FF;).</s>
                  </TUV>
                  <TUV
                        LANG="FR-CA",
                        CREATIONDATE="19970309T021145",
                        CREATIONID="EP"
                        CHANGEDATE="19970314T023401",
                        CHANGEID="ManonL", >
                        <PROP NAME="Origin">MT</PROP>
                        <s>donn&#xE9;es (avec un charact&#x00E8;re
                        non standard: &#xF8FF;).</s>
                  </TUV>
            </TU>
            <TU ID="0002">
                  <PROP NAME="Domain">Activities</PROP>
                  <TUV LANG="EN">
                        <s>grocery</s>
                  </TUV>
                  <TUV LANG="FR-CA">
                        <s>dépanneur</s>
                  </TUV>
                  <TUV LANG="FR-FR">
                        <s>épicerie</s>
                  </TUV>
            </TU>
      </BODY>
    </TMX>
```

Figure 13 : Document TMX

Cela appelle quelques commentaires :

La spécification de TMX n'est pas encore tout à fait fixée, mais ses éléments principaux ont été choisis
La mise en forme n'est pas représenté. En effet, dans les segments compris entre la balises doubles <s> et </s> (les unités de mémoire), seul le texte nu figure.
Plusieurs champs sont prévus pour la gestion de ces unités de traduction : domaine, dates, auteurs, etc..

1.3 Formats Internes des Outils de Traduction Fondée sur la Mémoire de première génération

Les OTFM1g représentent en interne l'information provenant des fichiers qu'ils traitent sous un format particulier. Nous montrons les représentations de quatre OTFM1g : Translation Optimizer, Eurolang manager, Transit et Workbench.

1.3.1 Format de LANT Eurolang Optimizer : ELDIF

Eurolang Optimizer utilise un format interne appelé ELDIF. Avant de traiter un document, un filtre d'import traduit le fichier à traiter en ELDIF. Inversement, une fois les fichiers traduits sous Optimizer, un filtre retour permet de récupérer le fichier en RTF ou MIF (les deux formats traités par Optimizer). Le fichier de travail (de nom "folder.mnp", où m, n, et p sont des chiffres) d'Optimizer comporte plusieurs parties. Nous en présentons ci-dessous quelques-unes.

Ce format se voulait d'un certain niveau de compatibilité avec MDIF (voir plus haut) [Girard 1994]. ELDIF est basé sur SGML. Il existe en fait trois niveaux pour ELDIF :

- ELDIF-D : niveau primaire de communication avec l'exterieur
- ELDIF-S : niveau du segment
- ELDIF-L : niveau linguistique

L'échange avec les formats extérieurs était ciblé sur MIF (FrameMaker, RTF, Interleaf ASCII, et WordPerfect).

Les données d'analyse linguistique d'Optimizer sont stockées dans ce fichier comme par exemple les mots "fonctionnels" anglais représentés ci-dessous.

```
@(#) ;-) RTT_FUNCWORD_EN 001.0 94/11/04 (c) Eurolang Optimizer
a
about
above
according
across
after
afterwards
again
```

Figure 14 : mots fonctionnels dans la représentation interne d'Eurolang Optimizer

Les différents lemmes et leurs attributs y figurent codés de la façon suivante :

```
P/1/rtt/in/lemme.dat 10089
a N N13 N_p
```

```
a N N14 N_p
a N N15 N_p
loca N N16 N_p
qualia N N280 N_p
definientia N N35 N_p
mina N N240 N_p
ra N N170 N_p
era N N140 N_p
pora N N150 N_p
ossa N N160 N_p
```

Figure 15 : Codage des attributs des lemmes dans la représentation interne de Eurolang Optimizer

De même une liste du vocabulaire utilisé par le document à traduire, pour les différentes langues concernées est présent. Les deux figures qui suivent illustrent l'enregistrement des mots anglais et de leurs flexions. Les chiffres font référence à un codage interne. Seul un extrait est donné.

```
zy N N37 N_pD:¥214¥TMP/1/rtt/out/demœn.unk 806
access 2
administrator 2
appendix 2
are 3
ATT 10
available 1
```

Figure 16 : Extrait de l'enregistrement des mots anglais utilisés dans la représentation interne de Eurolang Optimizer

```
<DICTIONNAIRE_FLEXIONS>
-1 0 0 *
-1 1 0 *
r 1 1
st 1 2
less 2 0 *
lesser 2 1
least 2 2
far 3 0 *
farther 3 1
farthest 3 2
```

Figure 17 : Extrait du codage des lemmes en anglais dans la représentation interne de Eurolang Optimizer

La figure qui suit montre enfin le codage du document, avec éventuellement la correspondance avec les segments cibles (français ici) trouvés en mémoire.

```
<S id='D3P5U1S3'><F ls='D3P5U1F3.s' style='(CP=N);(WT=R);(AN=R);(UL=N);(VW=V);' family='Times New Roman'>Options which are not available are greyed<SEG id='D3P5U1E3' type='delim'>.<ESEG refid='D3P5U1E3'><ES refid='D3P5U1S3'> úúü___ò___12 ___ <S id='D3P5U1S1'> <F ls='D3P5U1F1.s'
```

style='(CP=N);(WT=R);(AN=R) ;(UL=N); (VW=V); ' family='Times New Roman'>Not all the users have access to the options displayed
<SEG id='D3P5U1E1' type='delim'>.<ESEG refid='D3P5U1E1'><ES refid='D3P5U1S1'>úúl___b___13___
<S id='D0P0U0S0'>Not all the users have access to the options displayed.<ES refid='D0P0U0S0'>úúØ___Î___14___<S id='D3P5U1S1'><F ls='D3P5U1F1.s' style='(CP=N);(WT=R);(AN=R);(UL=N);(VW=V);' family='Times New Roman'>Les options présentées ne sont pas disponibles pour tous les utilisateurs. <ES refid='D3P5U1S1'>úúô___ê___15___
<S id='D3P5U1S3'><F ls='D3P5U1F3.s' style='(CP=N);(WT=R);(AN=R); (UL=N);(VW=V) ; ' family='Times New Roman'>Options w.....

Figure 18 : Correspondance de segments dans la représentation interne de Eurolang Optimizer

Le codage est une instance de SGML. Toutes les marques RTF d'attributs ont été interprétées et codées sous le langage ELDIF (souligné).

1.3.2 Format interne d'IBM Translation Manager

Le format est basé sur une structuration des documents à traiter et de la mémoire inspirée de SGML. Les segments représentant les documents à traiter reprennent le code trouvé dans ces documents, comprenant les marques originales de structure et de mise en forme, mélangées avec les chaînes de caractères à traduire. La figure suivante illustre le codage sous Translation Manager 2 d'un document RTF.

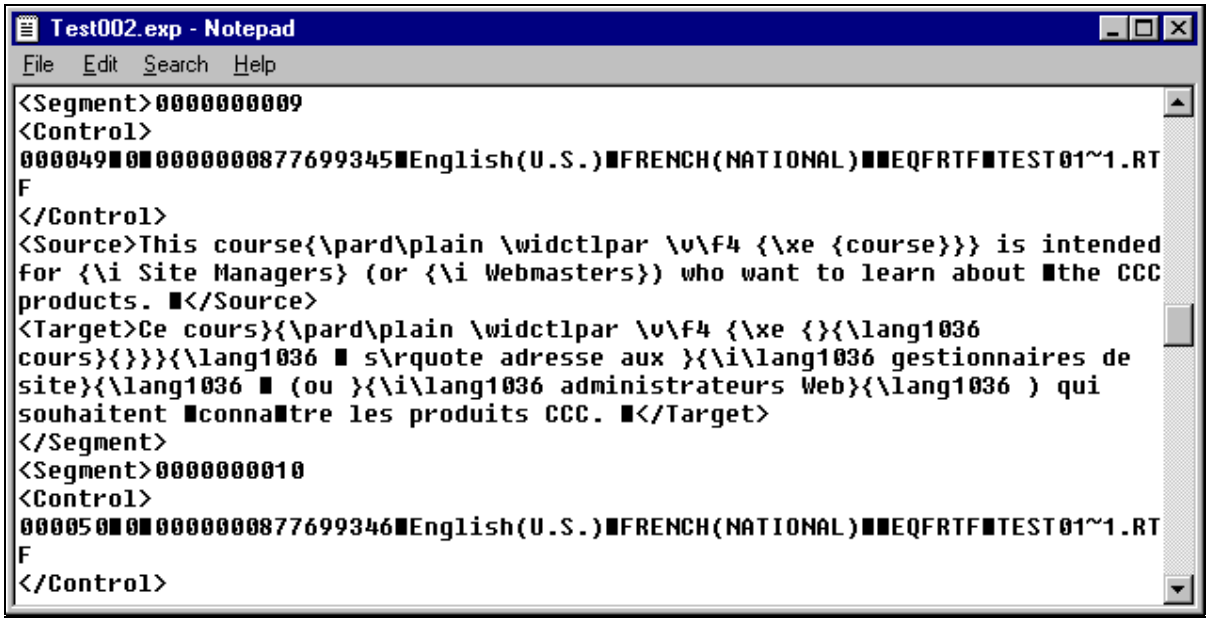


Figure 19 : Représentation d'un document par Translation Manager

Il ne s'agit donc pas exactement d'une interprétation des données, mais plutôt d'un "encapsulage" SGML. La mémoire d'unités de traduction sous la forme de bi-segments est composée d'un fichier du type montré ci-dessus (d'extension "tmx", plus un fichier index (d'extension "tmi").

1.3.3 Format de Star Transit

La représentation interne des documents traités par Transit fait preuve d'un degré de plus d'interprétation. Ce format est une représentation hybride dans laquelle figurent les textes à

traduire et certaines marques de formatage originales du document. Le document est divisé en segments de traduction délimités par des balises spécifiques, comportant un numéro d'ordre, et des références à des entités de formatages interprétés. Cette représentation est obtenue à partir de filtres aller. Des filtres retour permettent de revenir au format originel lorsque la traduction a été effectuée. La figure suivante montre un extrait la représentation par Transit du même document RTF que dans la figure précédente. Ce fichier est celui sur lequel ont porté les tests du chapitre 3.

|FMT - Text#-41
<{><¥fs24 >Course Overview-42
|FMT - Text#-43
Description-44
|FMT - Text#-45
This is a one-day, instructor-led course. -46
This course teaches students how to support the various features of AAA. -47
This course dœs not cover publishing Web page content.-48
|FMT - Text#-49
Audience-50
|FMT - Text#-51
This course is intended for Site Managers (or Webmasters) who want to learn about the CCC products. -52
In addition, it is intended for System Engineers who want to prepare for the forthcoming DDD exam.-53

Figure 20 : Extrait de la représentation Transit d'un fichier RTF

La mémoire de transit est composée de deux fichiers de ce type dont les segments se correspondent via leur numérotation.

1.3.4 Format de Trados Workbench

Workbench n'utilise pas de format de représentation propriétaire, mais se base plutôt sur la structure des fichiers RTF et la représentation de la mémoire à l'aide d'un réseau de neurones (fichier "ann") dont nous ne connaissons pas les détails. La forme d'échange de la mémoire est de type SGML (voir mémoire du test du chapitre 3).

Les mémoires de traduction de Workbench sont composées de quatre fichiers dont un fichier d'index basé sur une réseau de neurones, et un fichier de données (tmd). Les mots y sont classés sur la base de l'occurrence des lettres qui les constituent.

2. Choix du format primaire

2.1 Choix possibles

2.1.1 Contraintes

Notre but est de traduire des documents réels, c'est-à-dire mis en forme, comportant les types d'éléments dont on a dressé une liste (non exhaustive) précédemment. Or, une myriade de formats existe sur le marché, comme on vient de le voir. D'autre part, la *contrainte de compatibilité ascendante* que nous nous imposons nous oblige à choisir un format compatible avec la plupart des formats du marché, et des formats d'entrée des OTFM1g. Pour traiter les différentes caractéristiques évoquées plus haut, se pose donc le problème de choisir un type de représentation de document formaté assez général.

La plupart des OTFM1g se basent sur un langage de représentation interne hétérogène, contenant à la fois le texte lui-même et des marques de formatage. On peut se reporter au Chapitre 1, paragraphe E.1.2. pour plus de détails. Voici les contraintes que nous imposons au langage de représentation de documents que nous cherchons :

- il doit permettre de représenter la plupart des documents du marché actuel, via des filtres ou des interpréteurs.
- en particulier, la mise en forme et les objets imbriqués non textuels doivent obligatoirement pouvoir être représentés
- il doit être transmissible par Internet sans perte
- il doit être le plus simple possible dans sa structure
- il doit pouvoir présenter clairement la segmentation de la phrase en unités de traduction.

2.1.2 Un format physique

Choisir un format physique propriétaire nous réduit à ce type de format. On ne voit alors pas comment l'on pourrait représenter l'ensemble des documents que l'on veut traiter. Par exemple Word ne permet pas de gérer les scripts JAVA.

2.1.3 Un format interne de mémoire de traduction existant

Les formats de représentation interne des OTFM1g vus précédemment offrent une alternative intéressante. Ils sont en général basés sur une instance de SGML. Si cela est un progrès notable, cela n'est pas suffisant pour garder la généricité permettant de définir de nouvelles balises en fonction de nouveaux types de documents à traiter. Il est donc nécessaire de choisir non pas une entité physique, mais plutôt un format logique générique.

2.1.4 Un format logique

C'est la solution idéale permettant d'être assez strict pour définir des procédures, mais ouvrant d'autre part la possibilité d'incrémenter la description des éléments de documents représentés, en créant des éléments physiques sur le modèle d'objets logiques fixés. Le paragraphe suivant montre comment se baser sur XML, un format logique générique pour représenter l'ensemble des éléments listés plus haut, et un peu plus.

2.2 Choix de XML et degrés de liberté

2.2.1 Aspects génériques de XML

Les lignes qui précèdent montrent que SGML est largement accepté par la communauté industrielle et académique de l'édition comme un standard de spécification pour l'échange de documents (comme le recommande par exemple la TEI). La généricité de SGML permet de coder l'ensemble des éléments dont nous avons besoin. Cependant SGML définit une famille de documents extrêmement vaste.

Notre but est de nous servir d'une représentation assez générique pouvant contenir l'ensemble des éléments vus précédemment, tout en restant proche des documents couverts. XML offre cette possibilité en définissant en particulier clairement deux types de balises génériques qui semblent très adaptées à la représentation de mise en page et d'objets non linguistiques des documents vus plus haut : les balises à début et fin (que nous appellerons "balises doubles"), et les balises sans contenu (les "monobalises").

Les balises doubles peuvent marquer la zone de l'application d'une mise en forme, comme cela est fait par les marques des éditeurs courants.

Exemple :

Soit la phrase suivante de type RTF qui contient une marque de gras sur "mot en gras", l'utilisation de balises doubles XML pour spécifier ceci.

```
Voici un {/b mot en gras}.
```

```
<s> Voici un <hi attr=01> mot en gras </hi>.<s>
```

Il est à souligner que ce ne sont pas les balises de type <hi> qui déterminent le style gras du mot, mais plutôt l'application d'une "feuille de style" à ces balises, selon les recommandations de la DSSSL. L'instruction de la feuille de style ressemble à ceci.

```
(element <hi> (make font-style: bold))
```

Si ce mécanisme paraît a priori lourd, il offre en revanche une séparation du style et de la structure permettant de redéfinir à son aise (selon la langue par exemple) la mise en page, sans toucher au document logique lui-même.

Les monobalises permettent de représenter les objets non linguistiques non binaires comme les index, les hyperliens, ou les instructions pour programmes. Voici un exemple de représentation d'un index dans un document RTF, et sous XML.

```
Voici un mot {EX "mot"} indexé.
```

```
<s>Voici un mot <idx id=001 ref="mot"/> indexé.</s>
```

Voici encore quelques avantages de XML :

- XML peut être transmis sur Internet sans pertes (il a été conçu pour une utilisation via Internet)
- L'adoption par XML de la DSSSL doit permettre de gérer la mise en page ; une application, JADE [JADE 1998] et l' environnement de développement SAE [SAE

1988], montrent déjà qu'il est possible de générer des documents RTF à partir de documents XML et feuilles DSSSL par exemple.
- La structure des documents XML a été définie pour rester simple et lisible
- SGML offre un moyen simple de représenter une unité de traduction via la définition de balises dédiées aux unités de traduction. XML propose en particulier les couples <s> et </s> d'une part et <seg> et </seg> d'autre part.[3]
- OpenTag, une instance de XML décrite plus haut semble bien s'acquitter de son rôle de language pivot pour les formats du marché.

Nous choisissons donc XML comme type de représentation primaire.

Remarques :
- Il est nécessaire de définir des procédures de conversion aller retour entre le format du document à traiter et XML. Dans une application industrielle, cela demanderait donc de créer ces procédures. Quelques groupes y travaillent déjà tels ceux de OTELO pour OTHIF qui est un format, OpenTag qui sont des documents XML. En outre les travaux précédents d'échange pour les formats ELDIF (SGML) et Euramis Pivot Format (HTML) devraient pouvoir être utilisés en partie.
- Les attributs relatifs aux balises suivent les spécifications de XML. Nous ne donnerons pas une liste exhaustive de ces attributs, mais les introduirons plutôt selon les besoins. Il est clair que pour une application complète, cette liste d'attributs devra être spécifiée totalement.

2.2.2 Traitement des systèmes d'écriture

Un des avantages non encore exprimé de XML est la présence de mécanismes a priori pour gérer le codage des différentes langues ou transcriptions :

- La table des caractères utilisée par défaut dans la spécification est la table Unicode ISO/IEC 10646.
- lorsque le codage utilisé (un seul octet par exemple) ne permet pas d'afficher les caractères à deux octets d'Unicode, il est fait appel à des "références d'entités de caractères". Une référence d'entité de caractère est de la forme "&#n;" où "n" est un entier décimal, ou bien "&#xh;" où "h" est un entier hexadécimal. n et h font alors référence au code du caractère dans la table ISO/IEC 10646.
- Un attribut de langue ("xml:lang") est disponible pour les balises. Il consiste à introduire un code de langue ISO639, plus éventuellement un sous-code de pays de la liste ISO3166.

Exemples :
- Le premier caractère Hangul (coréen) a le code “44032” en décimal, cela donne “AC00” en hexadécimal, et il peut donc être codé soit 가 soit "가". Il est donc possible de coder un texte coréen avec un système d'exploitation peu performant au niveau de la représentation des langues comme Windows 95 version américaine.
- Dans le paragraphe suivant, l'attribut de langue indique que la langue est du catalan :

[3] On notera cependant que cela ne définit pas un mécanisme de segmentation.

```
<p xml:lang="ca">Si vols ben parlar, aprn a callar.</p>
```

2.2.3 Codage des bi-segments

Il s'agit d'une mémoire de bi-segments de type OTFM1g qui contient un segment source et un segment cible liés. Les segments ne sont pas alignés au niveau des mots, mais seulement au niveau macroscopique des segments eux-mêmes. Généralement cela correspond à un alignement au niveau des phrases.

Plusieurs formats d'échange de données linguistiques suivent SGML. On se reportera en particulier à la spécification de dictionnaires suivant la DTEI. Le format TMX du groupe de travail OSCAR (Open Standards for Container/Content Allowing Reuse) propose une représentation des mémoires de traduction adaptée à leur échange, particulièrement via Internet. TMX est une instance de XML. Outre sa conformité à XML dont on vient de montrer les avantages, voici encore trois points en sa faveur.

- TMX procède d'une réflexion commune à un groupe représentatif d'éditeurs et de sociétés traductrices.
- TMX est un standard ouvert
- TMX pourrait s'imposer comme un standard, permettant l'échange avec les mémoires provenant des OTFM1g principaux

Le formalisme TMX est cependant trop simple (il est fait pour être simple, c'est un choix !) pour pouvoir garder l'information provenant de l'analyse linguistique du segment (des arbres d'analyse par exemple). Garder cette information ne sert pas à grand chose entre logiciels qui ne peuvent s'en servir. En ce sens, la cohérence avec les OTFM1g est gardée.

Il peut cependant être intéressant de pouvoir conserver cette structure dans le cas où, par exemple, une partie de l'analyse est faite sur un serveur distant. **Nous considérerons que la représentation des mémoires de bi-segments se fait sous XML**. Il sera en particulier montré plus loin comment, sous une instance de XML, garder ces informations d'analyse linguistique.

2.2.4 Les documents dans leur ensemble

Nous ne traiterons par la suite que les segments XML, considérés comme des unités de traduction. On suppose que le format XML permet de représenter l'ensemble d'un document, et la cohérence des différents segments du document entre eux.

Il serait intéressant d'étudier plus profondément cette question, au niveau de la puissance de représentation des différents éléments d'un document : comment gérer les légendes d'un schéma, comme gérer le chevauchement visuel de deux paragraphes ou images, etc.

L'intérêt touche aussi à la cohérence de la traduction puisqu'il est a priori important de gérer les anaphores par exemple d'une phrase à l'autre. Or il y a de fortes chances pour que le référent d'un pronom se trouve dans une segment différent du segment courant.

Nous reportons cette question à une étude ultérieure, sans en oublier l'importance.

2.3 Exemple de segment XML

La phrase suivante est montrée telle qu'elle apparaît dans un document formaté (seul l'index n'est pas visible). Une version SGML est présentée plus bas :

Press the ***button{EX "button"}*** in order to get some **help**.

<s>Press the **<hi1><hi2>**button **</hi2></hi1><mobj attrib =”index” rid=”index1”/>** in order to get **<hi3>**some **<hi1>** help **</hi1></hi3>.</s>**

Figure 21 : Transcription d'un segment de document RTF en protoXML

La mise en forme d'un document (le gras par exemple) est matérialisée non pas par un fichier XML, mais par une feuille de style XSL appliquée à ce fichier XML. Il existe sous XML une façon d'exprimer cela sur la structure par les balises <hi> et </hi> (highlighted). Pour différencier les différents types de mise en valeur, nous utiliserons des balises numérotées. Dans l'exemple ci-dessous la balise de type <hi1> code des parties apparaissant dans le fichier RTF en gras, <hi2> en italiques, et <hi3> soulignées.

Les objets non-textuels seront représentés par une monobalise du type <mobj/>, et dont l'intitulé exact dépend de l'objet. Pour plus de clarté, un intitulé rappelant le nom de l'objet sera généralement employé. Par exemple pour les index, la balise sera <mobj attrib ="index"/>.

3. Choix du type de structure logique pour la représentation interne des données

Introduction

La première partie de cette étude a suggéré que l'information contenue dans un segment de document est multidimensionnelle. Nous explorons maintenant quelques possibilités de représentation de cette information multidimensionnelle. Parmi les dimensions fondamentales, on peut distinguer :

- le texte formé de caractères
- la mise en forme de ce texte
- les objets non textuels (champs, données, variables, etc.)

Notre but sera donc de trouver une représentation qui permette de stocker ces informations, d'exprimer les liens entre elles, et de les manipuler.

3.1 Exploration des différentes structures possibles

3.1.1 Représentation basée sur les ensembles

Un ensemble est une collection d'éléments groupés entre eux. Considérons la représentation du texte. Un texte peut être vu comme une suite de caractères. Pour représenter un texte comme un ensemble d'éléments que sont les caractères, il faut déterminer tout d'abord un moyen d'exprimer la succession de ces éléments. Cela nous oblige par exemple à définir un ensemble T dont les éléments sont des couples dont une partie représente le caractère lui-même, et l'autre partie sa position.

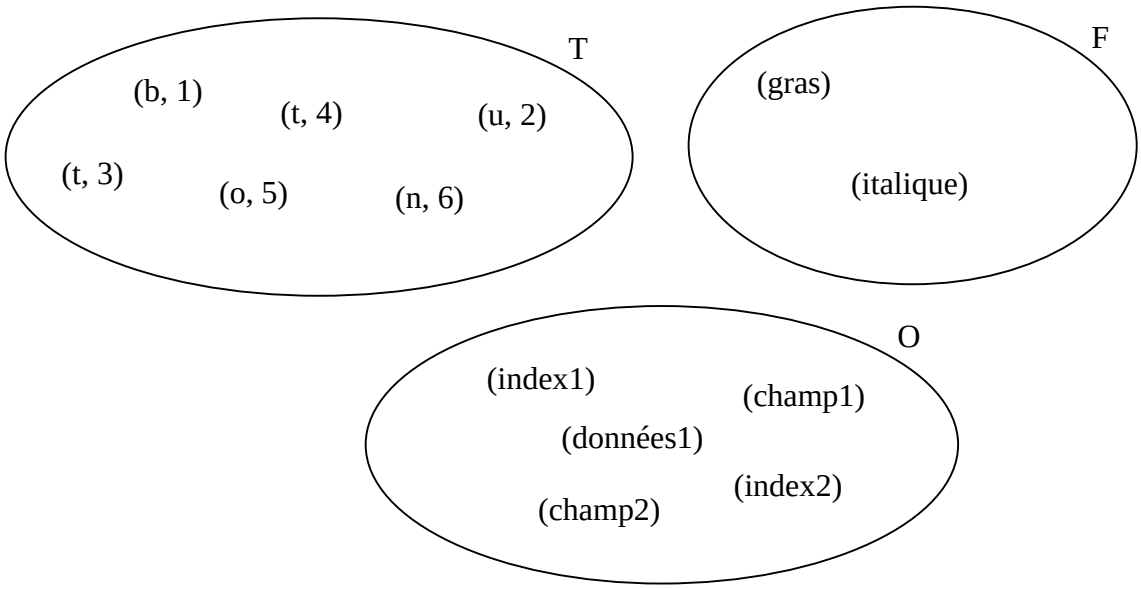


Figure 22 : Des ensembles pour représenter les données de la TFM

De même des ensembles pour la mise en forme et les objets non textuels pourraient être définis. Un premier problème se pose pour la représentation de sous-ensembles d'éléments : l'élément "données1" de l'ensemble des objets O doit pouvoir contenir l'information que portent ces données. Cela peut être un ensemble de bits pour une image, ou un ensemble de caractères. Il faudrait alors ne plus voir les éléments comme simples mais comme des

ensembles eux-mêmes. Les éléments devraient alors être vus eux-mêmes comme des objets dont la nature pourrait varier. On se rapproche alors d'une structure de base de données objets.

Les liens entre ces trois ensembles, et entre les éléments des ensembles pourraient être vus comme des relations d'ensembles. On peut penser par exemple à la relation "mise en forme" maf qui à un élément caractère de l'ensemble T associe un élément de F.

maf : C → F : u → gras

Mais cette représentation ensembliste dénature la séquentialité du flot de caractères du segment rendant la représentation difficile. La notion d'élément précédent ou suivant (d'un caractère ou d'un mot par exemple) est alors difficile à représenter au niveau de la structure (des ensembles) elle-même, et sera plutôt représentée par les relations.

3.1.2 Représentation basée sur les listes

Il semble donc important que la représentation possède une notion d'ordre a priori. Les structures les plus simples contenant cette notion sont les listes. Soit donc une liste par "dimension", par exemple une liste pour les caractères du texte, une liste pour la mise en forme, et une pour les objets non linguistiques.

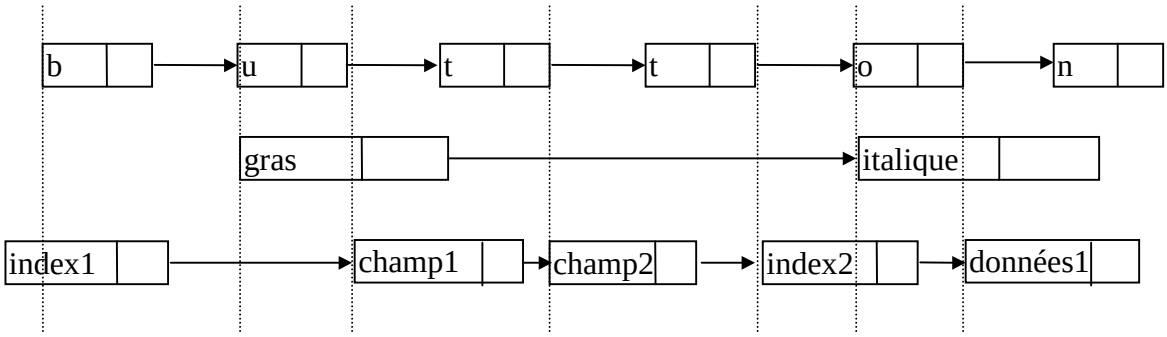


Figure 23 : Des listes pour représenter les données d'un document

La séquentialité des éléments (caractères, objets non linéaires) est alors bien représentée et peut être gérée de façon classique par ces listes. De plus il est alors possible de créer une correspondance géométrique entre les différentes dimensions de façon naturelle (matérialisée sur le schéma par des traits verticaux en pointillés).

Considérons maintenant que l'on veuille représenter plusieurs possibilités de séquences de caractères. En ne reprenant que la partie qui représente ces caractères, il est possible de faire ceci en proposant deux listes contiguës de la façon suivante (le troisième élément de la seconde est un "r", au lieu d'un "t" :

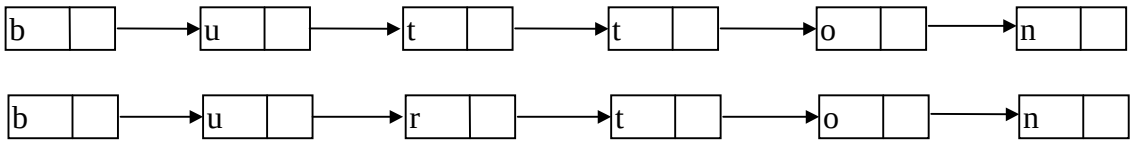


Figure 24 : Deux possibilités pour la liste des caractères

On voit alors que l'information est quasiment la même, à part un des nœuds et que cette structure n'est pas factorisante, ce qui pose des problèmes d'efficacité de stockage de l'information.

3.1.3 Représentation basée sur les treillis

Reprenant le problème précédant, il existe une solution de factorisation représentée par le schéma ci-dessous :

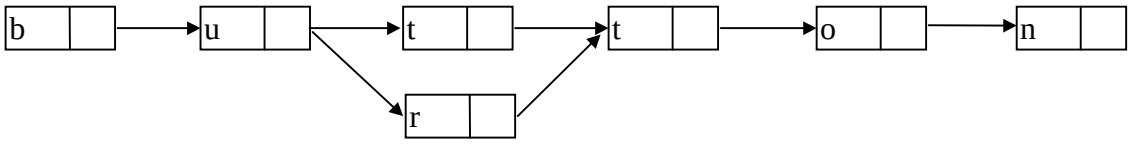


Figure 25 : Factorisation de l'information en un treillis

La structure relative aux caractères devient alors un treillis. Dans un treillis, il n'existe que des arcs explicites, les cycles sont interdits, et il doit y avoir un unique nœud inférieur à tous les autres, et un unique nœud supérieur à tous les autres [Boitet 1988]. La notion d'ordre partiel sur chacun des chemins est conservée et l'alignement géométrique des éléments d'un treillis à l'autre aussi, comme pour les listes précédentes. Les arcs peuvent porter des valuations pour permettre une approche statistique ou heuristique.

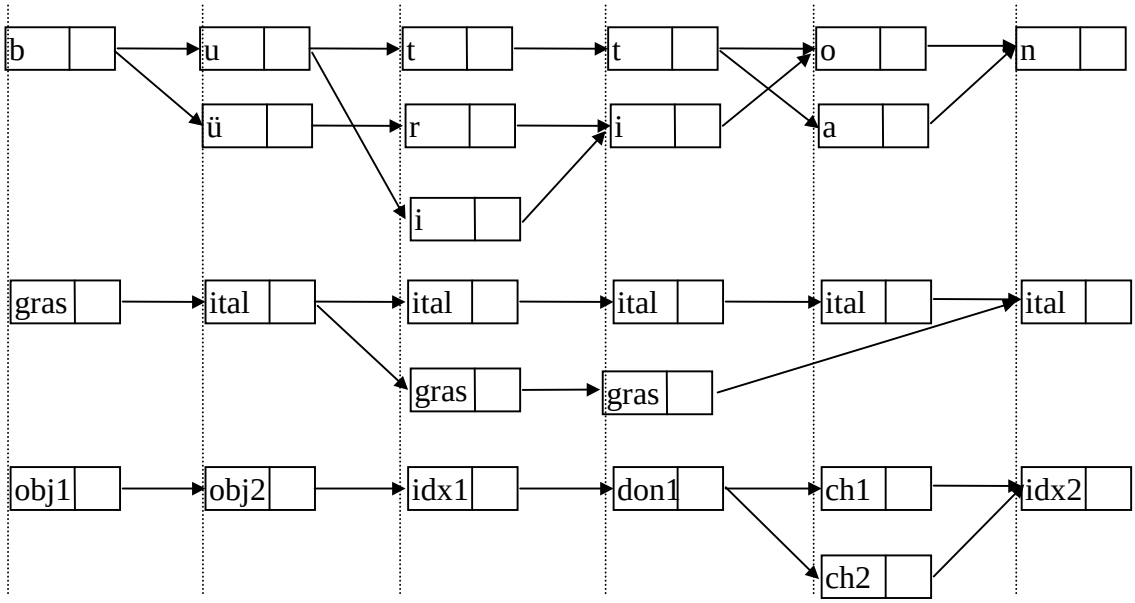


Figure 26 : un ensemble de treillis

3.1.4 Une alternative : les cartes

Les cartes, utilisées dans le système MIND [Kay 1973] par Martin Kay [Lafourcade 1994], offrent une puissance d'expression équivalente, mais ce sont les arcs, et pas les nœuds qui portent l'information (étiquettes et valuations).

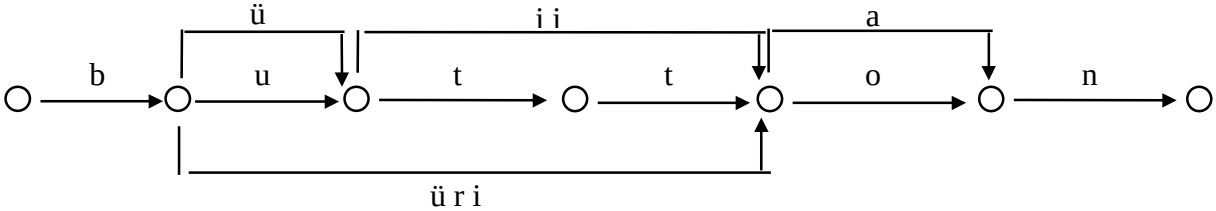


Figure 27 : Premier treillis de la figure précédente sous forme de carte

3.1.5 Un mot sur les arbres et les réseaux

Les arbres, très utilisés en traitement informatisé du langage naturel, ne permettent pas la séquentialité jugée nécessaire plus haut. Les réseaux, intéressants par leur puissance

d'expression, qui englobe les treillis, autorisent des cycles, qui augmentent la complexité des algorithmes qui y sont appliqués.

3.2 Choix de la structure interne

3.2.1 Structure logique

Nous optons donc pour représenter la partie analytique de notre représentation par une structure formée d'un **ensemble de treillis**. Cette structure, que nous qualifierons "d'étagée", permet à la fois de séparer les dimensions des différentes données présentes dans un segment à traduire, mais aussi de garder la séquence naturelle à un flot de données documentaires des éléments du treillis. Un treillis par étage permet aussi de générer autant d'alternatives que l'on veut au niveau de l'arrangement des éléments d'un même type (les caractères par exemple).

La structure de représentation adoptée pour représenter une unité de traduction comporte donc deux grandes parties : un segment au format SGML, et un ensemble de treillis. Le chapitre suivant montrera plus pratiquement comme appliquer et lier ces deux parties.

```
<s> <gras> b <gras/> <obj/> <italique> u <obj/> <> t <idx/> t <don/> o <ch/>n <idx/>
</italique></s>
```

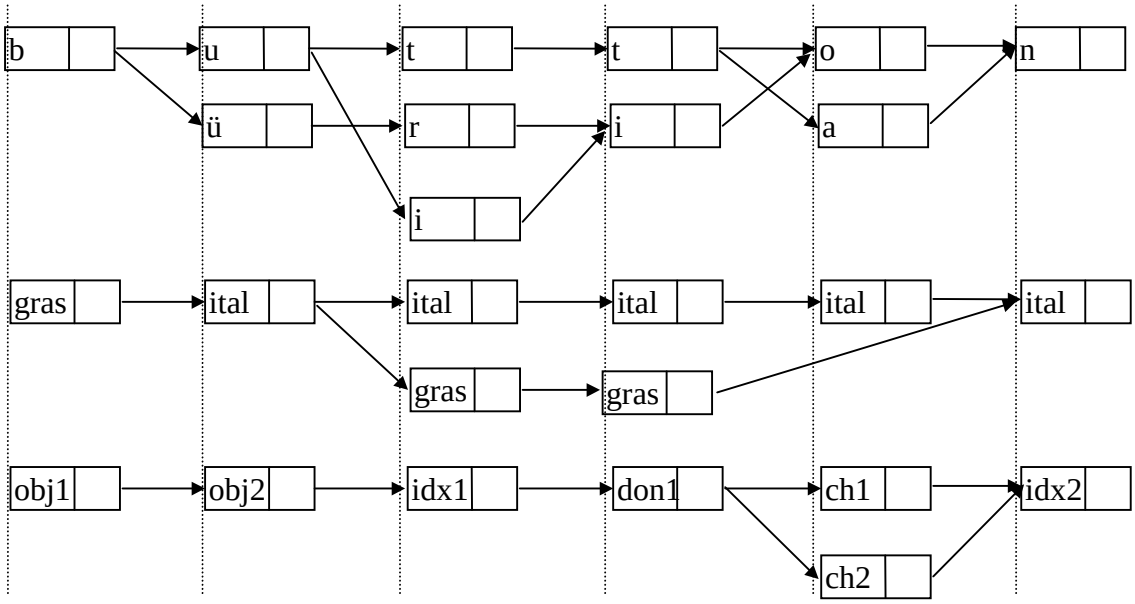


Figure 28 : La Structure Interne de Représentation des Données comportera deux parties : un segment SGML, et un ensemble de treillis

Dans la figure ci-dessus, les treillis représentent les données du segment SGML, mais aussi le résultat d'une analyse imaginaire qui donnerait de nouveaux chemins.

3.2.2 Liaisons entre nœuds et relation de correspondance

Les nœuds ne sont pas indépendants entre-eux. Ils vérifient des correspondances avec d'autres nœuds provenant de leur sémantique et de celle des nœuds auxquels ils sont liés. Par exemple les nœuds "t" du premier treillis des caractères de l'ensemble précédent sont liés avec les nœuds "gras" du treillis des mises en forme car le caractère les "t" sont en italique dans le texte original. Si dans ce cas le lien se fait naturellement de part la séquentialité des éléments, on pourrait imaginer que "b" est aussi en gras, et que c'est une des balises existantes qui le

spécifie. Pour exprimer ces relations entre nœuds, nous définissons la notion de *liaison* comme la relation d'un ensemble de nœuds à un autre.

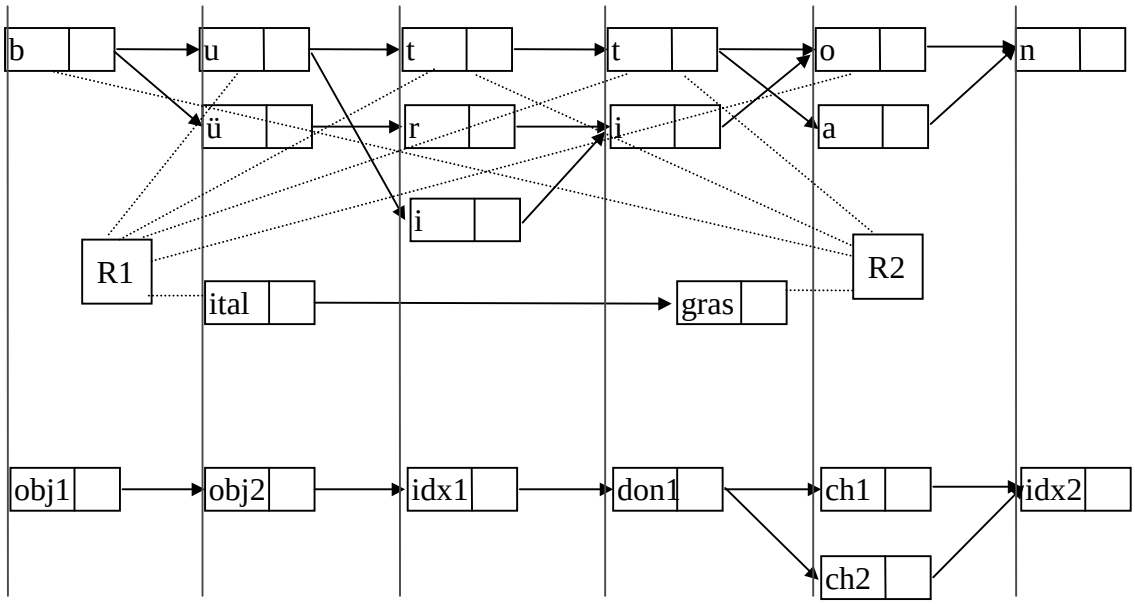


Figure 29 : Factorisation de nœuds de mise en forme

Le schéma ci-dessus montre comment utiliser ces relations pour factoriser les nœuds de mise en forme. Il n'existe plus qu'un seul nœud par type de mise en forme. Autant de liens relient le nœud de mise en forme concerné aux nœuds des caractères concernés, via des nœuds de liaison. Ces liaisons constituent donc une correspondance entre les différents nœuds des treillis. Nous appellerons un tel ensemble de liaisons : *relation de correspondance*.

3.3 Structure en étages et format d'échange

3.3.1 Structures internes des outils de traduction fondée sur la mémoire

En résumé, la représentation d'une unité de traduction dans la structure de représentation interne comprend donc deux parties :

- une forme primaire sous forme d'un segment XML
- une structure étagée constituée d'ensemble de treillis liés par une relation de correspondance

3.3.2 Langage de description des structures étagées

La première partie de la structure a déjà été définie pour être échangeable, en particulier sous Internet. Nous définirons au chapitre 6 un *langage de description* de la deuxième partie respectant les conventions XML, en proposant les nouvelles balises et conventions nécessaires.

La structure dans son ensemble prend alors un aspect d'auto-définition intéressant puisque l'analyse du segment XML peut à son tour être représentée comme un fichier XML.

3.3.3 Puissance de représentation

Introduction

La discussion précédente s'est inscrite dans le cadre classique où n'apparaissent qu'une langue source, et une langue cible. La représentation précédente est générique et permet sous le

même formalisme de représenter autant de langues sources ou cible désirés. Deux situations sont maintenant proposées.

Représentation d'une langue source vers plusieurs langues cibles
Il est très courant de trouver une configuration de traduction dans laquelle l'on part d'une langue source vers plusieurs langues cibles. On peut penser aux manuels de maintenance des avions Dassault dont la langue source est le français, aux manuels de voitures de Toyota dont la source est le japonais, ou encore aux manuels de référence d'Apple dont la langue source est l'anglais. Ce type de situation (1-n) demande alors de mettre en correspondance une structure source avec autant de structures cibles que de langues vers lesquelles on traduit. Dans ce cas, à un segment XML source et un ensemble de treillis sources, correspondent autant de segments et ensemble de treillis cibles.

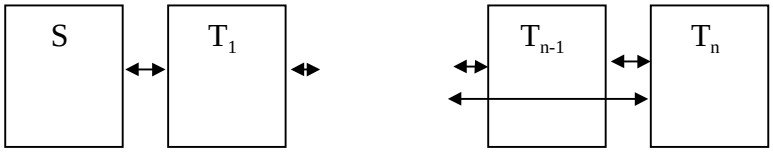


Figure 30 : Une structure TELA source vers n structures TELA cibles

Possibilité de représentation d'une interlingua
Une des structures pourrait être une structure abstraite de référence, basée sur une interlingua par exemple. Le type de configuration (1-n) permettrait alors le lien entre cette structure de base, et les segments et ensembles de treillis des langues cibles.

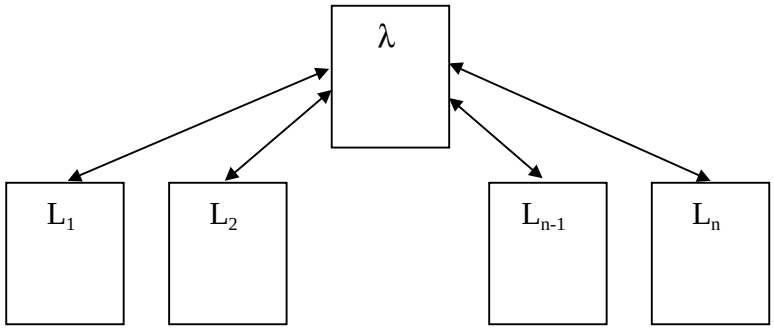


Figure 31 : Une structure TELA pivot vers n structures TELA cibles

Dans ce cas, se poserait alors le problème de posséder une représentation du dictionnaire adéquate. On peut alors penser à un dictionnaire à acceptions [Sérasset 1994], ou un dictionnaire basé sur les concepts, comme dans le projet UNL.

Conclusion

Après avoir passé en revue les types de documents présentés en entrée aux OTFM, les formats de représentation physiques utilisés actuellement, et les formats logiques, une forme de représentation primaire (non analysée) a été choisie vérifiant les *spécifications XML*. De même les possibilités pour représenter la forme analysée de l'unité de traduction ont été explorées. Le choix a porté sur une *ensemble étagé de treillis liés*. Il est ainsi proposé de considérer la S*tructure Interne de Représentation des Données* d'une *unité de traduction* comme la donnée d'un segment XML et un ensemble étagé de treillis liés.

Chapitre 5

Définition empirique

Contenu du chapitre

Introduction

Ce chapitre détaille les étages de la partie analytique de la Structure Interne de Représentation de Données (SIRD) pour les systèmes de Traduction Fondés sur la Mémoire. La démarche s'effectue volontairement du "bas" vers le "haut" pour vérifier la bonne adaptabilité des structures de treillis à la représentation des données nécessaires à la Traduction Fondée sur la Mémoire. Les fonctionnalités y sont décrites pour chaque étage, engendrant la structure des nœuds des treillis, et des liens entre ces nœuds.

Cette représentation interne sera la même pour manipuler lors du processus de traduction, à la fois les segments de documents d'entrée à traduire, les bi-segments des mémoires utilisées pour la traduction, et le segment traduit final. Les fonctionnalités abordées ci-après s'appliqueront donc à ces trois types d'information, qui ont la même nature.

Nous posons ci-après quelques choix pour la représentation des *unités de traduction*. Puis les paragraphes II à IX décrivent chacun l'un des étages. Le paragraphe X donne une vue globale de la structure en guise de conclusion.

1. Rappels de choix

1.1 Segmentation en unités de texte

Nous définissons ici la représentation d'unités de traductions selon le formalisme XML. Il y a plusieurs choix possibles pour représenter les unités de traduction, comme à l'aide des balises et ou <s> et </s>.

Définition 1 : Unité de texte (segment)

Par convention, une unité de traduction (ou "segment") est le texte contenu entre le couple de balises XML : <s> et </s>.

Une unité de traduction**{**xe "unité de traduction"**}**, (un segment) correspond dans la plupart des cas, à une phrase. Ce choix nous permet de remédier dès maintenant aux problèmes de segmentation rencontrés dans la première partie. En effet, cette segmentation ne se base plus sur les marques de ponctuation**{**xe "ponctuation"**}** et les balises d'objet (comme dans les OTFM1g). Elle se fonde plutôt sur l'occurrence de deux balises spécifiques.

Pour être plus précis, soit le texte a été rédigé sous un éditeur XML, et cette segmentation est effectivement disponible, soit le document XML que l'on traite provient d'un filtrage depuis un document du marché (comme les procédures construites pour l'échange avec OpenTag). Dans ce cas le problème de la segmentation en unités de texte demande de vrais choix au moment du filtrage.

Notre position n'est donc pas de déclarer que dans tout document XML les balises <s> et </s> définissent des unités de traduction, mais plutôt de considérer comme une convention que dans la suite, toute unité de traduction sera codée entre les balises <s> et</s>. Cela est aussi valable pour les mémoires de bi-segments.

Remarques :

- L'unité de traduction pourrait être différente pour la suite de nos traitements sans modifier le raisonnement qui suit. Cela pourrait être par exemple le paragraphe XML <p> ou le sous-segment XML <seg>. Dans ce cas il faudrait que la définition des segments de mémoire de traduction soit la même pour que les correspondances entre les segments à traduire et les segments de la mémoire se fassent.
- Le choix d'une unité de segment balisée par <s> et </s> est arbitraire. Le but étant de définir un cadre de réflexion pour la suite. La segmentation du texte en unités de traduction (segments) peut cependant être un problème difficile qui s'approche de celui de la définition des phrases (voir [Véronis 1997]) En effet décider si un point dans un texte anglais termine le segment, fait partie d'un abréviation ou d'un chiffre à virgules peut ne pas être évident. Il est supposé pour la suite que la segmentation a préalablement été effectuée soit par une procédure, soit qu'elle est donnée a priori comme dans le cas de fichiers de mémoires de traduction au format TMX.

1.2 Les éléments XML

Nous groupons ces éléments par leur comportement syntaxique en trois classes dans lesquelles sont rangés les huit types de base des éléments XML (cf. chapitre 4).

- classe 1 : les caractères de texte et références d'entités
- classe 2 : les balises "doubles"
- classe 3 : les monobalises

Un document XML est donc perçu comme une suite d'éléments de ces trois classes. Les balises doubles comportent obligatoirement deux parties entourant d'autres éléments du type <balise> </balises>. Les monobalises ne comportent qu'une partie. La forme générale d'un tel segment ressemble donc à ceci :

<balise double début 1> texte1 <mobj1/> texte2 texte3 &référence; <balise double début 2> Texte4 </balise double fin 2> </balise double fin 1>.

2. Étage 0 : forme primaire

2.1 Opérations

C'est la représentation de base à partir de laquelle sont construits les étages suivants. Elle contient l'ensemble de l'information de façon brute. Au niveau des procédés de traduction, aucune opération n'est a priori basée sur ce niveau de représentation. Cette représentation du document est la forme primaire définie au chapitre précédent.

2.2 Description

2.2.1 Généralités

De part sa spécification, XML permet de représenter les documents électroniques, y compris leur formatage, et les objets non-textuels qu'ils possèdent (dans la limite des formats et objets pris en considération par XML, et de l'habileté des transcodeurs à convertir ces documents sources de formats divers vers et depuis XML). Voici une phrase (représentée ici en Word 7), et sa représentation sous forme de segment XML :

Exemple :

This course { XE "Web" ¥t "course" }dœs *not* cover publishing **Web**{ XE "Web" ¥t "See Web Pages" } page content.

<s>This _ course_ **<mobj rid="index1"/>** dœs_ **<hi2>** not **</hi2>** _ cover _ publishing **<hi1>** Web **</hi1>** **<mobj rid="Index2"/>**_ page content_**</s>**

Les "marques" du document RTF et "balises" du segment XML sont indiquées en gras. Elles peuvent être visibles ou pas dans le document d'origine. Dans un document Word par exemple, la marque indiquant une note de pied de page est partiellement visible, et la marque d'index ne l'est pas (ici elle est visible, soulignée en pointillés pour des raisons explicatives).

2.2.2 Codage et transcription

Dans la suite nous considérons que les chaînes de caractères représentant les mots et les caractères de la langue du texte traité suivent un codage, comme par exemple ASCII 7bits ou 8bits, Unicode, ou Shift-JIS (un codage pour le japonais). Le codage est l'ensemble des caractères de base qui permettent de composer le document. D'autre part, pour interpréter un de ces caractères, une transcription est utilisée. La notion de transcription est expliquée ci-dessous :

Exemple :

- Ch. Boitet (qu'il nous pardonne cet exemple) a pour habitude d'adopter un codage post-fixé des diacritiques du français quand il s'adresse par courrier électronique en français à un interlocuteur dont la passerelle risque d'être en ASCII 7 bits :

"Cher Monsieur,
vous trouverez ci-joint l'intitule' de la the`se de Monsieur ..."

- Voici un exemple de transcription des caractères chinois par leur phonétique, un chiffre associé à leur ton, et deux lettres discriminatoires (il existe des phonèmes et leur ton associé qui possèdent plusieurs - dizaines de - caractères correspondants).

"TA1ab HUE4aa SHUO1bc ZHONG1ak GUO4ad HUA2am"[1]

- Voici maintenant un exemple plus connu sur Internet, de texte représenté par un segment provenant d'une mémoire TMX avec la convention "Byte Order Mark" d'Unicode (indiqués en gras dans l'exemple ci-dessous). Le codage est en ASCII 7 bits, et sert à la transcription de caractères Unicode (s'ils sont différents des caractères ASCII 7 bits).

données (avec un caractère non standard: é)

<**s**>donn**é**es_(avec_un_charact**è**re_non_standard:_**é**)_<**/s**>[2]

La forme d'échange du document possède donc un codage et une transcription qui devront être précisés.

2.2.3 Double numérotation des éléments du segment

Rang00

Un segment XML peut être vu comme un suite de caractères. Le"rang00" du caractère est son numéro d'occurrence dans le segment XML à partir du premier caractère après le caractère ">" de la balise "<s>". Le dernier caractère du segment est celui qui précède le caractère "<" de la balise de fin de segment </s>.

Rang01

Un texte XML est aussi une succession de trois types d'éléments : les textes et références, les balises doubles, et les monobalises. Ces éléments sont soit séparés par des blancs (notés "_" ici), soit par des "<" ou ">" qui commencent ou finissent les balises. Nous adopterons une numérotation de ces éléments qui est simplement leur numéro d'occurrence dans le segment, et l'appelons "rang01". Ainsi dans l'exemple de la Figure 1, "**<s>"** a pour rang01**{**xe "rang01"**}** "0", et "not" a pour rang01 "6". Les espaces visibles dans les segments ne sont pas à prendre en considération puisqu'un espace est noté par un souligné "_".

2.3 Calcul

2.3.1 Généralités

Cette représentation se basant sur des textes décrits sous le formalisme XML, aucun calcul autre que l'extraction du segment hors du document n'est nécessaire.

On note que dans un système complet, toutefois, il est nécessaire d'inclure des convertisseurs depuis (respectivement vers) les formats des documents d'origine comme RTF, FrameMaker, Interleaf ou Ichitaroo (Japon) vers (respectivement depuis) XML. Dans le cadre d'OpenTag par exemple, ces convertisseurs sont définis, implémentés et mis à jour soit par les développeurs de filtres OpenTag qui sont soit les initiateurs (société ILE), soit par les éditeurs de logiciels qui souhaitent la compatibilité avec OpenTag. Il ne s'agit donc ici que de leur intégration.

---

[1] "Il sait parler chinois".

[2] Exemple pris de la spécification d'OpenTag 1997 [OpenTag 1997]

2.3.2 Conversion des mémoires au format TMX

Les descriptions de TMX sont compatibles avec XML. Ainsi, aucune conversion n'est nécessaire. Un mécanisme de gestion des segments de la mémoire est toutefois à définir.

2.3.3 Génération du texte traduit

Une fois la structure cible de la traduction obtenue à un niveau d'abstraction élevé, il est nécessaire de redescendre vers ce premier étage décrit sous XML pour obtenir la version finale cible. La génération d'un segment de document au formalisme XML est donc nécessaire.

2.4 Lien entre le segment source et le segment cible

Les étages 0 des structures sources et cibles (c'est-à-dire les segments XML sources et cibles) ne sont liés qu'au niveau macroscopique : ils se correspondent en tant que traduction l'un de l'autre. Cette correspondance est construite implicitement lors de la traduction d'un segment nouveau. Elle est donnée implicitement par la contiguïté des segments sources et cibles pour des segments provenant de la mémoire.

3. Etage 1 : caractères

3.1 Opérations

Les caractères sont codifiés dans le segment XML d'une façon qui peut être différente de celle dont on a besoin pour les traitements. En voici deux exemples :

- XML utilise des références d"entités comme par exemple "é" qui correspond à "é" dans une représentation "classique" (ASCII 8 bits) en français
- certaines structures peuvent ne s'appliquer qu'à un seul caractère. Dans le cas de fichiers d'aide sous Windows, les noms de menus sont par exemple soulignés et en majuscule sous la lettre qui servira pour appeler ce menu par raccourci clavier comme dans "ReChercher". Nous devons avoir un moyen de coder cela.

Pour ces raisons, nous avons besoin de pouvoir représenter les caractères pour pouvoir montrer quels traitements peuvent leur être appliqués, et pour représenter également le résultat de ces traitements.

3.2 Description

3.2.1 Généralités

Les nœuds de caractères forment ainsi le premier treillis de la structure. Ces nœuds peuvent se transformer en d'autres nœuds de lettres ou de glyphes via des liaisons qui retracent les traitements appliqués aux nœuds. Par ces liaisons, les structures liées à des caractères peuvent aussi être représentées par des liens avec les autres étages supérieurs qui sont décrits par la suite. Voici un segment et son treillis de nœuds associé :

<**s**>This_course_<**mobj rid=”index1”**/>dœs_<**hi attrib =”02”**>not</**hi**>_cover _publishing <**hi attrib =”01”**>Web</**hi**> <**mobj rid=“Index2”**>_page content.</**s**>

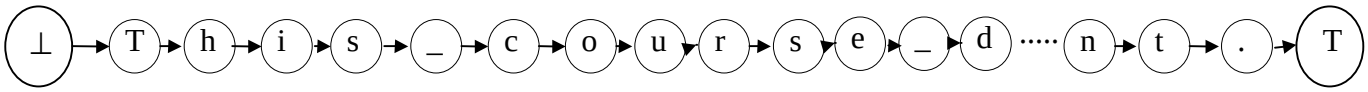


Figure 1 : Représentation des caractères : étage 1

Tous les caractères du segment XML qui ne font pas partie d'une balise sont représentés.

3.2.2 Définition du type de nœud pour l’étage 1

Définition : Nœud de type 1

estampille par rapport à l'ensemble de la structure[3]
rang00 dans le segment XML
rang01 dans le segment XML (c'est-à-dire élément dont il fait partie)

[3] Cet identificateur deviendra clair avec l'ajout d'autres étages : elle permet d'identifier le noeud par rapport à l'ensemble de la structure

rangT caractère représenté nœud de caractère précédent nœud de caractère suivant

Remarque :

- L'élément dont fait partie le caractère est, dans le cas d'un segment représentant un texte en français par exemple, soit un mot soit une référence d'entité. Il y a cependant des exceptions fréquentes qui sont les entités représentant les caractères diacritiques du français. Un tel caractère est représenté par une référence d'entité dans un segment XML, qui est elle-même un élément à part entière. Le rang01 du premier caractère "&" de la référence d'entité sera donc le rang01 de cette entité.
- Dans le cas de l'exemple précédent, une transformation de ces caractères en d'autres plus adaptés aux traitements linguistiques qui suivent (une lemmatisation par exemple) peut avoir lieu, comme dans l'illustration suivante :

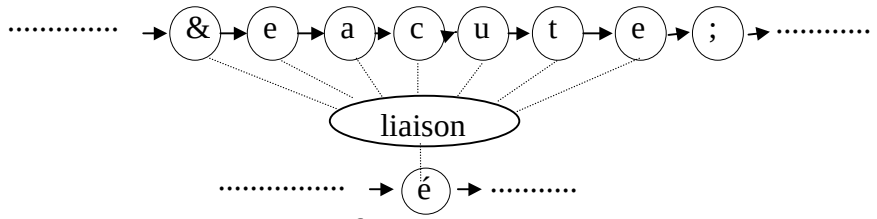


Figure 2 : Liaison formalisant un transcodage

- Le rangT exprime le numéro d'occurrence du nœud parmi les autres nœuds de caractères. De façon simplifiée cela correspond au n-ième nœud de caractère du treillis.
- Les caractères blancs, s'il y en a, sont aussi numérotés dans le segment XML, et représentés par un nœud de caractère.
- Les ponctuations donnent aussi lieu à un nœud de caractère de cet étage.

3.3 Calcul

3.3.1 Procédure de création à partir du segment XML

Première phase

La procédure consiste à parcourir le segment original XML, et à créer un nœud pour chaque caractère ne faisant pas partie d'une balise. Dans chaque nœud seront enregistrés les informations correspondant à ce caractère (se reporter à la définition d'un nœud de caractère).

Deuxième phase

Si l'application de règles de conversion de caractères ou le passage d'analyseurs montre qu'un groupe de caractères correspond à une nouvelle entité, un nœud représentant cette entité peut alors être créé à un chemin parallèle, ou un "sous-étage" (cf. "é" et "é" ci-dessus). Une liaison montre alors l'origine de ce caractère.

3.4 Correspondance entre les étages 0 et 1

Le lien avec l'étage 0 du segment XML se fait donc via l'occurrence du mot dans le segment XML, qui est donnée par ses rang00 et rang01.

3.5 Lien entre source et cible pour l'Etage 1

Ces liens permettent de faire se correspondre les mots source et cible des treillis de mots source et cible. Ce sont des liens fondamentaux exprimés par une liaison.

4. Étage 2 : mots

4.1 Opérations

L'intérêt majeur de cet étage est de fournir la liste des mots du texte utilisables par la plupart des outils linguistiques car seuls les caractères du texte à traiter y sont représentés. Ainsi les extracteurs de terminologie, les aligneurs, ou les lemmatiseurs pourront être appliqués à ce niveau. C'est aussi sur cette chaîne de caractères que se fait la recherche de correspondances avec la langue cible ou source dans un dictionnaire de formes.

4.2 Description

4.2.1 Généralités

L'ensemble des mots est représenté par le deuxième treillis de nœuds. Un nœud est affecté à chaque mot. Ce texte doit être représenté à l'aide d'une table de caractères qui a été choisie au niveau du treillis des caractères, comme indiqué précédemment. Un nœud comporte au moins les informations suivantes :

- le mot lui-même
- le rang00 d'occurrence défini à l'étage 0
- le rangT d'occurrence au niveau du texte (le n-ième mot)

Une fois ce niveau construit, si cela s'avère nécessaire pour certains traitements (comme la lemmatisation, par exemple), la suite de mots peut être récupérée par simple lecture séquentielle de la chaîne des nœuds.

Voici une illustration de ce deuxième étage. Nous reprenons l'exemple précédent.

<s>Ce_ **<mobj rid="index1"/>** cours_est_dispensé_en_un_ **<hi attrib ="02">** jour **</hi>**_et_animé_par_un **<hi attrib ="01">** instructeur **</hi>** .**<mobj rid= "Index2"> <s>**

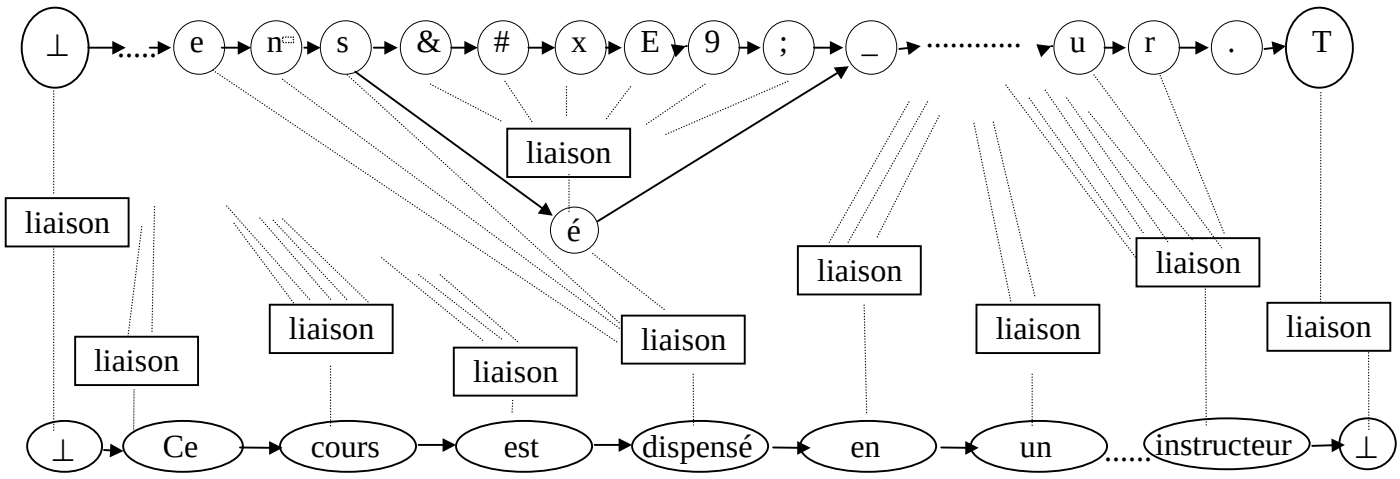


Figure 3 : Etages 1 et 2 : les caractères, les mots, et leur relations

Chaque mot est représenté par nœud elliptique. Les informations relatives au mot représenté sont listées dans le nœud. Un nœud de début appelé "inf" et un nœud de fin

appelé "sup" entourent la liste des nœuds de mots. Des "liaisons" (des rectangles sur le schéma) expriment les liens entre nœuds (voir le "é" ci-dessus). Ces relations peuvent s'appliquer à des nœuds d'un même étage ou d'étages différents.

4.2.2 Définition du type de nœud pour l'étage 2

Voici une proposition de structure pour les nœuds de type t2 du premier étage :

Définition 2 : Nœud de type t2

```
Nœud t2= {
        (estampille par rapport à la structure entière)
        (rang00 du dernier caractère du segment XML formant le mot)
        (rang01 du dernier élément du segment XML formant le mot)
        (rangT, soit encore l'identificateur pour cet étage)
        (chaîne des caractères du mot, dans le codage adéquat)
        (liste des nœuds précédents)
        (liste des nœuds sortants)
        (liaisons)
              }
```

4.3 Calcul

4.3.1 Procédure de création à partir du document XML

Les nœuds de mots se construisent à partir du treillis de caractères, et non pas directement à partir des caractères du segment XML. Pour ce faire, les caractères doivent être exprimés dans le codage et la transcription adéquate pour le travail linguistique qui va suivre. Cette opération est assurée au niveau du treillis des caractères, comme on l'a vu.

Le passage des caractères aux mots se fait par une procédure de segmentation du flot de caractères en mots. Cette procédure peut être triviale pour les langues à séparateur : il suffit de récupérer les chaînes comprises entre espaces ou ponctuations. Dans le cas de langues comme le japonais ou le thai qui n'ont pas de séparateurs entre mots, il est fait appel à un algorithme de séparation qui se base en particulier sur les données d'un dictionnaire. Nous ne discutons pas les procédures de segmentations de ce type qui relèvent d'une autre étude. Pour la suite nous supposons donc que nous disposons d'une procédure Segment(liste) qui, pour une liste de caractères données donne la liste des mots contenus.

4.3.2 Algorithme

A partir de l'étage 0 décrit sous XML, l'extraction des mots du segment XML se fait selon l'algorithme suivant :

Algorithme 1 : Création du niveau 2

```
Début
l = liste vide
Pour chaque nœud du dernier chemin du treillis des caractères
      concaténer l'élément à la fin de L
Fin Pour
LL = Segment(L)
Créer un nouveau treillis T2 (étage 2)
Pour chaque mot de LL
      créer un nœud de T2 comportant
```

```
Fin Pour
Fin
```

4.3.3 Autres procédures pour l'étage 2

Procédure de récupération du segment de texte nu

La chaîne linéaire du segment de texte nu est récupérée par simple filage des nœuds de mots.

Groupement, remplacement de mots

Il peut être nécessaire d'effectuer des opérations d'édition sur les mots (effacement, ajout, replacement, fusion, division,..). A chaque opération, une partie de la première liste de mots peut être modifiée. Dans ce cas, la partie modifiée est reconstruite en une nouvelle liste qui vient "s'accrocher" à l'ancienne. Un nouveau chemin est alors créé.

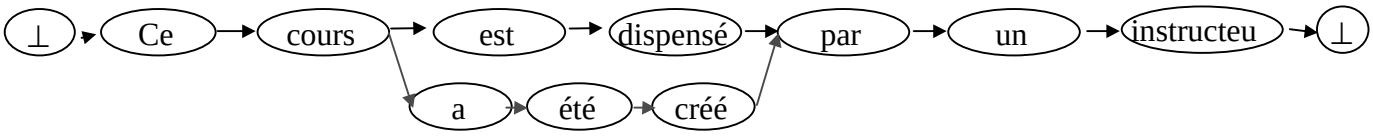


Figure 4 : Création d'un nouveau chemin dans le treillis.

Il sera montré plus loin comment distinguer les chemins entre eux à l'aide de types d'arcs de structure.

4.4 Correspondance entre les étages 0 et 1

Le lien avec l'étage 0 du segment XML se fait donc via l'occurrence du mot dans le segment XML, qui est donnée par ses rang00 et rang01 par rapport au segment XML.

4.5 Lien entre source et cible pour l'Etage 2

Ce lien est important et jouera un rôle dans le transfert vers la structure cible, bien que ce transfert doive être supporté principalement au niveau des lemmes (voir étage 5). Ainsi, les liens entre l'étage 2 source et l'étage 2 cible se font au niveau des mots (des formes). Ces liens pourront être établis soit par un dictionnaire bilingue de formes, soit par une base de données terminologiques comme celles utilisées actuellement dans les OTFM1g (Multiterm, TermStar). Cela permet d'utiliser la connaissance contenue dans ces bases de données (qui est importante), sans recourir à une lemmatisation et un dictionnaire de lemmes. Ces informations ne permettront certainement pas de construire tous les liens. Ces liens apportent cependant une information complémentaire à celle de l'analyse linguistique dans le cas d'une ambiguïté. Par exemple si le lemme du mot ne se trouve pas dans le dictionnaire de l'analyseur, alors qu'un terme correspondant au mot est présent au niveau d'un lexique ou d'une base de données gérée par l'utilisateur, c'est ce dernier qui donnera l'information pertinente.

5. Étage 3 : balises doubles

5.1 Opérations

Les balises doubles XML (ou “balises à contenu”) se comportent toutes de la même façon par rapport au reste du code XML : elles qualifient leur contenu. Une des applications de ces balises doubles est la spécification des groupes de mots qui pourront être mis en valeur par l'application d'une feuille de style. Cela permet de gérer la mise en forme du texte.

Ce niveau de représentation doit permettre en particulier de faire correspondre les balises doubles aux mots eux-mêmes, tout en séparant information de la chaîne de caractères du mot. Au niveau de l’analyse, cette correspondance est trouvée par un parcours du segment de l’étage 0. Au niveau de la génération, lorsque le transfert de structure a lieu pour la traduction de ce segment, les liens entre mots et balises doubles du segment cible sont aussi représentés à cet étage, dans la structure cible. Cette correspondance permet alors le transfert du formatage sur les mots, ou sur les groupes de mots cibles.

Dans les documents XML, un des types classiques de balises de mise en valeur sont les balises doubles <hi> et </hi> entourant les mots concernés. Plus généralement, d’autres balises doubles de XML peuvent gérer autre chose que la mise en valeur, mais syntaxiquement leur comportement est le même. Toutes les balises doubles seront donc gérées à cet étage. Nous n'utiliserons dans nos illustrations que le type <hi>, mais il n'y a rien de restrictif en cela.

5.2 Description

5.2.1 Présentation

La spécification des balises doubles se fait sur un étage supérieur parallèle à celui du treillis des mots. La structure comporte alors en tout trois étages, un pour les caractères, un pour les mots, et ce dernier pour le formatage.

Voici la représentation d'un treillis de balises doubles (un nœud par couple de balises doubles), dans lequel seuls les attributs valeurs des mots apparaissent dans les nœuds du premier étage :

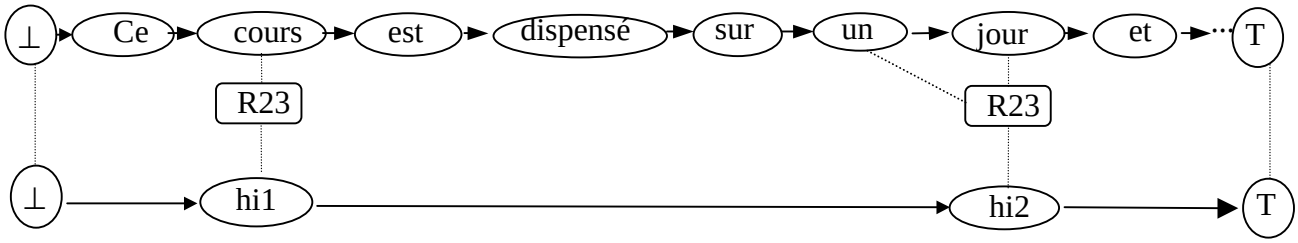


Figure 5 : Treillis à deux étages : un pour le texte, l’autre pour les couples de balises doubles

En toute généralité, le domaine d'application de ces balises doubles peut être autre qu'un mot en entier. Elles peuvent en effet ne porter que sur quelques caractères. Dans le segment XML qui est repris ci-après, c'est le cas pour les deux premières lettres de "cours" qualifiées par les balises <hi3> et </hi3>. Cela est alors représenté par une liaison de type R13, entre l'étage 1 des caractères et l'étage 3 des balises doubles, comme le montre la figure suivante :

<s>Ce_<hi1><hi2>co</hi2>urs</hi1>_est_dispens&#xE9_sur_un_<hi1>jour</hi1>_et...</s>

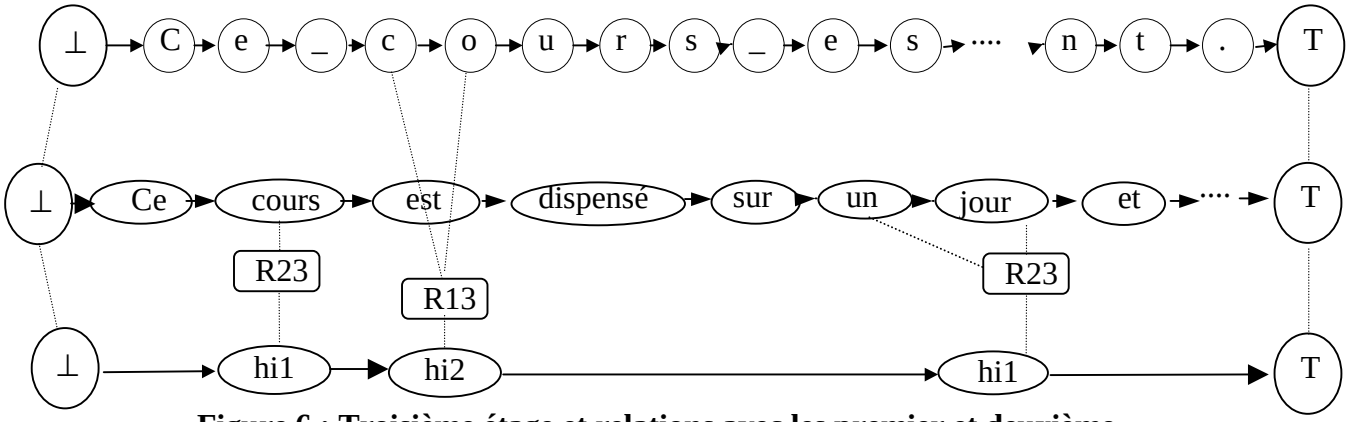


Figure 6 : Troisième étage et relations avec les premier et deuxième.

Un nœud est associé à un couple de balises doubles. Les nœuds sont disposés sur l'axe des nœuds dans l'ordre d'occurrence du premier nœud du couple de balises.

5.2.2 **Séquentialité**

La notion de séquentialité de l'information est importante au niveau des caractères (étage 1) et des mots (étage 2). Elle garde une importance, moindre toutefois, pour les balises, au niveau du segment, en ce sens que l'application de ces balises se fait sur un ensemble de caractères qui ont eux une séquence.

5.2.3 Factorisation de l'information

La représentation précédente a l'avantage d'être factorisante, en ce sens qu'elle ne demande qu'un nœud par couple de balises doubles. Cette factorisation va plus loin pour les formats appliqués à deux endroits différents, comme au paragraphe suivant.

5.2.4 Cas de l'application de couples de balises croisées

Une représentation basée sur XML autorise le croisement des balises doubles. Le choix précédent permet de représenter cela aisément :

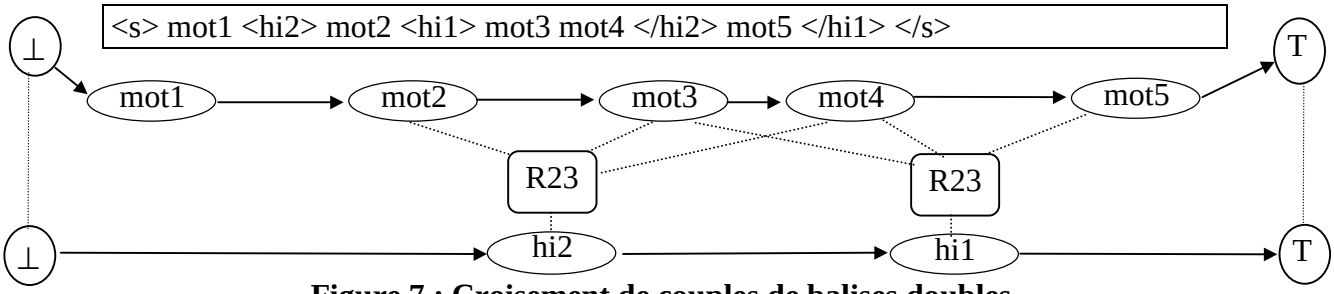


Figure 7 : Croisement de couples de balises doubles.

On se rappellera d'autre part que ce choix permet de gérer également l'application d'un objet de l'étage 3 (une double balise) à des objets de l'étage 1 (des caractères ou des glyphes, après analyse), comme le montre la figure 6. Cette notion s'étend aux étages qui seront définis ultérieurement avec la même souplesse.

5.2.5 Définition du type de nœud pour l'étage 3

Un nœud de type t3 pour l'étage 3 doit comporter à la fois des informations d'identité, de données (ce que représente le nœud) de structure (arcs liés), et de liens

(avec les nœuds d'étages différents). Ces informations sont résumées dans la définition suivante :

Algorithme 2 : Nœud de type 3

```
Nœud t3= {
        (estampille de ce nœud par rapport à la structure entière)
        (rangT du nœud à l'étage 3, parmi l'ensemble des nœuds de l'étage)
        (nombre n de couples de balises doubles de même type du segment)
        [(code 1 trouvé dans le document,
                rang00 des places de caractères occupées par l'instance,
                rang01 d'occurrence de l'instance dans le segment XML)
        ....
        [(code n trouvé dans le document,
                rang00 des places de caractères occupées par l'instance,
                rang01 d'occurrence de l'instance dans le segment XML)]
        (liste des arcs entrants)
        (liste des arcs sortants)
        (liste des relations liées à ce nœud)
            }
```

Il est important de garder l'expression du code d'origine trouvé dans le document, et cela est rendu possible grâce aux enregistrements des tableaux. Il se peut en effet que la nature exacte de la balise double (sa sémantique), ne soit pas connue et ne puisse être interprétée. On pourra alors la reconstituer au moment de la génération du texte cible en se servant de cette expression d'origine. Cela permet en particulier un comportement robuste de transfert du code vers la partie cible.

5.2.6 Arcs entrants et sortants, types d'arc.

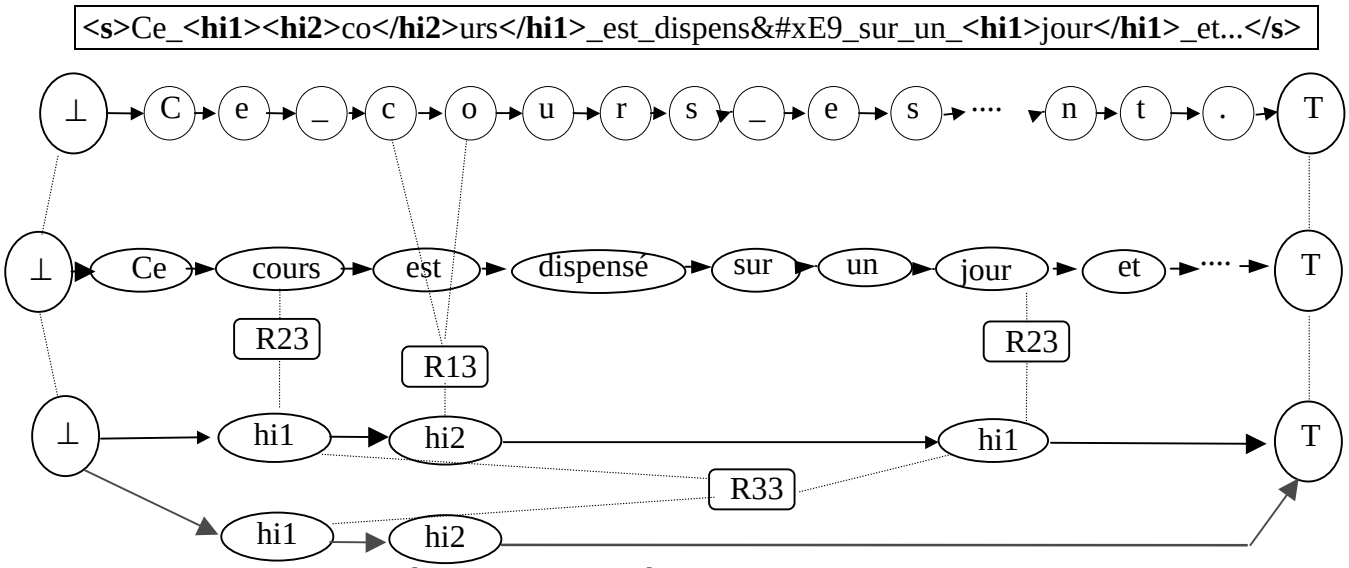


Figure 8: Trois étages : les caractères, les mots, et les balises doubles

Il est important de donner un sens de parcours des nœuds. Cela est obtenu grâce à l'orientation des arêtes, et entraîne la classification des arcs en deux classes au niveau du nœud : les "arcs entrants" qui "arrivent" sur le nœud, et les "arcs sortants" qui en partent. Cela ne suffit pas pour couvrir l'ensemble des cas de structure, comme le montre la figure 8.

Dans le schéma précédant le même type de couple de balises doubles (<hi1>) est appliqué à deux mots différents.

Si un premier parcours du segment XML peut mener à la suite de nœuds (hi1), (hi2), (hi1), une éventuelle analyse de redondance peut entraîner la création d'un deuxième sous-étage dans lequel il n'y a pas de répétition d'information, et donc 2 nœuds au lieu de trois, comme dans le schéma ci-dessus. Cela donne à l'étage une structure de treillis non trivial.

Dans ce cas, il est intéressant de montrer que l'arc partant du nœud inf du deuxième chemin (de haut vers le bas) est d'un type différent de celui qui part tout droit. Il peut bien sûr il y avoir plus que deux arcs par nœud. Cela mène à la définition de la notion de type d'arc. Le type d'un arc est un entier qui ordonne les arcs suivant l'ordre de création par exemple (dans ce cas cela revient à "colorer" les arcs). Un nœud différent des bornes inf et sup comporte donc au moins un arc entrant et un arc de même type, et éventuellement des arcs entrants ou sortants d'un autre type d'arc.

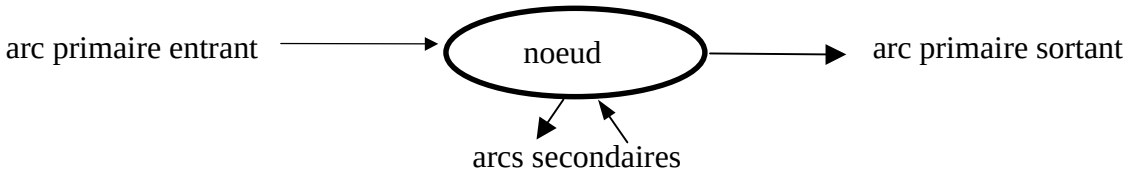


Figure 9 : Arcs primaires entrants et sortants, supérieurs et inférieurs

Comme chaque nœud ne comporte qu'un nœud primaire entrant et un nœud primaire sortant, il est possible de reconstituer une structure de liste interne au treillis, en parcourant tous les arcs d'un type donné.

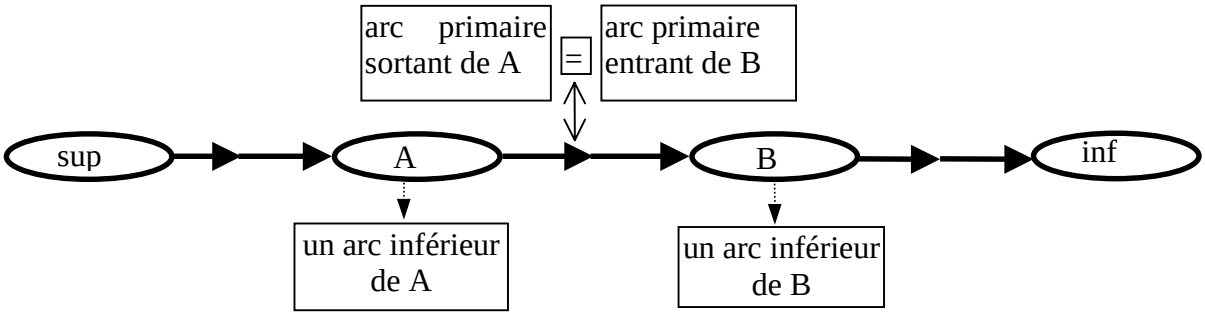


Figure 10 : Structure de liste interne au treillis : un sous-étage

5.2.7 Cas de texte à traduire au sein d'une balise : notion de sous-segment

Parfois des chaînes de texte internes aux balises sont à traduire. Dans le cas des balises d'index, par exemple, outre l'identité de l'index ("id = 1"), la chaîne sous laquelle il doit apparaître dans la table des index peut être spécifiée (ref = "bouton"), et doit être traduite.

Nous sommes alors amenés à créer un sous-segment. Il s'agit d'un segment comme les autres, mais qui sera à intégrer à l'intérieur de la balise à la génération. Ce sous-segment sera représenté par une nouvelle structure complète (un segment XML accompagné d'en ensemble de treillis liés).

5.3 Calcul

Le calcul se fait en plusieurs "passes" dont la première consiste en un parcours du segment XML, pour repérer les balises doubles entourant le mot formaté. Aussi, lorsqu'une balise de format (de début comme , ou de fin comme) est trouvée, un nœud relatif à cette balise est créé. Cela donne un treillis réduit à une liste de nœuds. Une deuxième passe sert à grouper les nœuds couples de début et de fin de balise en un seul nœud ; cela crée un second chemin dans lequel il y a exactement la moitié des nœuds du chemin précédent. Alors une troisième passe peut intervenir pour grouper entre elles les balises de même type s'appliquant à différents endroits. La trace de la portée de l'application de chaque balise est alors gardée dans des enregistrements du nœud. Voici la trame d'un algorithme pour ces trois phases.

Algorithme 3 : Calcul de l'étage 3

```
Début
                                                    //Première passe
Pour (chaque élément du segment)
    Si (c'est une balise double de type <b>[4] ou </b>)
        Créer un nœud de balise double Ni
            en spécifiant en particulier le code exact des deux parties de la
                balise double trouvée dans le segment XML
            le rang00 des places des caractères composant la balise
            le rang01 d'occurrence de la balise double
        Créer un arc primaire de Ni sup
        Supprimer l'arc primaire entre le nœud précédent N(i-1)[5] et le nœud "sup"
        Créer un arc primaire entre N(i-1) et Ni
    Fin Si
Fin Pour
                                                    //Deuxième passe
Pour (chaque nœud qui vient d'être créé, représentant une balise de début <b>)
    Parcourir la liste des nœuds suivants jusqu'à trouver sa balise de fin </b>
    Créer un nouveau nœud concentrant les information du couple de balises
    Placer ce nœud sur un nouveau sous-étage parallèle
Fin Pour
                                                    //Troisième passe
Début
Pour chaque nœud N du sous-étage précédent
    Pour chaque nœud N' du même étage
        Si N est similaire à N'
            intégrer à N les enregistrements de N'
            supprimer N'
        Fin Si
    Fin Pour
```

[4] Les balises doubles peuvent être reconnues directement en parcourant le segment (qui est fini) : pour la balise en cours de type , on recherche alors l'occurrence de la balise . Il est aussi possible de maintenir une table de telles balises. Une méthode mixte sera certainement la plus adaptée : on regarde dans la table si la balise fait partie d'un type connu, sinon on cherche à déterminer s'il s'agit bien d'une balise double par parcours du segment. Il faudra aussi gérer les omissions de balises ou les balises implicites (voir [Goldfard 1990]).

[5] F_{i-1} représente le noeud de balise double précédent à l'étage 2

Fin

Exemple :

Considérons le segment XML suivant dans lequel ne sont représentées que les balises doubles. Alors les trois passes de l'algorithme précédent créent chacune une listes du treillis représenté ci-dessous.

<s><hi1>.....<hi2>.....<hi1/>.....<hi2/>.....<hi2>.....<hi1>....<hi1/>...<hi2/></s>

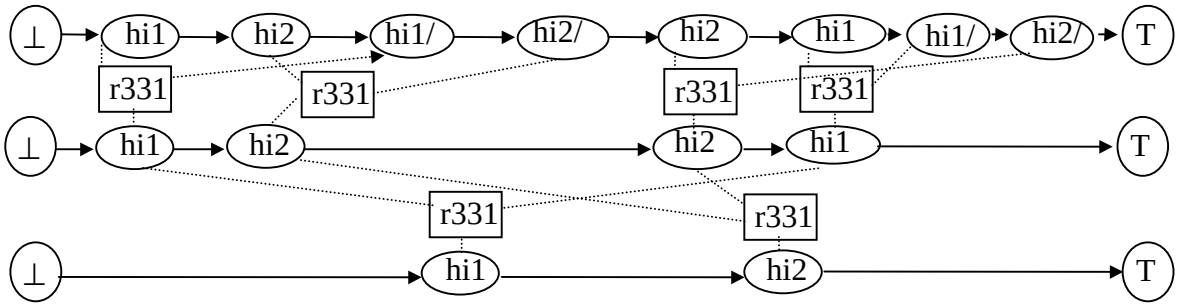


Figure 11 : Construction du treillis des balises doubles en trois passes

5.4 Correspondance de l'étage 3 et les étages inférieurs

La correspondance entre les étages 0 et 3 est assurée par les identificateurs des rang00 et rang01 de la balise double. Les correspondances avec les autres étages sont assurées directement par les liaisons inter-étages.

5.5 Lien entre source et cible pour l'étage 3

5.5.1 Principe

La correspondance entre les nœuds de balises doubles de la partie source et de la partie cible de notre représentation ne se fait a priori pas directement. Elle se fait via les nœuds de texte de l'étage 1, ou via les nœuds des lemmes de l'étage 5 à venir.

5.5.2 Etage 3 et correspondances de formats sources et cibles

Sous le formalisme XML, une feuille de style sera appliquée en particulier aux balises doubles pour donner la mise en forme du document source. Le passage à la mise en forme du document cible se fait alors en trois temps :

Transfert de structure

Il sera expliqué plus tard comment la structure cible est trouvée par transfert via un modèle pris parmi les couples de segments stockés en mémoire de traduction. Une procédure permet donc d'obtenir la structure cible.

Transfert de feuille de style

La feuille de style de la langue cible peut être différente de la source. Cela peut être dû à la culture des personnes qui utilisent cette langue cible, ou à une politique de localisation de tel ou tel constructeur pour ce pays. Il est donc important de procéder à un transfert de l'une à l'autre (par heuristiques idiosyncratiques par exemple).

Application de la feuille de style à la structure cible

C’est seulement une fois que l’on a la structure cible et la feuille de style cible que l'on pourra trouver la mise en forme du document cible par application de la deuxième sur la première.

6. Étage 4 : monobalises

6.1 Opérations

Cet étage a pour but de représenter les monobalises des documents XML. Cela permet de coder en particulier les éléments non textuels comme les marques d'index, de notes de bas de page, les hyperliens des fichiers d'aide, ou les variables des fichiers de ressources. Cette séparation permet de ne pas mélanger ces données avec le flot de texte (nous avons vu dans la première partie que cela est source d'erreurs).

6.2 Description

6.2.1 Type souhaité de représentation

Les objets non linguistiques ont un comportement de "pseudo-mots". Ils s'insèrent dans le flot de la phrase entre les mots, et sont représentés par des monobalises à contenu vide dans le document XML. Nous les insérons ainsi dans notre représentation en créant un nouveau treillis, venant compléter ceux des caractères, des mots, et des doubles balises.

Exemple :

<s>Ce_<hi1><hi3>co</hi3>urs</hi1><ix1/> _est_ dispens&#xE9_sur_un_ <hi1> jour <ix2/> </hi1> _et...</s>

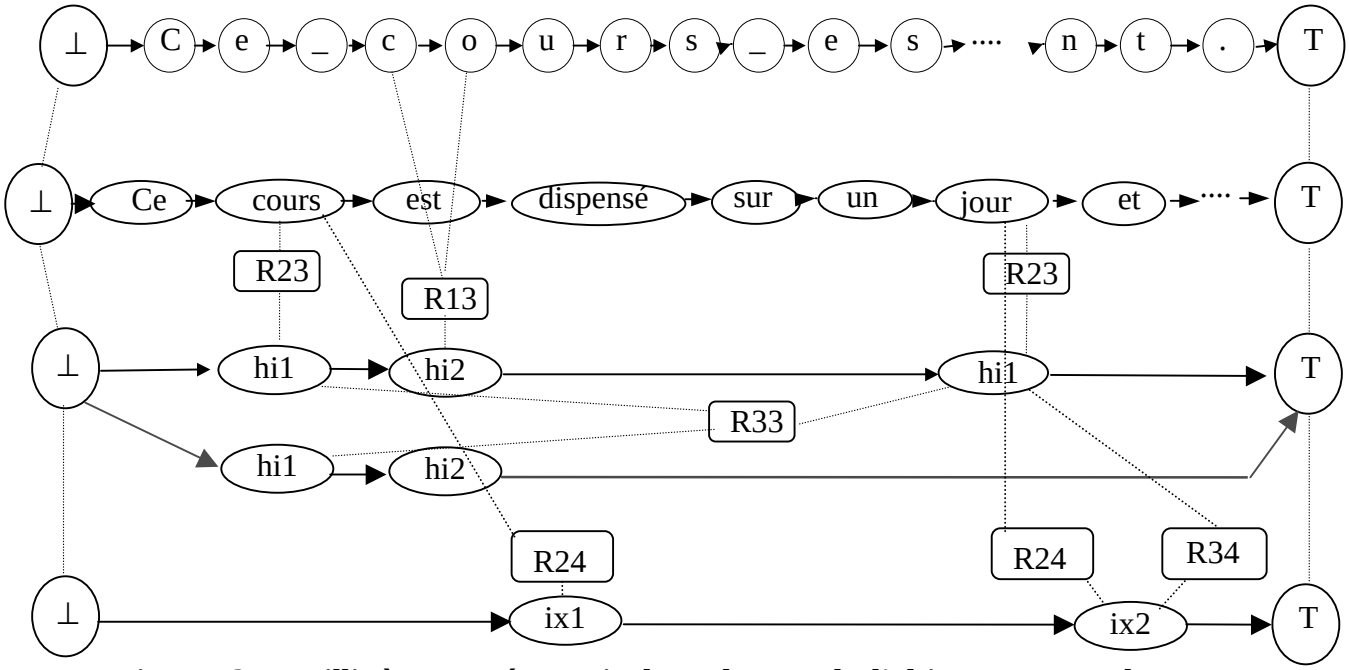


Figure 12 : Treillis à quatre étages, incluant les nœuds d'objets non textuels

Un quatrième étage de nœuds de monobalises est ajouté. Les balises sont disposées dans une première phase sur l'axe dans l'ordre d'occurrence trouvé par un parcours de gauche à droite du segment XML. Les informations relatives à la monobalise sont enregistrées dans le nœud. En particulier, la position de la monobalise sur le segment XML (les rang00 et rang01) sont conservés.

Dans une seconde phase, une règle de comportement des balises d'index pour la langue source peut impliquer que l'index s'applique sur le mot précédent qui est retrouvé par sa position relative à celle de la monobalise d'index, et un parcours de

l'étage 2. Alors il est possible de placer une liaison entre le nœud de mot concerné et le nœud de monobalise.

Les doubles balises étant parfois placées autour de monobalises dans les segments XML, et selon la sémantique des doubles balises et celle des monobalises, la zone d'influence des balises doubles peut s'étendre aux monobalises. C'est le cas par exemple du formatage de liens hypertextes dans les fichiers RTF codant les documents d'aide de Windows. Dans notre représentation, ceci est alors matérialisé par une liaison entre le nœud de balise double et le nœud de monobalise concernés.

De la même façon que pour les étages précédents, la construction d'un sous-étage supplémentaire du treillis de l'étage 4 est envisageable. Un nœud de monobalise peut donc représenter plusieurs instances de monobalises du même type. Dans ce cas, autant d'enregistrements gardent dans le nœud les renseignements relatifs à une instance donnée. Ceci est d'ailleurs permis par les nœuds de type 4 définis ci-dessous.

6.2.2 Définition du type de nœud pour l'étage 4

Algorithme 4 : nœud de type 4

```
Nœud t4= {
        (estampille de ce nœud par rapport à la structure entière)
        (rangT du nœud à l'étage 4, parmi l'ensemble des nœuds de l'étage)
        (nombre n de monobalises du segment représentées)
        [(code 1 trouvé dans le document,
                rang00 des places de caractères occupées par l'instance,
                rang01 d'occurrence de l'instance dans le segment XML)
        ....
        [(code n trouvé dans le document,
                rang00 des places de caractères occupées par l'instance,
                rang01 d'occurrence de l'instance dans le segment XML)]
        (liste des arcs entrants)
        (liste des arcs sortants)
        (liste des relations liées à ce nœud)
            }
```

La définition est donc la même que pour les nœuds de l'étage précédent.

6.3 Calcul

6.3.1 Première phase

Dans un premier temps, le calcul des nœuds de ce quatrième étage se fait aussi par parcours des éléments du segment XML. Lorsqu'une monobalise est rencontrée, un nœud de type 4 est créé au quatrième étage. Les monobalises sont reconnues grâce à la syntaxe XML : elles sont toutes du type : <balise/>, ou "balise" est un ensemble de caractères. Ainsi l'algorithme de construction du troisième étage s'écrit :

Algorithme 5 : Calcul de l'étage 4.

```
Pour chaque élément du segment
    Si c'est une monobalise de type <b/>
        Créer un nœud de monobalise balise Ni en spécifiant:
```

rangT du nœud à l'étage 4, parmi l'ensemble des nœuds de l'étage 4
estampille de ce nœud par rapport au treillis entier
le code exact trouvé dans le segment XML pour la balise
rang00 des places de caractères occupées par les deux parties de balise
rang01 d'occurrence des deux éléments de la balise double
Lier le nœud à la structure en :
créant un arc primaire de Ni sup
supprimant l'arc primaire entre le nœud N_{i-1} et le nœud "sup"
créant un arc primaire entre N_{i-1} et Ni
Fin Si
Fin Pour

6.3.2 Deuxième phase : création des relations

Les nœuds de monobalises étant créés, seules les relations avec le segment XML ont été indiquées dans la première phase. Cette deuxième phase a pour but de créer les autres liaisons par comparaison des places de caractères du segment XML relatives à ce nœud et aux autres nœuds de la représentation. Ainsi, en parcourant l'ensemble des nœuds existants, si ces nœuds occupent une place relative adéquate, et selon leur nature, une liaison sera établie entre eux. La construction de ces liaisons est guidée par des règles à définir.

6.3.3 Concentration de l'information

De la même façon que pour l'étage précédent, une concentration de l'information pour les balises de même type peut être réalisée.

6.4 Correspondance entre les niveaux 4 et inférieurs

La correspondance avec l'étage 0 est toujours assuré via le numéro d'occurrence dans le segment XML (rang00 et rang01). La correspondance avec les mots de l'étage 4 se fait via les liaisons.

6.5 Lien entre source et cible pour l'étage 4

Ces liens sont aussi gérés par un graphe de liaison d'un type donné.

7. Étage 5 : Lemmes, formes, et informations grammaticales

7.1 Opérations

Pour traiter linguistiquement le segment, nous avons besoin d'information linguistique sur les mots du premier étage. Cette information est insérée à ce cinquième étage. Cet étage sera donc le réceptacle de la recherche d'information linguistique sur les mots et l'ensemble de l'expression des mots du premier étage. Cette recherche pourra par exemple s'effectuer par l'appel d'un analyseur morphologique. Il existe un cinquième étage pour la partie source, et un pour la partie cible. Les opérations que supporte cet étage sont des opérations de transfert de lemmes vers la partie cible (traduction) et des modifications linguistiques dans la partie cible. Ainsi les mises au pluriel par exemple pour générer correctement le mot du premier étage se font à partir de cet étage.

7.2 Description

7.2.1 Généralités

Un nœud est affecté à chaque entité linguistique. Pour chaque nœud de mot de l'étage 2, il y a donc en principe au moins un lemme (en dehors des mots inconnus par l'analyseur, bien sûr). Les nœuds s'échelonnent sur une liste de nœuds à l'étage 5.

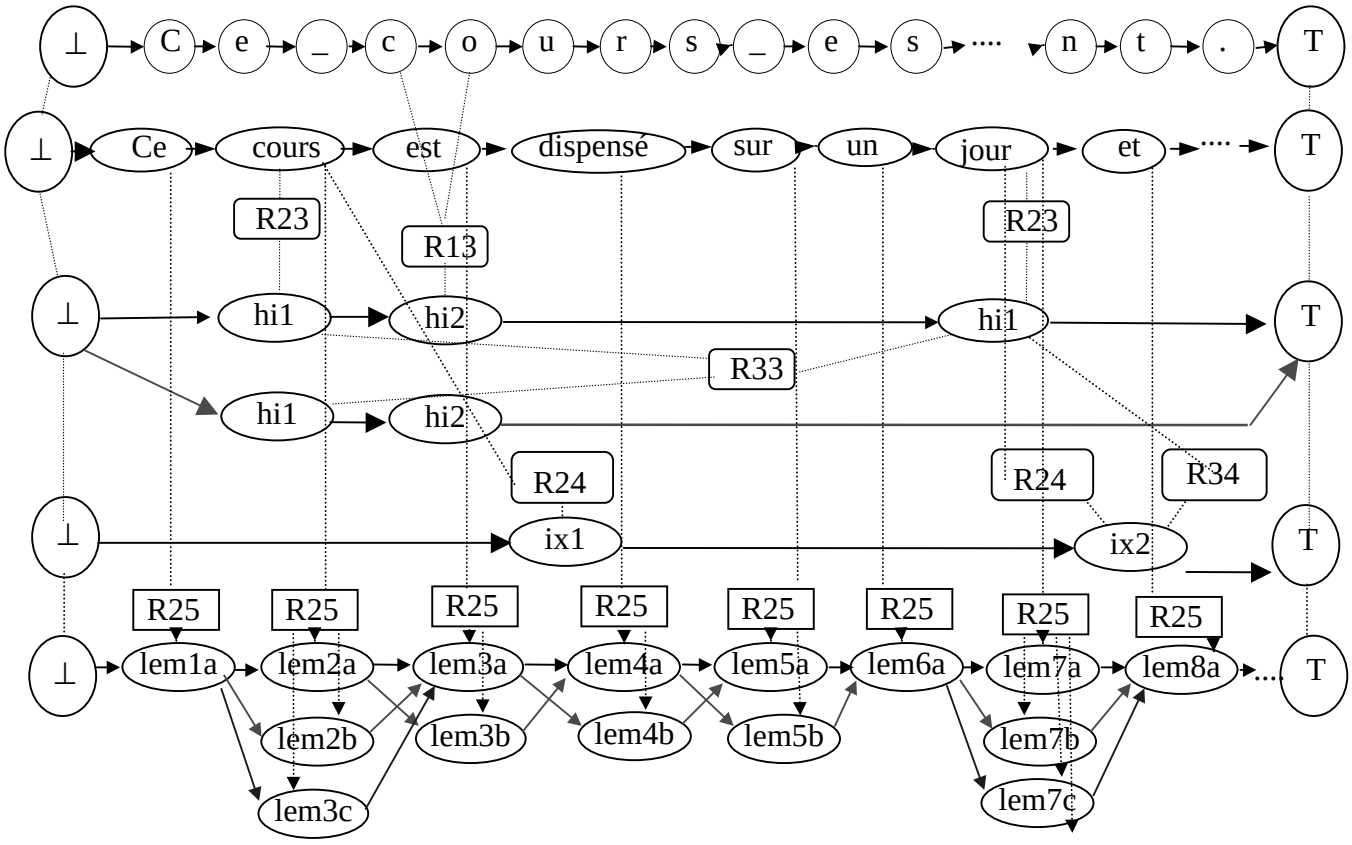


Figure 13 : Représentation d'une lemmatisation

7.2.2 Ambiguïté d'analyse

La lemmatisation de l'ensemble des termes d'une phrase peut mener à plusieurs lemmes possibles par mot. La structure doit donc prévoir cette possibilité. Cela est pris en compte par la possibilité de placer plusieurs nœuds sur un étage, transformant la liste de mots en un treillis non trivial. Des arcs de type (ou couleur) différent sont alors utilisés pour matérialiser les différents chemins parcourables donnant chacun une possibilité d'analyse linguistique du segment.

7.2.3 Chemins préférenciels, poids des arcs et complexité

Les algorithmes de parcours des différents chemins peuvent rapidement complexifier les traitements des structures de treillis. Pour permettre une approche plus rapide, il peut être intéressant de parcourir d'abord les arcs préférentiels. Dans le cas de différents lemmes possibles pour un mot donné, cette préférence peut être donnée par exemple par le lemmatiseur. Afin de pouvoir représenter ces préférences, il peut être utile de permettre à l'arc de posséder un poids entier ou réel (plutôt entier d'ailleurs pour que cela n'amène pas de problèmes de portage).

7.2.4 Mots composés

Une analyse plus précise peut montrer qu'en fait les lemmes de base peuvent être regroupés pour former un mot composé. Cela peut être représenté par un nouveau nœud lié aux nœuds de lemmes de base par des arcs typés (colorés) différemment, et par une liaison explicatrice de cette composition. Cela est montré dans la figure suivante.

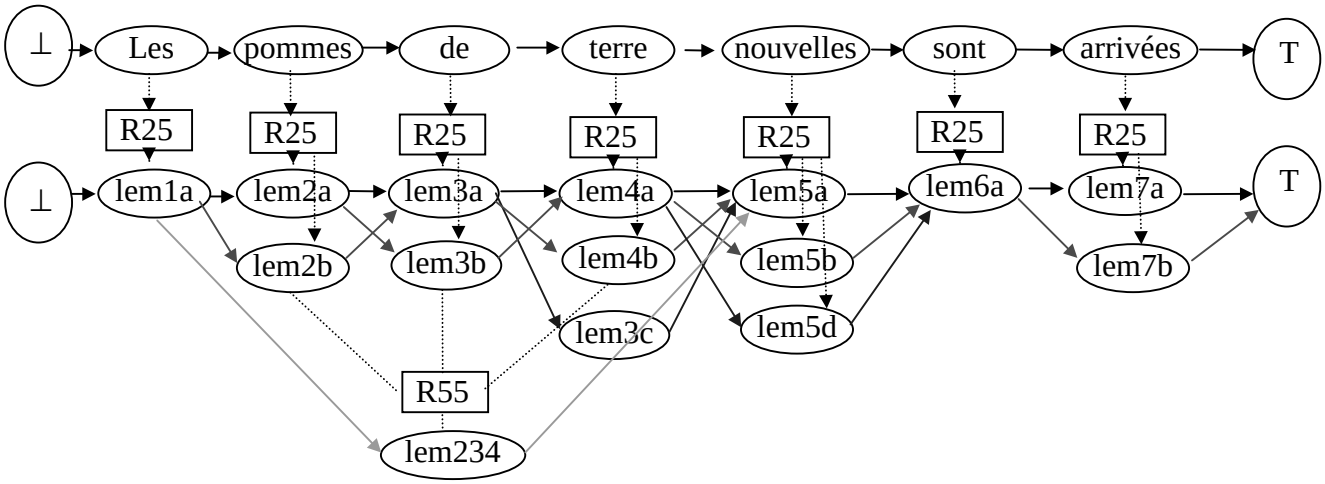


Figure 14 : Représentation des mots composés

7.2.5 Définition du type de nœud pour l'étage 5

Voici la définition du type de nœud nécessaire pour l'enregistrement des informations linguistiques de l'étage 5.

Algorithme 6 : Type de nœud t4 adapté à l'étage 5.

```
Nœud t5{
    (estampille d'identification au niveau de la structure complète)
    (rangT du nœud à l'étage 5 parmi l'ensemble des nœuds de l'étage 5)
    [ (lemme),
        (forme),
        (attributs lexicaux),
        (attributs sémantiques) ]
```

```
            (liste des arcs entrants)
            (liste des arcs sortants)
            (liste des relations liées à ce nœud)
    }
```

Remarques :

Rappeler la forme du mot correspondant peut permettre de ne pas avoir à aller le chercher au premier étage, car c'est une information qui sera utilisée plusieurs fois à cet étage.

Nous verrons plus tard que nous avons besoin d'information sémantique pour pouvoir traiter correctement notre traduction. Celle-ci peut être attachée à chaque occurrence de lemmes dans les nœuds de ce quatrième étage. Cette information sémantique peut se présenter sous différentes formes. Typiquement, une f-structure plus ou moins complexe, ou une décoration d'attributs plus ou moins complexes.

7.3 Calcul

C'est un fait nouveau pour les OTFM1g que de baser le processus de traduction sur des informations provenant d'une analyse linguistique (aussi simple soit-elle), et c'est un des points essentiels qui fait la différence entre les OTFM1g et les OTFM2g. L'algorithme va consister à parcourir la chaîne de mots.

Algorithme 7 : création de l'étage 5

```
Début
    Créer une chaîne de caractères vide L
    Concaténer les mots de l'étage 2 en une liste L
    Pour chaque lemme
        Si c'est le premier lemme du mot correspondant
            créer un nouveau nœud à l'étage 5
        Sinon
            ajouter un nœud en parallèle au nœud précédent relatif au même mot
        Fin Si
        Remplir les enregistrements du nœud en indiquant :
            les différents rangs
            le lemme
            la forme (correspondant au "mot" de l'étage 2)
            les attributs lexicaux
            les attributs sémantiques
            les arcs entrants et sortants
            les relations avec les nœuds de l'étage 2
    Fin Pour
Fin
```

Par la suite, l'application de connaissances concernant les groupes de mots entraîne la création de nouveaux nœuds et relations, dont on a montré plus haut la représentation.

7.4 Correspondance entre l'étage 5 et les étages inférieurs

La correspondance avec l'étage 2 est donnée directement par les liaisons. Il n'y a plus de correspondance directe avec les autres étages inférieurs.

7.5 Lien entre source et cible pour l'étage 5

Les étages 5 sources et cibles se correspondent par exemple à l'aide de liaisons trouvées grâce à des dictionnaires de lemmes bilingues. Ces liaisons lient les lemmes traductions l'un de l'autre. La figure ci-dessous donne un exemple de liaisons entre les nœuds des 4ièmes étages source et cible.

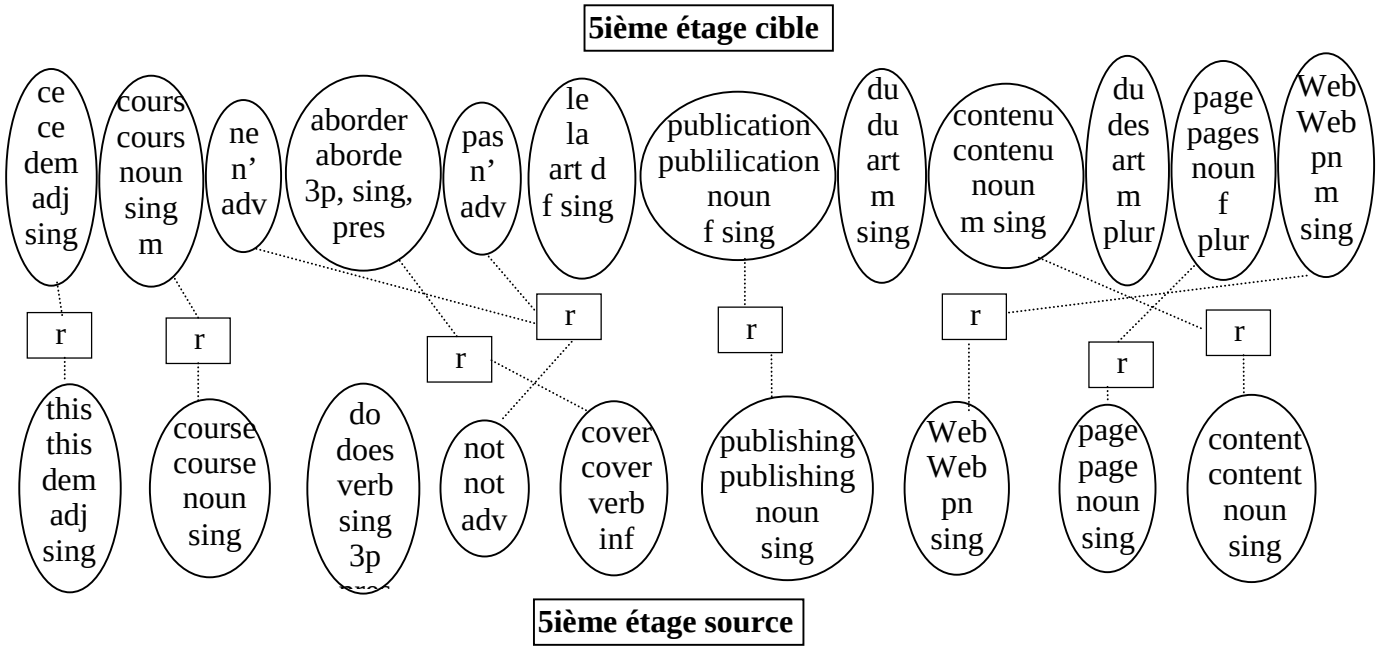


Figure 15 : Liens entre les cinquièmes étages source et cible

Les liaisons entre lemmes ne définissent en général ni de relation totale ni de relation fonctionnelle. En effet certains lemmes comme "le" n'ont pas de correspondants en chinois par exemple; les deux mots "ne" et "pas" correspondent à "not" en anglais.

8. Étage 6 : schémas linguistiques

8.1 Opérations

Les OTFM1g ne permettent pas d'utiliser simultanément plusieurs segments pour traduire une seule phrase. Pourtant, de simples compositions pourraient amener une fréquence de réponse plus élevée de ces outils. Prenons par exemple le cas du couple de phrases source et cible suivant :

	Source	Cible
Traduction souhaitée	This course teaches students **how** to support the various features of AAA.	Les stagiaires y apprendront **comment** prendre en charge les différentes fonctionnalités de AAA.

Ces phrases sont composées en langue source et cible de deux parties séparées par une expression "pivot" (en gras ici). Chacune des parties sources et cibles se correspondent une à une. Dans ce cas, nous voudrions bien pouvoir composer ces parties si elles ont été trouvées par ailleurs en mémoire comme suit :

Mémoire	Source	Cible
segment 1	this course teaches students	les stagiaires y apprendront
segment 2	to support the various features of AAA	prendre en charge les différentes fonctionnalités de AAA

Cela est d'autant plus intéressant que dans ce cas particulier, une traduction à l'aide de règles de traductions ne donnerait certainement pas cette correspondance, qui est pourtant celle du choix du traducteur, et qu'il faut donc respecter. Nous allons introduire pour cela la notion "d'expression pivot". Ce sixième étage sera dédié à la gestion de ces expressions pivots**{**xe "expressions pivots"**}**.

8.2 Description

8.2.1 Phrase à pivot

Introduction

Algorithme 8: Phrase à pivot

Une langue étant donnée, une phrase à pivot**{**xe "phrase à pivot"**}** est une expression textuelle composée de trois parties :

un premier segment
une expression pivot invariante
un deuxième segment

Aucune autre contrainte linguistique que la présence et l'invariance d'un "pivot" n'est requise. Les pivots peuvent être des locutions formées de plusieurs mots ou d'un seul. Voici une liste de quelques pivots possibles:

Japonais	Espagnol	Anglais	Français
kara	porque	because	parce que
moshi	si	if	si
soshite	y	and	et
koto	como	how	comment
toki wa	para	in order to	pour

Certains signes de ponctuation comme la virgule "," ou les deux points ":" jouent un rôle similaire. Voici maintenant des exemples de phrases à pivot :
Exemple :

Segments à pivot	Source	Cible
Phrase 1 (En-Fr)	This course teaches students **how** to support the various features of AAA.	Les stagiaires y apprendront **comment** prendre en charge les différentes fonctionnalités de AAA.
Phrase 2 (En-Jp)	Press the button **in order to** get some help.	helupu ga hoshii **toki wa** botan wo tsukete kudasai.
Phrase 3 (EsAm-Fr)	Compre se un pero **si** tiene miedo.[6]	Achète-toi un chien **si** tu as peur.

On notera que l'ordre des segments peut être inversé, comme dans l'exemple ci-dessus de l'anglais vers le japonais (traduction littérale du japonais : "aide (particule "ga") vouloir **quand** bouton {particule "wa"} appuyer {forme polie}.").

Multiplicité de la traduction d'une expression pivot

Ce type d'expression pivot n'a certainement pas de traduction unique en général. On pourra se reporter par exemple à [Furuse & Iida 1994] pour s'en convaincre à propos de la traduction de la préposition anglaise "at" vers le Japonais, où plusieurs solutions comme "ni", "de" ou "wo" sont possibles.

Il n'est donc pas adéquat de traduire n'importe quelle phrase à pivot source par une phrase à pivot cible qui aurait "**un**" des pivots correspondants. Il est possible que certaines expressions pivot ne se traduisent que par une seule expression pivot cible, mais ce ne sera pas le cas en général. En voici une illustration de l'anglais vers le français avec la particule "as".

Source	Cible
I saw him **as** I was going to the shop.	Je l'ai vu **alors que** j'allais au magasin.
He fought **as** hard **as** he can.	Il s'est battu **aussi** fort **qu'**il a pu.
He loves her **as** she loves him.	Il l'aime **autant qu'**elle l'aime.
The baby laughs **as** his parents do.	Le bébé rit **comme** ses parents.

Les heuristiques de restriction, et le choix de l'expression pivot adéquate seront expliqués plus loin. Notre but ici est de présenter SIRD seulement d'un point de vue structurel, et cette introduction aux expressions pivot n'est qu'indicative. Une première idée est de conserver pour une expression pivot l'ensemble des premières et secondes parties "interchangeables" en mémoire, et de n'utiliser que ces expressions stockées.

[6] Espagnol cubain

8.2.2 Schéma d'expression pivot

Introduction

Une expression pivot apparaîtra à deux endroits. D'une part, des *schémas d'expressions pivots potentielles{*xe "schémas d'expressions pivots potentielles"*}* seront stockés en mémoire en tant que références. D'autre part, lors de l'analyse du segment, une *expression pivot source concrète{*xe "expression pivot source concrète"*}* sera éventuellement trouvée dans le segment, à laquelle correspondront un ou plusieurs schémas d'expression pivot potentiels stockés en mémoire. L'application de ces schémas pivots au segment courant donnera un ensemble de *nœuds d'expression pivot{*xe "nœud d'expression pivot"*}*.

La différence entre un schéma potentiel en mémoire et un schéma concret pour un segment donné, est que le schéma de la mémoire est un enregistrement unique dans lequel un certain nombre de contraintes sont indiquées, alors que l'implémentation est un double nœud faisant partie à la fois du treillis source et du treillis cible du segment représenté. On se reportera aux exemples suivants pour une illustration de ceci.

Expression potentielle en mémoire

L'expression pivot en tant que schéma potentiel en mémoire, est représentée par un schéma d'expression pivot. Ce schéma en mémoire spécifie la correspondance entre l'expression pivot source et l'expression pivot cible, la correspondance entre les quatre parties sources A, B, et cibles A' et B', et un certain nombre de contraintes sur les éléments de ces quatre parties. L'enregistrement logique d'un schéma pivot en mémoire peut être décrit comme suit :

Algorithme 9 : Schéma d'expression pivot en mémoire

```
Pivot {
      (identificateur)
      (expression pivot source)
      (expression pivot cible)
      {correspondance A, B, A' et B'}
      {conditions sur la partie A source}
      {conditions sur la partie B source}
      {conditions sur la partie A' source}
      {conditions sur la partie B' source}
      {condition sur les correspondances entre mots sources et cibles éventuellement
        obligatoires}
           }
```

Exemple :

```
Pivot {
      (p001)
      (because)
      (car)
      {A : A' ; B : B'}
      {il existe au moins un verbe V1 dans A}
      {il existe au moins un verbe V2 dans B }
      {il existe au moins un verbe V1'dans A' }
      {il existe au moins un verbe V2'dans B' }
      {V1 : V1' ; V2 : V2'}
           }
```

Exemple d'application :

He called **because** he was worried about her. Il a appelé **car** il s'inquiétait pour elle.

L'expression pivot stockée en mémoire représentée ci-dessus permettra de faire correspondre des couples de phrases anglaises et françaises du type présenté.

Expression réelle au niveau du treillis

Nous verrons plus précisément plus loin que l'utilisation d'une expression pivot se fait en quatre étapes principales :

- recherche d'un ensemble expressions pivot potentielles
- évaluation de la meilleure des expressions
- insertion dans la partie source (analyse)
- application de l'expression pivot cible (transfert)

L'utilisation d'un schéma pivot stocké en mémoire fait appel à la fois à la structure source et à la structure cible. La phrase pivot sera représentée par deux chaînes de nœuds, une source et une cible. Chaque chaîne comporte trois nœuds (au stade de notre description) : le nœud de la première partie, le nœud pour l'expression pivot, et le nœud pour la troisième partie. Ces nœuds sont placés à l'étage 6, et sont uniques sur leur étage : un seul pivot est permis par segment à ce stade de la description (une généralisation sera proposée plus loin).

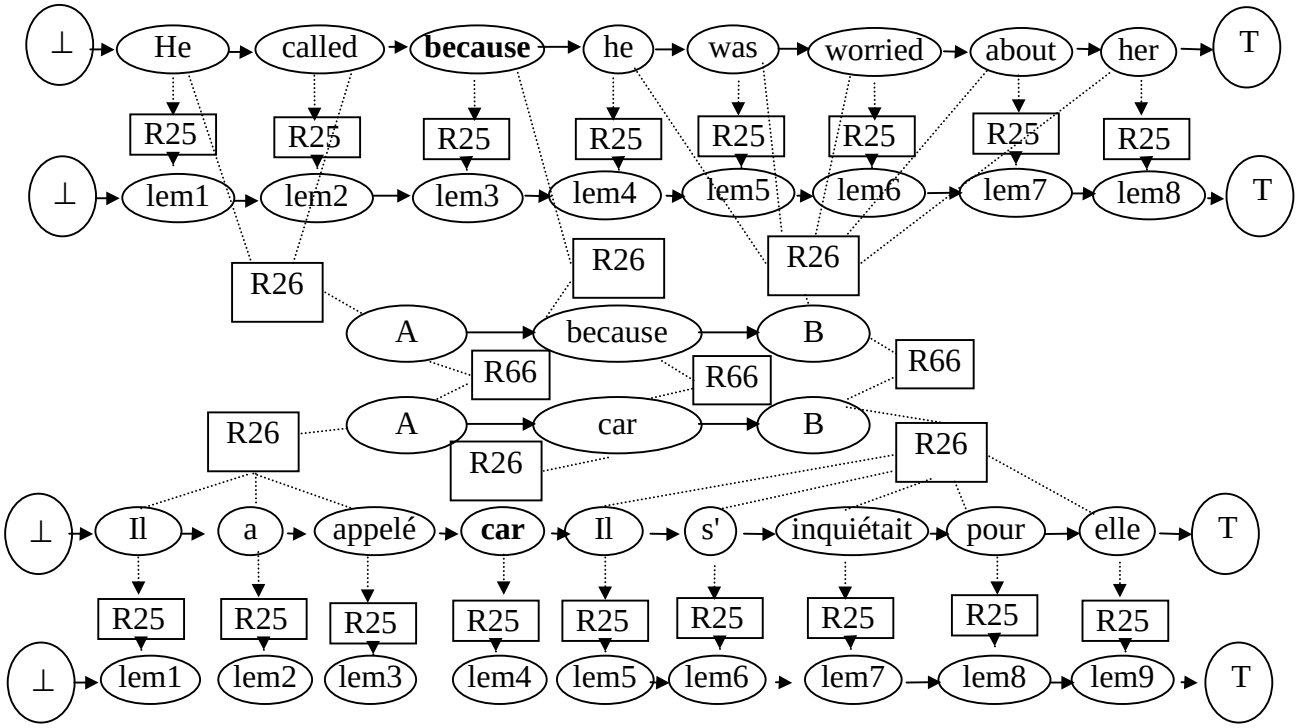


Figure 16 : Les nœuds d'expression pivots d'une expression pivot concrète

L'exemple ci-dessus donne un treillis dont voici représentés les étages 3, 5 et 6 de l'anglais et du français, sans les ambiguïtés d'analyse.

L'ensemble des mots (ou des lemmes, suivant le degré d'implémentation) formant l'une des parties (A par exemple) est spécifié par l'attachement à cette partie d'une liaison**{**xe "liaison"**}** d'un type particulier. Les liens obligatoires indiqués dans le schéma d'expression pivot de la mémoire sont implémentés par d'autres liaisons vers les mots concernés. Si certains nœuds cibles de mots obligatoires n'existent pas encore, il sont créés à ce moment là, et les contraintes linguistiques y sont appliquées. Voici maintenant un treillis complet avec un "dernier" étage des schémas pivots.

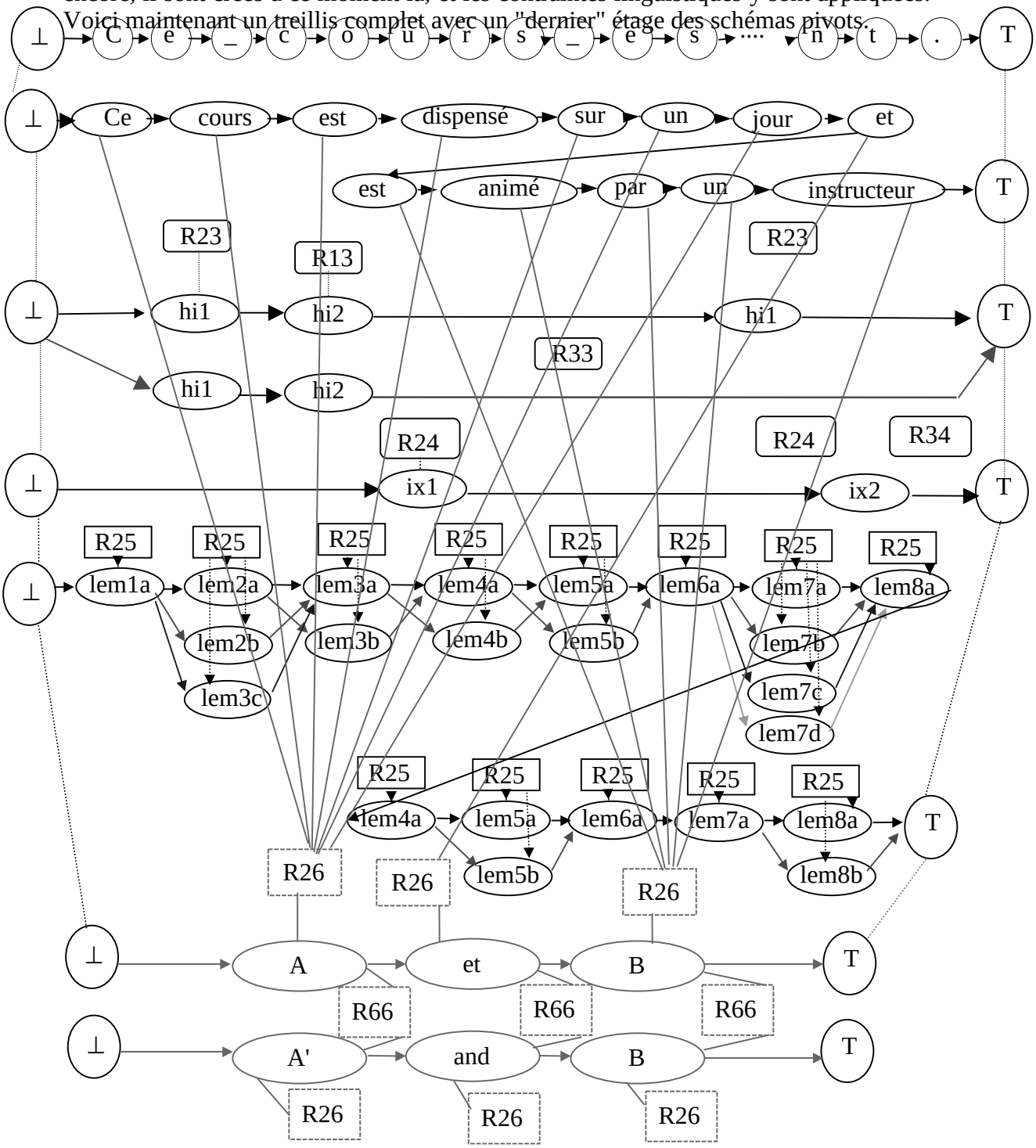


Figure 17 : Exemple d'insertion d'un schéma d'expression pivot

On notera que pour une meilleure visibilité les liens de relations entre étages inférieurs à 6 ont été cachés. A part l'étage 6, la partie cible, qui a un aspect identique, n'est pas représentée.

Remarque :

- La correspondance entre A et A' d'une part, et B et B' d'autre part est parfois croisée, comme dans cet exemple avec le japonais :

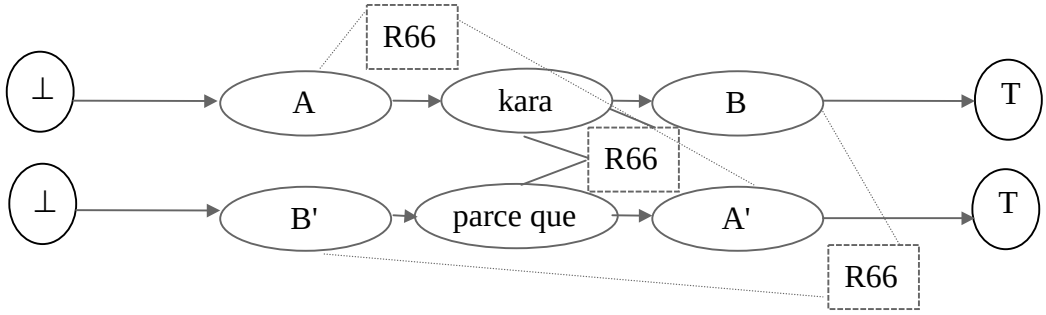


Figure 18 : Expression pivot croisée

- Les nœuds d'expressions pivots sont soit liés avec les nœuds de mots, soit avec les nœuds de lemmes, suivant la spécification des schémas d'expression pivot en mémoire. Il est évident qu'une liaison avec les lemmes est plus générique. Cependant elle demande plus de connaissance linguistique pour les nœuds.

8.2.3 Type de nœud pour l'étage 6

La description ci-dessus est formalisée par la spécification suivante de la structure d'un nœud de type t6 pour l'étage 6, source ou cible.

Algorithme 10 : Type de nœud t6 adapté à l'étage 6.

```
Nœud t6{
      (estampille d'occurrence dans la structure complète)
      (rangT du nœud à l'étage 6 parmi l'ensemble des nœuds de l'étage)
      [(partie du pivot (source ou cible)
            {conditions sur la partie}]
      (liste des arcs entrants)
      (liste des arcs sortants)
      (liste des relations liées à ce nœud)
}
```

Remarques :

Dans la phase actuelle de notre description, un seul pivot est autorisé à l'étage 6. Cela donne trois nœuds à l'étage 6. Il est envisageable d'étendre cette notion à d'autres types de "schémas" de traduction qui feraient éventuellement intervenir plus de trois nœuds.

Des liaisons doivent lier les nœuds de mots relatifs à la partie A, ou B comme résultat de l'analyse du segment source.

De même, une liaison par nœud de mot cible doit lier ce mot à la partie correspondante, comme résultat de la génération.

- la correspondance entre nœuds sources et cibles est aussi exprimée par des liaisons d'un type particulier.

8.3 Calcul

Le calcul détaillé, en particulier en ce qui concerne la recherche et le choix de l'expression pivot adéquate sera donné dans la troisième partie. Nous en donnons ici les grandes lignes.

Algorithme 11 : Création du nœud pivot

```
Début
    Construire la chaîne des mots L du segment à traiter, comme concaténation
     de l'ensemble des mots des nœuds du deuxième étage.
    Si (un groupe d'expressions pivots de la mémoire peut s'appliquer à L)
        Calculer la meilleure expression cible
        Créer les nœuds pivot source pour la meilleure expression
            instancier les enregistrements de ces nœuds
        Générer les nœuds pivots cibles par référence à l'expression en mémoire
            instancier les enregistrements de ces nœuds
        Créer les nœuds de lemmes et mots correspondants à A' et B'
        Créer les liaisons entre les deux nœuds pivots sources et cibles
        Créer les liaisons entre les deux nœuds pivots et les autres
    Fin Si
Fin
```

8.4 Correspondance entre les étages 6 et inférieurs

Elle est à cet étage exclusivement gérée par les liaisons.

8.5 Lien entre source et cible pour l'étage 6

De même, ce sont les liaisons qui gèrent ces liens.

9. Autres étages

La liste des étages proposée n'est pas fermée. On pourra être amené à construire d'autres étages pour des informations d'un autre type, par exemple ontologiques ou terminologiques.

9.1 Ontologie

La description des nœuds précédente a inclus les informations sémantiques dans les nœuds de lemmes. Une classification ontologique requiert une analyse de niveau supérieur à celui d'une classification sémantique classique.

Cela pourrait conduire à un étage important dans lequel on puisse préciser une partie ou la totalité d'une ontologie sous-jacente à la sémantique des mots. La structure de treillis permet à ce titre de représenter effectivement une classification ou ontologie. Dans ce cas, les classes ontologiques seraient disposées en nœuds sur un treillis et formeraient donc un nouvel étage. Les nœuds de cet étage seraient reliés aux lemmes par des liaisons.

9.2 Les données de la base terminologique

Les Outils de Traduction Fondés sur la Mémoire sont souvent couplés à des outils de gestion de bases terminologiques. Ces bases terminologiques, comme Multiterm ou TermStar qui ne possèdent pas toujours la rigueur des dictionnaires ou lexiques, possèdent pourtant souvent des informations utiles pour la traduction, comme des listes d'équivalents techniques ou des termes à choisir par défaut. Intégrer cette information au sein de notre structure interne peut permettre un meilleur traitement.

Ainsi un treillis supplémentaire pourrait être créé, où chaque nœud représenterait un terme, ou un domaine peut être, relatif à un mot ou un lemme du segment traité. Les liens avec les mots, lemmes ou sèmes seraient concrétisés par des liaisons entre nœuds. Cela donnerait un type de nœud t7 :

Algorithme 12 : Type de nœud t7 adapté à l'étage 6.

```
Nœud t5{
    (estampille d'identification au niveau de la structure complète)
    (rangT du nœud à l'étage 7 parmi l'ensemble des nœuds de l'étage 7)
    [ terme,
        attributs terminologiques ]
        (liste des arcs entrants)
        (liste des arcs sortants)
        (liste des relations liées à ce nœud)
}
```

10. Vue globale de la structure interne: un ensemble de treillis étagés liés

10.1 Vue d'ensemble de la structure

La structure interne se présente donc comme un segment XML plus un ensemble de treillis liés. Un segment XML source et un ensemble de treillis est affecté à la partie source, un segment XML cible et ensemble de treillis à la partie cible. Dans la cas ou cette structure est utilisée pour représenter un segment de document à traduire (voir un exemple suivant en (b)), la partie source est construite par analyse du segment, et la partie cible par transfert, via un bi-segment de traduction provenant de la mémoire (cf. partie 3). Pour utiliser ce bi-segment de traduction au format TMX, ce bi-segment est lui-même analysé (pour les deux parties sources et cibles cette fois) en une structure interne (voir un exemple en (c)).

Un treillis est composé de nœuds et d'arcs et forme un "étage". Les nœuds sont de différents types, un par étage. Les arcs, orientés, tous de même description formelle, sont étiquetés par un "type" et un "poids". Le type d'un arc (un entier) permet de différentier les chemins. Les poids permettent de donner la préférence à tel ou tel chemin. Des liaisons permettent de décrire les liens entre nœuds d'un même treillis ou de treillis (et donc étages) différents. Ces relations servent aussi à exprimer les correspondances entre les parties sources et cibles.

Rappelons que le but de ce Chapitre 5 est de définir la structure de façon générale. Nous essayons de montrer pour quelle fonction telle ou telle partie peut être utilisée, mais l'utilisation précise de cette structure pour la traduction basée sur la mémoire n'est abordé que de façon indicative. Pour une description complète des mécanisme de traduction, le lecteur devra se reporter à la troisième partie de ce document.

10.2 Exemple de structure construite à partir d'un segment XML de document.

Nous détaillons ici comment représenter un segment de traduction source et sa traduction cible à l'aide d'une structure telle que nous venons de la décrire. Nous ne montrerons cependant pas les mécanismes de traduction eux-mêmes. Le segment représente une phrase courte mais possédant une expression pivot, de façon à pouvoir montrer en perspective sur la même page le treillis source et le treillis cible, ainsi que les liens qui relient certains de leurs nœuds. La source est anglaise, la cible française. Soit donc un segment source et cible, d'un fichier Microsoft RTF :

source	cible
Click a **color** and press ENTER.	Cliquer sur une **couleur** et appuyez sur ENTREE.

Les marques d'index placées après "color" et "couleur" sont visibles dans le code RTF.

```
… {¥lang1033 Click a }{¥b¥lang1033 color}{¥pard¥plain ¥widctlpar ¥v¥f4¥lang1033 {¥xe {}{¥b¥lang1033 color}{}}}{¥lang1033 and press ENTER}
```

La transcription en XML donne :

```
<s> <? SPELL=ENG ?> Click_a_ <GRP> <? TERM=SPELL ?> <hi1> color_</hi1> <MOBJ attrib =index/> <? TERM=SPELL ?>and_ press_ ENTER. </s>
```

Pour simplifier la représentation, nous allons factoriser l'instruction de traitement pour le correcteur orthographique indiquant que les termes sont des termes anglais. Cela donne :

<s> <? SPELL=ENG ?> Click_a_ <hi1> color_ </hi1> <MOBJ attrib =index/> and_ press_ ENTER.</s>

De même, on obtient la représentation XML suivante pour le segment cible, et le double treillis correspondant:

<s> <? SPELL=FRE ?>Cliquer sur une <hi1>couleur </hi1> <MOBJ attrib =index/>et appuyez sur ENTREE.</s>

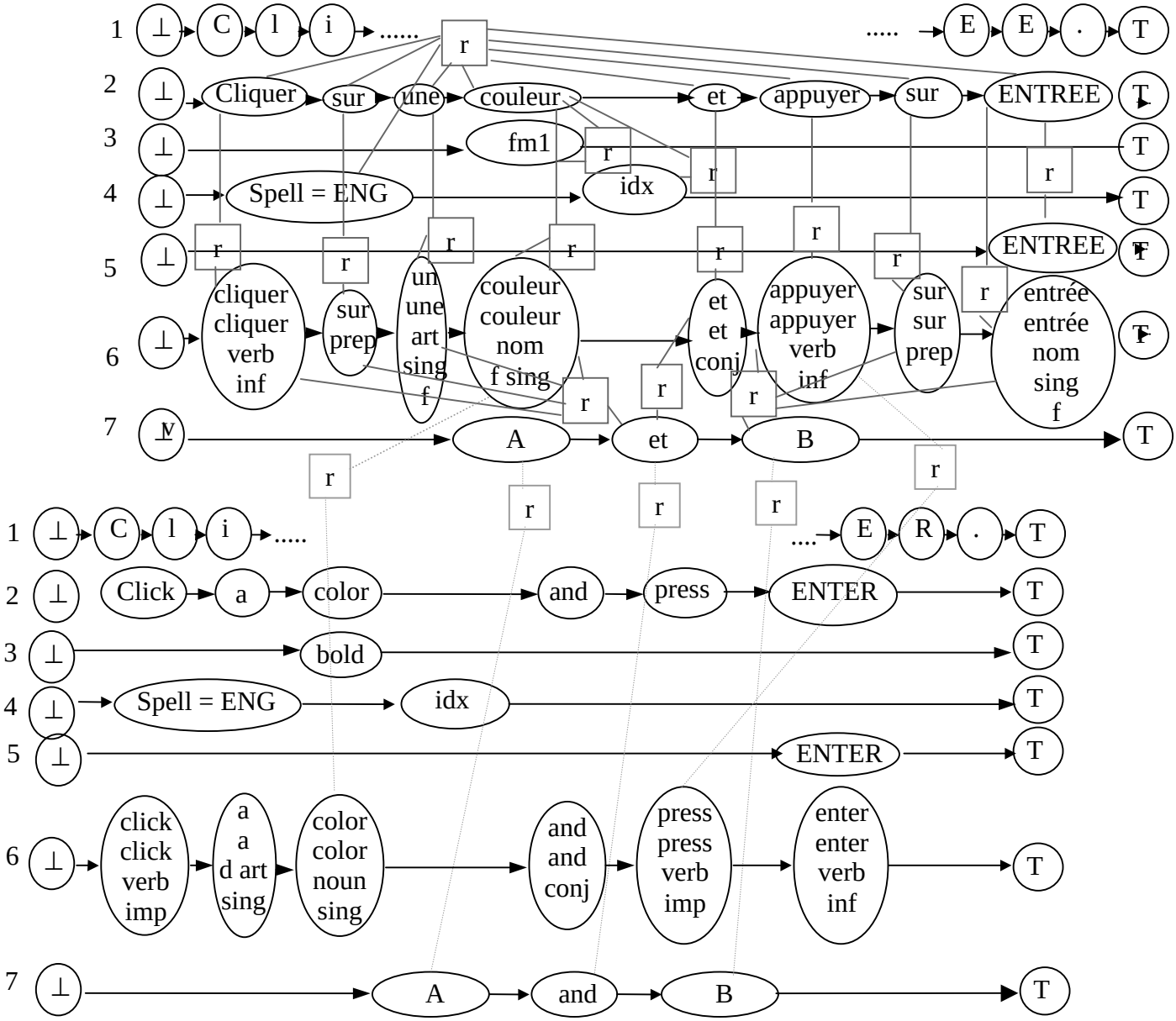


Figure 19 : Représentation d'une phrase et de sa traduction

Remarques :

- "ENTREE" est traité de façon particulière. En effet, ce terme se reporte à une touche du clavier. En tant que terme interface du logiciel à traduire (ou plus

exactement d'une partie de l'ordinateur qui est le support du logiciel que l'on traduit), il doit être géré. Dans un processus de localisation classique, le client et le traducteur s'entendent sur les termes de l'interface à employer. Cela signifie qu'une liste de termes obligatoires est dressée. Dans ce cas "ENTER" sera donc traduit par "ENTREE". Un étage supplémentaire est ici créé pour représenter les termes venant de la base terminologique qui contiennent les termes comme "ENTREE".

ENTREE est aussi considéré comme un objet (étage 4), qui sera traduit par référence à une liste. Noter qu'à différents objets, différentes méthodes de traduction peuvent s'appliquer. Ainsi, si cet objet est une icône, on peut vouloir l'adapter à la culture source. Dans ce cas, une procédure spéciale décrite dans [Semmar, Fluhr et Planas 1997] est à appliquer.

- Les relations internes à la partie source sont représentées. Les relations internes à la partie cible ne le sont pas. Certaines relations (en pointillés) entre la source et le cible sont représentées.
- Pour simplifier ce schéma, seuls des treillis réduits à une liste ont été utilisés. On a en particulier omis ici les ambiguïtés de lemmatisation.

10.3 Exemple de structure construite à partir d'une unité de mémoire TMX

Nous montrons maintenant la représentation des deux bi-segments de la mémoire utilisés pour traduire la phrase à pivot représentée ci-dessus.

source	cible
Click a color	Cliquer sur une couleur
press ENTER	appuyez sur ENTREE

Aucun formatage n'est indiqué en mémoire ici. La représentation TMX de ces deux unités de traduction est donnée ci-après :

```
<TU ID="0001">
        <TUV LANG="EN">
          <s>Click a color</s>
    </TUV>
    <TUV LANG="FR-FR">
        <s>Cliquer sur une couleur</s>
    </TUV>
</TU>
<TU ID="0001">
    <TUV LANG="EN">
        <s>press ENTER</s>
    </TUV>
    <TUV LANG="FR-FR">
        <s>appuyez sur ENTREE</s>
    </TUV>
</TU>
```

On notera que c'est la partie du fichier TMX correspondant aux unités de traduction. Cette représentation étant faite sous TMX, elle est acceptée par XML, et aucune conversion n'est à réaliser. L'extraction des segments donne les simples chaînes suivantes :

```
<s>Click a color</s>
<s>Cliquer sur une couleur</s>
et
<s>press ENTER</s>
<s>appuyez sur ENTREE</s>
```

Leur représentation sous forme interne donne les deux doubles treillis suivants, dans lesquels il n'y a pas de troisième et quatrième étage, car il n'y a pas de balise à représenter.

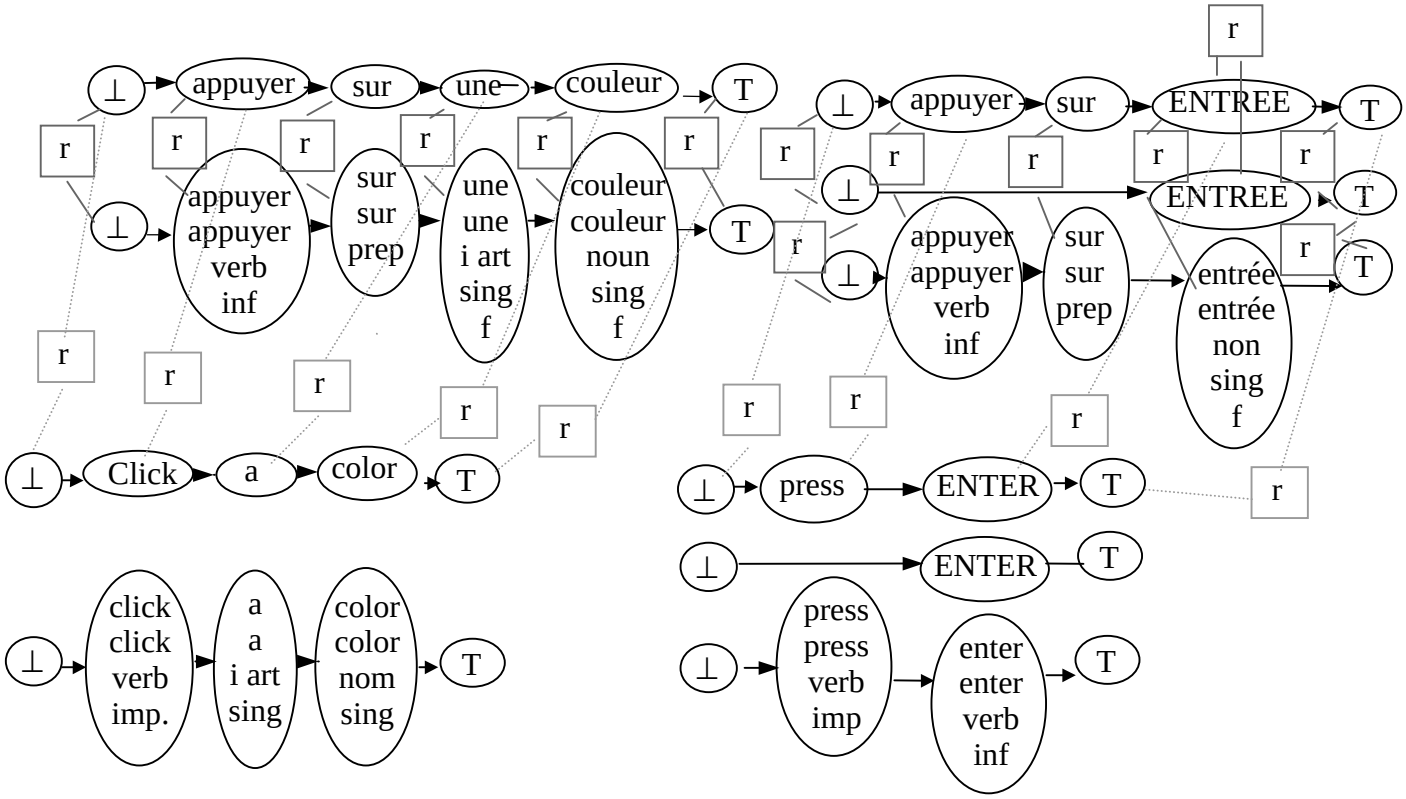


Figure 20 : Représentation des unités de mémoires

Remarques :

- Certaines relations ont été omises pour la clarté du schéma
- Des relations lient les mots sources et cibles correspondants de façon systématique ici. Il peut arriver que seulement quelques relations aient été trouvées au niveau des mots, et d'autres au niveau des lemmes (comme dans le schéma précédent)

Conclusion

Nous venons donc de montrer ce que doit représenter une structure de type TELA au niveau de l'information provenant d'un segment XML. Sept étages ont ainsi été proposés. La notion d'expression pivot a été introduite à cette occasion.

Chapitre 6

TELA : une formalisation des structures

Contenu du Chapitre

Introduction

Le chapitre précédent a concrétisé le choix d'une structure interne de représentation des données basée sur la donnée de deux objets :

- un segment XML (la structure primaire)
- ensemble de treillis liés (la structure analytique)

Il a été montré comment représenter concrètement l'information relative à la partie primaire par la partie analylique. Le présent chapitre formalise et généralise cette structure que nous désignerons désormais par l'acronyme "TELA" pour "Treillis Etagés et Liés pour traitements Automatisés".

Le premier paragraphe propose une définition mathématique des structures de données et des opérations de TELA. Le second propose une implémentation abstraite pour manipuler plus aisément les données portées par TELA. Dans le troisième paragraphe, sont donnés un langage externe de spécification basé sur XML, et un langage de spécification objet de TELA.

1. Définition mathématique de TELA

1.1 Structure de données

1.1.1 Treillis et relations de correspondance

Définition 1 : Treillis

Un **treillis** est un ensemble ordonné d'éléments E tels que deux éléments quelconques admettent une borne inférieure (notée "inf") et une borne supérieure (notée "sup").

Définition 2 : Relation de correspondance et liaisons

Nous appelons **relation de correspondance** entre deux ensembles E et F une relation de P(E) dans P(F), c'est-à-dire un sous ensemble fini C du produit cartésien P(E) x P(F) de l'ensemble des parties non vides de E par l'ensemble des parties non vides de F. Nous dénommerons chaque élément de C une **liaison**.

Remarque :

Nous considérons le plus souvent par la suite que E = F = l'ensemble des treillis de TELA.

1.1.2 Représentation graphique

Les treillis accompagnés de leurs relations de correspondance peuvent être représentées à l'aide de deux entités : la première est un ensemble de *nœuds de structure* connectés par des *arcs orientés typés* qui représentent les éléments du treillis, l'autre est un ensemble de *nœuds de relations de correspondances* et de *liens orientés* qui matérialisent les relations de correspondances. Les nœuds peuvent porter des *étiquettes*. Les orientations des arcs rappellent l'ordre partiel que vérifient les éléments des treillis. Des *types* d'arcs (des couleurs par exemple) permettent de distinguer des chemins particuliers dans les treillis.

Voici un exemple de la représentation de deux treillis liés par une relation de correspondance construite à partir de quatre liaisons.

Exemple :

Soit deux treillis E_1 et E_2 possédant chacun cinq éléments :

E_1 = {inf1, e11, e12, e13, sup1}

E_2 = {inf2, e21, e22, e23, sup2}

Les notations "inf1", "inf2", "sup1", et "sup2" dénotent respectivement les bornes inférieures de E_1 et E_2, et les bornes supérieures de E_1 et E_2. L'ordre sur chacun des treillis est explicité par un schéma aux conventions décrites ci-dessus ; par exemple e11 < e12 car il existe un arc entre les nœuds représentant e11 et e12, orienté de e11 vers e12[1].

Considérons maintenant l'ensemble $E = E_1 \cup E_2$, et soit P(E) x P(F) le produit cartésien des parties finies non nulles de E par ce même ensemble. Alors le sous-ensemble suivant C de $E^+ \times E^+$ définit une relation de correspondance sur E :

C = {[{inf1}, {inf2}], [{sup1}, {sup2}], [{e11, e12}, {e13}], [{e12}, {e22, e23}]}

Cette relation de correspondance est composée de quatre *liaisons* exprimées entre crochets ci-dessus, et représentées par un nœud (de forme rectangulaire) lié aux nœuds participants à la relation par des arcs en pointillé.

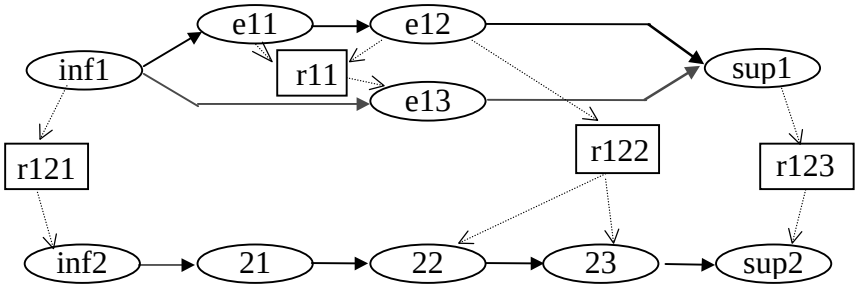


Figure 1 : Représentation de deux treillis et d'une correspondance sur l'ensemble des deux treillis

Notation :

[1] Par abus de notation, nous exprimerons désormais les noeuds du graphique et les éléments du treillis avec la même notation (ex : e11).

Pour un tel ensemble de treillis, chaque treillis sera aussi appelé "étage". Dans cet exemple, deux treillis forment les deux "*étages*" de l'ensemble E.

Remarque :

- On peut voir les relations de correspondances comme un ensemble fini de liaisons entre un ensemble fini d'éléments d'entrée, et un ensemble fini d'éléments de sortie : r = $\{\{\{e^p_{11}, e^p_{12}, .., e^p_{1m_p}\}, (\{e^p_{21}, e^p_{22},...,e^p_{2n_p}\}\}_{p\in P}\}$
- Les relations de correspondances définissent un type de correspondance local concernant les éléments de E entre eux uniquement.

1.1.3 Echelles de mesures linéaire

Nous introduisons maintenant une notion formalisant la "séquentialité" vue aux chapitres 4 et 5 : les échelles de mesures linéaires. Elles permettront d'exprimer "l'alignement" des éléments du treillis par exemple sur une échelle d'occurrence dans un parcours gauche droite des éléments du segment XML.

Définition 3 : Echelle de mesure linéaire

Soit E un ensemble, et D un ensemble discret muni d'une relation d'ordre total notée "<". Nous appellerons *échelle de mesure linéaire* une application τ de E vers D x D, qui à un élément de E associe un couple de D x D tel que : $\tau : E \rightarrow D \times D$, avec $\tau(e) = (d1, d2)$ tel que $d1 < d2$.

Exemple :

Soit D = N, et $E = E_1 \cup E_2$ vus dans l'exemple précédent. L'ensemble suivant définit une échelle de mesure de mesure linéaire N sur E :

τ(inf1) = (0,0), τ(e11) = (1,1), τ(e12) = (2,4), τ(e13) = (5,7), τ(sup1) = (8,8),
τ(inf2) = (0,0), τ(e21) = (2,2), τ(e22) = (3,4) τ(e23) = (5,7), τ(sup2) = (8,8).

La représentation graphique de cette échelle peut se faire à l'aide d'un axe associé au graphe précédent. Les nœuds sont alors positionnés selon leur échelle linéaire :

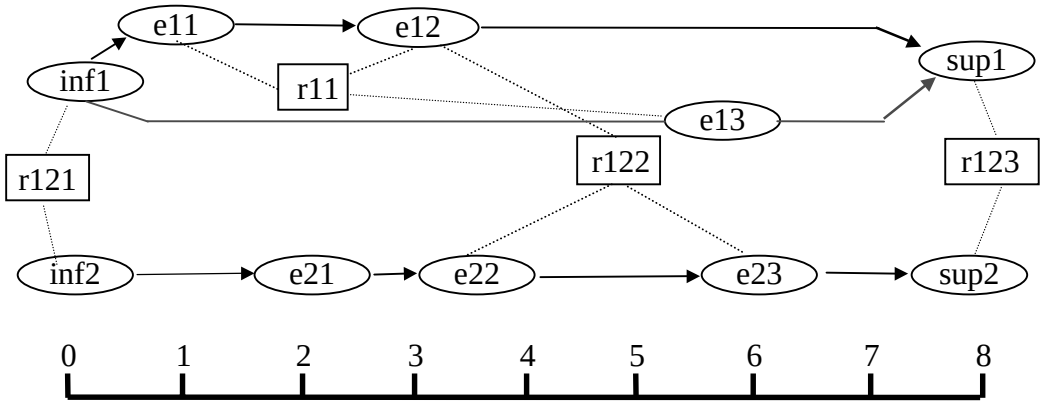


Figure 2 : Echelle de mesure linéaire sur deux treillis.

Remarque :

- Les deux coordonnées d1 et d2 correspondent à un "début" et une "fin". Si l'on considère l'axe comme temporel, alors τ peut être vu comme une durée entre deux temps t1 et t2.

- L'image (d1, d2) d'un élément par une échelle de mesure linéaire peut être telle que d1=d2. Dans ce cas l'élément n'a pas de "durée", et correspond à un instant.
- D'autres métaphores sont applicables à τ. On peut par exemple penser à l'échelle des positions d'éléments par rapport à celle des caractères d'un segment XML, comme dans le chapitre 5.

1.1.4 Décorations

Définition 4 : Décoration

Une décoration est une application δ de l'ensemble des éléments de E vers un ensemble S

$\delta : E \rightarrow S$

Les éléments de S sont considérés comme des éléments structurels.

L'ensemble S des décorations peut être de nature diverse.

Exemple :

Soient :

- E l'ensemble des nœuds vus aux exemples précédents,
- $S = C \times A^+ \times \{0, 1\}$, où C = {indigo, bleu, vert, jaune, orange, rouge, violet}, et A = {a, b, c, ..., z}.

C représente donc une des couleurs de l'arc-en-ciel, A^+ représente l'ensemble des mots (au sens large).

Alors [$\delta : E \rightarrow S$] associe à chaque élément de e un triplet (c, n, b) constitué d'une couleur, d'un nom, et d'une caractéristique binaire (vrai ou faux par exemple).

1.1.5 Structure TELA

Définition 5 : structure TELA

Une structure TELA est la donnée des éléments suivants :

- un ensemble de treillis $E = \{E_i\} i \in I$
- dont les éléments vérifient un ensemble de relations de correspondances $\{r_j\}$, $j \in J$
- un ensemble d'echelles $\{\tau_k\}$, $k \in K$
- un ensemble de décorations $\{\delta_l\}$, $l \in L$

1.2 Opérations sur la structure

Introduction

Nous introduisons maintenant deux opérations de base sur la structure TELA : l'ajout d'un élément à un treillis, et l'ajout d'une liaison à une relation de correspondance.

1.2.1 Ajout d'un élément à un treillis

Définition 6 : ajout d'un élément au treillis

Soit un élément E d'un ensemble. Soit e un nouvel élément. Alors son ajout au treillis E consiste en :

- l'union $E \cup \{e\}$
- la définition d'un ordre partiel sur e dans E

Exemple :

Reprenons l'exemple précédent où $E = E = E_1 \cup E_2$, et E contient dix éléments. Soit e26 un troisième élément. Son ajout à E pour donner E' peut être spécifié par :

- $E' = E \cup \{e26\}$
- $e21 < e26 < e23$, $e26 \neg< e22$, $e22 \neg< e26$[2]

E' est représenté ci-dessous :

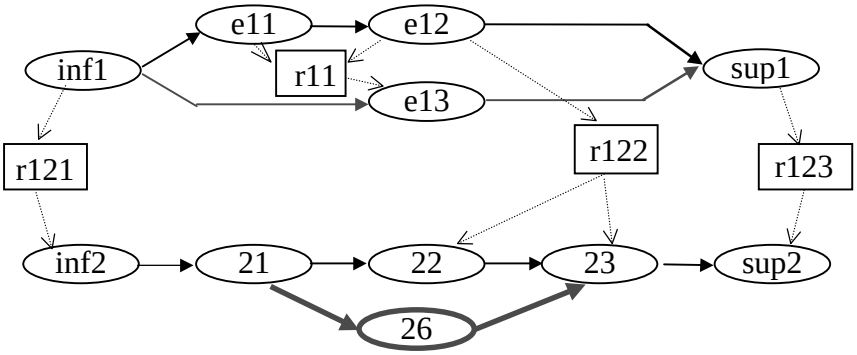


Figure 3 : Ajout de l'élément e26 au treillis E : E'

1.2.2 Ajout d'une liaison à la relation de correspondance

Une autre opération importante consiste en l'ajout d'une liaison à la relation de correspondance liant les nœuds d'une structure TELA.

Définition 7 : ajout d'une liaison à une relation de correspondance

Soit un treillis E. Soit C une relation de correspondance sur E. Soit maintenant un élément l de P(E) x P(F). L'ajout de la liaison l à C consiste en la définition de C' tel que :

$C' = C \cup \{l\}$

Reprenons l'exemple précédent, dans lequel la relation C vérifiait :

C = {[{inf1}, {inf2}], [{sup1}, {sup2}], [{e11, e12}, (e13)], [{e12}, {e22, e23}]}

[2] $\neg<$ signifie non inférieur

Considérons cette même relation sur E' et soit maintenant la liaison :

r124 = [{e21, e22}, {e26}] ∈ P(E) x P(F)

Alors l'ajout de la liaison l à C donne C' tel que :

C' = C ∪ r124 = {[{inf1}, {inf2}], [{sup1}, {sup2}], [{e11, e12}, {e13}], [{e12}, {e22, e23}], [{e21, e22}, {e26}]}

Voici maintenant une représentation de C' sur E' :

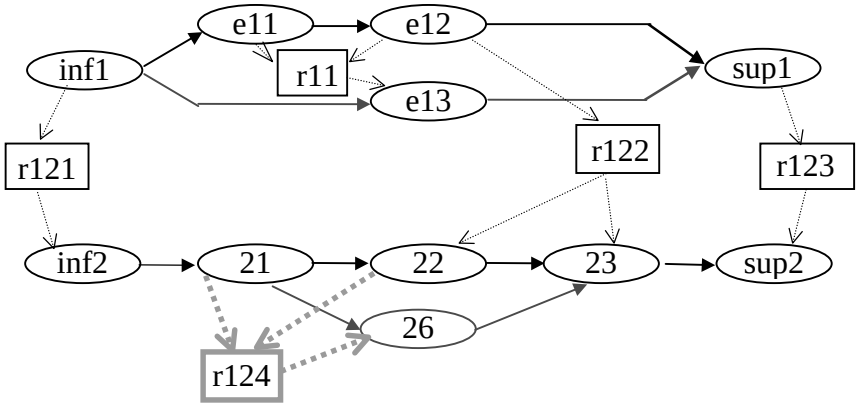


Figure 4 : ajout de la liaison r124 à la relation de correspondance C : C'

1.3 Lien avec LEAF

Une structure similaire, LEAF, a été proposée dans [Lafourcade 93]. LEAF (Lattices Engine And Features) est un modèle de représentation et de manipulation d'objets linguistiques mettant en jeu des treillis, des moteurs et des décorations. Dans TELA seuls les porteurs d'information (les treillis et les structures de données) sont exprimés.

Dans TELA, la manipulation de la structure n'est pas contrainte, et est laissée libre aux algorithmes qui ne respectent aucun formalisme pré-contraint. Dans la structure de données de LEAF, les étages sont appelés des strates et correspondent à des treillis de nœuds. Les liaisons sont appelées "nœuds de règles".

Les nœuds des treillis de LEAF portent une étiquette et une décoration tout comme dans TELA. Un couple de bornes inférieure et supérieure permet de situer le nœud sur un espace temps. Dans TELA, cela est pris en compte par les échelles de mesures linéaires. Les arcs de LEAF portent une valuation. Ceux de TELA peuvent porter à la fois un type qui permet une sélection de chemins a priori, et un poids (discret ou continu) qui permet une sélection de chemin lors du calcul (discret ou continu).

Le modèle LEAF définit des "règles" permettant de formaliser la construction de nœuds à partir d'autres. Cela correspond dans TELA aux "liaisons". Nous retrouverons cette notion avec les "schémas de relations" qui sont introduits dans le paragraphe suivant.

2. Une description informatique générale

Introduction

Un formalisme proche de l'implémentation d'une telle structure est maintenant proposé, qui se veut assez général pour ne pas compromettre les choix d'implémentation.

2.1 Structure de données

2.1.1 Nœuds de structure

Un élément d'un des treillis de TELA est donc vu comme un nœud . Nous garderons maintenant cette terminologie. Pour sa manipulation algorithmique, un nœud possède les informations suivantes.

Un identificateur unique
Il identifie formellement le nœud en définissant une bijection entre l'ensemble des identificateurs et l'ensemble des nœuds. C'est un **entier**.

Un type de nœud
Cela permet par exemple de distinguer les nœuds des différents treillis qui peuvent représenter concrètement différents objets (des caractères ou des lemmes par exemple). Le type d'un nœud est un **entier**.

Un ensemble de "coordonnées" (définies comme les images d'échelles linéaires)
Les échelles servent à repérer le nœud par rapport à telle ou telle structure de référence particulière, interne ou externe. La notion de structure de référence est proche de celle des référentiels de la dynamique : un même objet peut être vu différemment sous tel ou tel référentiel, et donc avoir des coordonnées différentes.

L'élément e de l'ensemble E que représente le nœud admet par une échelle linéaire τ une image dans un ensemble D x D. L'ensemble D x D peut alors être vu comme les axes d'un système de coordonnées qu'est l'échelle linéaire τ. Les coordonnées du nœud se présentent sous la forme d'un couple d'entiers. Il y a, a priori, autant de coordonnées possibles pour un nœud que de référentiels, donnant donc un **ensemble de couples d'entiers** par nœud.

Une structure de données (comme image d'une décoration)
Une décoration δ associe à un élément du treillis (représenté par un nœud), une structurede l'ensemble S. Cette structurepermet de stocker l'information que porte le nœud . En toute généralité cette structure peut être simple ou multiple. Au niveau de l'implémentation, cela peut être une chaîne de caractères, un entier, une liste, un tableau ou un arbre. Si l'on se réfère au chapitre précédent, pour un nœud de caractères, l'information sera simple : le caractère. Pour un nœud de lemme, elle comportera tous les enregistrements qui caractérisent un lemme. Dans le cas de la reconnaissance vocale, les enregistrements digitaux pouvant être caractérisés par plusieurs axes (volume, profil, longueur d'ondes couvertes, etc.), et ceci demanderait donc une structure multiple, alors qu'une syllabe ne demandera plutôt qu'une chaîne de caractères pour la représenter.

L'ensemble des arcs de structure connectés à un nœud .
Au sein du treillis, chaque nœud est immédiatement précédé (au sens de la relation d'ordre partiel) par un ensemble de nœuds, et est immédiatement suivi par un autre ensemble de nœuds. Leur ordre (de haut en bas sur le schéma), qui peut être induit par un ordre sur les types d'arcs peut être significatif, et devra donc être gardé par des listes de nœuds plutôt que des ensembles. Il y a donc deux listes principales de nœuds au sein du nœud donné :

- **la liste des nœuds précédents**
- **la liste des nœuds suivants**

2.1.2 Arcs de structure

Un arc est considéré comme la donnée de quatre éléments :

Un identificateur unique d'arc
Celui-ci permet de référer à cet arc, de la même façon que pour les nœuds. Ce sera un **entier** pour la suite.

Le type de l'arc
Le type de l'arc permet de distinguer différentes classes d'arcs. Il correspondant à la notion de coloration de l'arc. Ainsi, au sein d'un treillis, si l'on veut distinguer plusieurs "chemins" différents, on peut employer un chemin construit avec des nœuds et un type d'arc (d'une couleur), et un autre possédant des nœuds de même type car la nature de l'objet représenté reste le même, mais des arcs d'un type différent (une autre couleur). Le type de l'arc est une notion discontinue (un **entier** pour la suite).

Le poids de l'arc
Le poids de l'arc permettra une approche statistique (**réels**) ou symbolique (**entiers**) dans la résolution de problèmes basés sur cette structure. Cela peut permettre aux algorithmes manipulant une structure TELA d'être plus efficaces en leur donnant la possibilité d'utiliser des "préférences" ou des "indices".

Un nœud de départ
Le **nœud de départ** de l'arc permet de situer et d'orienter l'arc dans le treillis.

Un nœud d'arrivée
De même pour le **nœud d'arrivée**.

2.1.3 Treillis de TELA

Un treillis est formé d'un ensemble de nœuds, et d'un ensemble d'arcs. Voici le détail de ses constituants :

Un identificateur de treillis
Un identificateur de treillis permet de le manipuler au sein de la structure TELA. Cet identificateur sera une **chaîne de caractères**.

Un ensemble de nœud
Ce sont les nœud qui forment le treillis. Chacun peut avoir un type différent. Il existe toujours un nœud "**inf**", plus petit que tous les autres, et un nœud "**sup**", plus grand que tous les autres.

Un ensemble d'arcs
Ce sont ces arcs structurels qui donnent la structure de treillis. Ils permettent par le type de différencier différents chemins dans le treillis.

2.1.4 Liaison

Une relation de liaison sur un treillis E est la donnée d'un élément de P(E) x P(F). Nous considérerons une telle double suite comme la donnée :

Identificateur
Indentifie de façon unique la liaison

Liste des nœuds de départ
C'est en fait une liste d'identificateurs.

Liste des nœuds d'arrivée

Idem

Type de liaison
On peut être amené à distinguer un type de liaison pour par exemple lui donner une sémantique en fonction de l'ensemble des étages auxquels appartiennent les nœuds qui y participent. Les types de liaisons peuvent provenir d'autres besoins que nous verrons plus tard. Le type d'une liaison sera noté Rα, où α est un **entier**.

2.1.5 Relation de correspondance

Une relation de correspondance est la donnée d'un ensemble de liaisons. Elle sera donnée implicitement par la définition de l'ensemble des liaisons.

2.1.6 Structure analytique de TELA

Une structure TELA est la donnée de :

Un identificateur unique
C'est une chaîne de caractères.

Un ensemble de treillis
Ils forment la structure de base de TELA.

Un ensemble de liaisons
Cet ensemble de liaisons définit une relation de correspondances.

On notera que les ensembles d'échelles et les décorations sont portées par les nœuds eux-mêmes, et ne sont pas exprimés au niveau de la structure TELA elle-même, bien qu'au niveau formel, ce soient des applications dont le domaine de départ est l'ensemble des nœud de TELA.

2.2 Opérations de base sur la structure TELA

Nous présentons ici les opérations de base nécessaires à l'implémentation d'une structure TELA. La spécification de ces opérations reste générale, mais décrit en même temps les opérations de manipulation des structures d'implémentation abstraite.

2.2.1 Ajout d'un nœud

La création d'un nouveau nœud de treillis correspond à l'ajout d'un élément à l'un des treillis de TELA vu dans la description mathématique précédente. Au niveau de l'implémentation, cet ajout comporte les étapes suivantes. Les quatre premières étapes correspondent à la création proprement dite du nœud. Les deux dernières au placement du nœud dans le treillis qui le contient.

- donnée de son identificateur
- spécification de son type
- donnée de la suite de ses coordonnées
- remplissage de la structure de données
- création et connexion aux nœuds précédents, via les arcs de structure entrants
- création et connexion aux nœuds suivant, via les arcs de structure sortants

2.2.2 Ajout d'un arc de structure au treillis

Les arcs de structures sont ajoutés lorsque l'on veut connecter un nœud à ses différents voisins.

- un identificateur unique
- son type
- son poids
- son nœud de départ
- son nœud d'arrivée

On remarquera que l'on pourrait intégrer au nœud les données concernant les arcs, et ne pas créer d'arc en soi. Les arcs ne sont en effet que la matérialisation des précédences et successions des nœuds, à leur type et leur poids près. Cela obligerait alors soit à inscrire ces informations dans chaque nœud, soit à choisir l'un des nœuds que lie l'arc comme contenant l'information de l'arc.

2.2.3 Ajout d'une liaison

L'ajout d'une liaison consiste à définir des liens entre un ensemble de nœuds et une relation typée qui définit la liaison. Cela se fait par la spécification des éléments d'une liaison tels qu'ils ont été vus dans la définition d'une liaison précédente.

- un identificateur unique
- le type de la relation
- l'ensemble ordonné des nœuds de départ
- l'ensemble ordonné des nœuds d'arrivée

2.2.4 Application de schémas de liaison

Schéma de liaison

Un schéma de liaison S_R est la donnée des éléments suivants :

- un type de liaison : t_r
- une liste ordonnée de types i de nœuds d'entrée: (t_1^i)
- une liste ordonnée de types j de nœuds sortie: (t_2^j)
- des conditions d'entrées sur (t_1^i)
- des conditions de sorties sur (t_2^i)

L'application d'un tel schéma à une liste de nœuds d'entrée (e_1^i) permet de créer une liste de nouveaux nœuds de sortie (e_2^j) tels qu'ils vérifient ceci : si un nombre requis de nœuds d'entrée E vérifie les conditions d'entrées données par :

- les types de nœuds requis
- les conditions d'entrées

Alors les nœuds de sortie sont créés selon :

- les types de nœuds de sortie
- les conditions de sortie

Et une liaison de type t_r est créée entre ces nœuds. Les nœuds de départ de la liaison sont alors les nœuds d'entrées, et les nœuds d'arrivée, les nœuds de sortie.

Application d'un schéma de liaison

Les schémas de liaisons définissent des procédures locales de création de nœuds. Le principe consiste à appliquer ce schéma sur un ensemble de nœuds d'entrée vérifiant les conditions d'entrées du schéma, pour créer les nœuds de sortie vérifiant des conditions de sortie. En voici un exemple.

Exemple :

Soit le schéma de liaison S suivant qui joue le rôle d'une règle de transcription des caractères nœuds des caractères "& e a c u t e ;" en "é" :

```
S =  [   R12
         ((N1,t1), (N2, t1), (N3, t1), (N4, t1), (N5, t1), (N6, t1), (N7, t1),
                                                                    (N8, t1))
         ((N9, t1))
         (    N1.obtenirDonnées(1,1) = "&",
              N2.obtenirDonnées(1,1) = "e",
              N3.obtenirDonnées(1,1) = "a",
              N4.obtenirDonnées(1,1) = "c",
              N5.obtenirDonnées(1,1) = "u",
              N6.obtenirDonnées(1,1) = "t",
              N7.obtenirDonnées(1,1) = "e",
              N8.obtenirDonnées(1,1) = ";" )
         (    (N9 = "é"),
              (N1.obtenirNœud Prec, N9, ta2),
              (N9, N8.obtenirNœudsuiv, ta2))   ]
```

Nous anticipons ici sur la notation objet qui sera introduite plus loin, car elle est de compréhension immédiate et très pratique. Alors en parcourant les ensembles de huit nœuds du treillis des caractères, si les conditions sont vérifiées pour les huit nœuds, le neuvième nœud est créé et un arc de type t_{a2} est créé entre le précédent de N1 et N9, et entre N9 et le suivant de N8. Cela est illustré par l'opération de la figure 2 du chapitre 5.

2.3 Exemples

2.3.1 Exemples du chapitre 5

On pourra se reporter au chapitre 5 pour avoir une série d'exemples d'application de TELA dans le domaine de la traduction fondée sur la mémoire. Cette spécialisation de TELA est explicitée au paragraphe V, par la définition de TELAM.

2.3.2 Un exemple d'application à un autre domaine que la Traduction Fondée sur la Mémoire

Le schéma suivant montre en particulier l'utilité des procédures de liens. Il s'agit de gérer la composition de glyphes compositionnels à partir de caractères de bases comme dans les langues thai ou coréennes.

Voici plus particulièrement un exemple en thai. La première figure montre une règle de composition de caractères extraite de [Kataoka 1997]. La deuxième figure montre comment notre modèle représente cette composition de caractères.

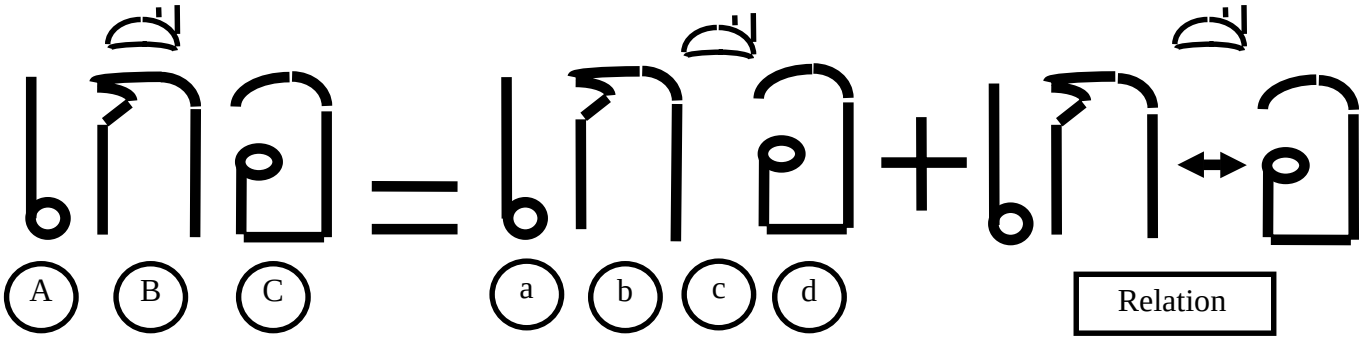


Figure 5 : Une règle de composition de caractères pour l'obtention d'un glyphe thai

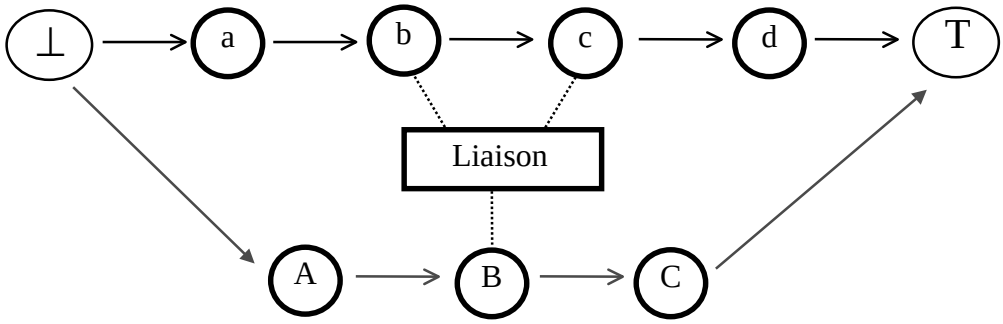


Figure 6 : La formalisation de la règle précédente

Ainsi la relation de la règle de composition présentée en Figure 5 est traduite par un schéma de liaison dans notre représentation dans notre formalisme. Ce schéma de liaison spécifie que les caractères c et d doivent être réunis pour donner la forme finale B.

La langue thai, comme la langue arabe ou l'hébreu, ont ceci de particulier que les lettres de l'alphabet (les lettres de base) se composent pour former des glyphes dans le texte. C'est un phénomène un peu comparable à la liaison du o et du e, qui donne œ en français.

3. Langages de description de TELA

Introduction

Le paragraphe précédent a donné une définition informatique générale des structures TELA. La structure **TELA** peut être ainsi vue comme un objet composé de sous-objets que sont les **treillis**, les **relations de correspondances**. Les treillis eux-mêmes sont composés de sous-objets : les **nœuds** et les **arcs**. Les relations de correspondance sont définies par un ensemble de **liaisons**.

Nous proposons maintenant un langage de description d'une structure TELA sous forme d'un document XML. Cela donnera donc une description XML de la forme analytique du segment qu'est TELA. La forme primaire de la représentation d'un segment étant définie sous la forme d'un segment XML, les deux parties de la représentation interne des données (formes primaire et analytique) pourront alors être manipulées de façon homogène sous forme de document XML.

3.1 Un langage de description de TELA basé sur XML

Introduction

La formalisation d'un document XML se faisant par la définition de son type de document (DTD), nous proposons maintenant une telle définition

3.1.1 Définition de Type de Document XML de TELA

La DTD suivante suit les conventions XML que l'on peut trouver dans [XML 1997].

Définition 8 : DTD XML de TELA

```
<!ELEMENT     Tela                (Treillis+, Relation*)          >
    <!ATTLIST Tela        id          ID                    #IMPLIED>

<!ELEMENT     Treillis            (NœudInf, NœudSup, Nœud*, Arc+)>
    <!ATTLIST Treillis    id          ID                    #IMPLIED>

<!ELEMENT     NœudInf         EMPTY                         >
    <!ATTLIST NœudInf id  ID                          #IMPLIED
                          type        NUMBER          #REQUIRED
                          coord       CDATA           #FIXED "(0, 0)">

<!ELEMENT     NœudSup             EMPTY                         >
    <!ATTLIST NœudSup     id  ID                        #IMPLIED
                          type        NUMBER          #REQUIRED
                          coord       CDATA                 "(0, 0)">

<!ELEMENT     Nœud                (Decoration)                  >
    <!ATTLIST Nœud        id          ID                #IMPLIED
                          type        NUMBER          #REQUIRED
                          coord       CDATA                 "(0, 0)">

<!ELEMENT     Decoration          (#PCDATA)                     >
    <!ATTLIST Decoration  valeur      CDATA           #IMPLIED
                          refdecor    IDREF           #IMPLIED>
```

```
<!ELEMENT     Arc                EMPTY                    >
    <!ATTLIST Arc        NœudPrec      IDREF     # REQUIRED
                         NœudSuiv      IDREF     # REQUIRED
                         type          NUMBER    #REQUIRED
                         poids         CDATA               '0'>

<!ELEMENT     Relation   (Liaison+)>
    <!ATTLIST Relation   type          CDATA     #REQUIRED>

<!ELEMENT     Liaison    (Dep & Arr)>
    <!ATTLIST Liaison    type          CDATA     #REQUIRED>

<!ELEMENT     Dep        (Lien+)                          >
<!ELEMENT     Arr        (Lien+)                          >

<!ELEMENT     Lien       EMPTY                                  >
    <!ATTLIST Lien       Ref           IDREF     #REQUIRED>
```

Remarques :

- Les "( )$^{+}$" indiquent que l'élément entre parenthèses doit apparaître au moins une fois, et peut être répété autant de fois que nécessaire.
- Les "( )* indiquent que le nombre d'occurrences de l'élément est 0 ou plus.
- Chaque élément est décrit dans sa structure d'abord ("<!ELEMENT...), puis la liste de ses attributs est donnée (<!ATTLIST...).
- Le type (et le poids) des arcs de structure est conservé dans les nœuds de telle façon à lever l'ambiguïté de construction d'un treillis qui existe si l'on se contente de ne donner que les ordres partiels. Ces types ont pour effet de fixer le "chemin" sur lequel se trouve le nœud.

3.1.2 Arbres SGML et treillis de TELA

On notera qu'il n'y a qu'un lien partiel entre les arbres construits par un analyseur SGML lors du parcours de la description d'une structure TELA sous forme d'un fichier XML, et les treillis de TELA. En effet, lors de l'analyse du document XML, l'analyseur SGML associe en fait un arbre, à chaque treillis de TELA. La structure de treillis est en fait codée dans le document XML par la combinaison de la description structurelle et des références qui décrivent la succession des nœuds (l'ordre partiel).

3.1.3 DTD de la structure complète : un segment XML et TELA

La structure de représentation interne comporte deux parties : la forme primaire, codée par un segment XML, et la forme analytique (TELA). Comme nous venons de donner une description de la forme analytique sous le même formalisme XML, l'ensemble de la structure est maintenant manipulable sous XML. Cela nous permet de concevoir le document XML comme auto explicatif de sa structure analytique.

3.2 Une spécification objet de TELA

3.2.1 Le modèle objet

Les objets du modèle objet classique [Masini et al. 1991] pris en considération ici sont chacun les instances d'une classe. Une classe est composée de champs, et de méthodes qui s'appliquent sur ces attributs.

CLASSE = CHAMPS + METHODES

3.2.2 Les classes de TELA : aspect structurel

Une spécification des différentes classes abstraites de TELA est maintenant proposée, selon le modèle ci-dessus. La spécification suivante suit l'implémentation informatique abstraite donnée au paragraphe II. Une classe est ici définie par ses champs et ses méthodes. Les méthodes reprennent les opérations de base du paragraphe II, plus un certain nombre d'autres méthodes nécessaires à l'implémentation objet de TELA.

classe TELA	intitulé	spécification	type de donnée
Champs	identificateur	identifie l'instance	chaînes de caractères
	listeDeTreillis	liste des treillis de TELA	liste d'instances de la classe TREILLIS
	listeDeLiaisons	liste de liaisons de TELA spécifiant la relation de correspondance sur TELA	liste d'instances de la classe LIAISON

classe TELA	intitulé	spécification
Méthodes	nouvelleTELA	crée une nouvelle instance de la classe TELA
	suppressionTELA	supprime l'instance de TELA
	affecterIdentificateur	permet d'affecter l'identificateur
	obtenirIdentificateur	renvoie l' identificateur de l'instance de TELA
	ajouterTreillis(i, T)	crée un nouveau treillis pour la structure à l'étage "i"
	obtenirTreillis(i)	donne le i-ème treillis de la liste des treillis de TELA.
	SupprimerTreillis(i) :	supprime le treillis de l'étage i, et toutes les relations afférentes
	ajouterLiaison (type, liste ordonnée de nœuds)	crée une nouvelle relation entre treillis de TELA, du type donné, et pour les nœudspécifiés (appartenant éventuellement à plusieurs treillis).
	obtenirLiaison(id)	renvoie la relation d'identificateur "id" de TELA.
	supprimerLiaison	supprime la relation et tous les liens afférents

classe TREILLIS	intitulé	spécifications	représentation

Champs	identificateur	identifie de façon unique le treillis	chaîne de caractères
	inf	nœud borne inférieure du treillis	instance de la classe NŒUD
	sup	nœud borne supérieure du treillis	instance de la classe NŒUD
	listeNœuds	ensemble des nœuds du TREILLIS	liste d'instances de la classe NŒUD
	listeArcs	ensemble des arcs du treillis	liste d'instances de la classe ARC

classe TREILLIS	intitulé	spécifications
Méthodes	nouveauTreillis	crée une nouvelle instance d'un treillis
	supprimerTreillis	supprime le treillis
	affecterIdentificateur(Id)	affecte l'identificateur Id
	obtenirIdentificateurs	permet de fixer les identificateurs
	ajouterNœud (liste de triplets (nœud précédent, type de l'arc, poids de l'arc), liste de triplets (nœud suivant, type de l'arc, poids de l'arc)) :	crée un nouveau nœud et autant d'arcs entre les nœuds spécifiés, de type et de poids spécifiés.
	obtenirNœud (Id)	permet de récupérer un nœud par son identificateur Id
	supprimerNœud (Id)	supprime le nœud, les arcs afférents, et les liens de relations connectés à ce nœud . Si une des relations dont on vient de supprimer un lien n'a plus de raison d'exister, elle est aussi supprimée.

classe NŒUD	intitulé	spécifications	type de donnée
Champs	identificateur	identificateur unique du nœud	liste de chaînes de caractères
	typeNœud	type du nœud	entier
	coordonnées	ensemble des coordonnées du nœud suivant le référentiel	liste de couples d'entiers
	données	cette structure comporte les données que le nœud porte.	liste de listes d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'instances de la classe ARC
	arcsSuivants	liste des arcs qui partent du nœud	liste d'instances de la classe ARC
	listeLiaisonsE	liste des liaisons	liste d'identificateurs de

	listeLiaisonsS	entrantes liste des sortantes	liaisons	liaisons liste d'identificateurs de liaisons

classe NŒUD	spécifications	représentation
Méthodes	nouveauNœud	crée un nouveau nœud vide
	supprimerNœud	supprime le nœud, tous les arcs de structures relatifs, tous les liens relatifs, et éventuellement les liaisons qui deviennent caduques
	obtenirIdentificateur	renvoie l'identificateur
	affecterIdentificateur	permet d'affecter l'identificateur
	obtenirType	renvoie le type
	affecterType(t)	affecte le type t au nœud
	obtenirCoordonnées(i)	renvoie la i-ième coordonnée
	affecterCoordonnées(i, (a,b))	affecte le couple (a, b) à la i-ième coordonnée du nœud
	obtenirDonnées(i, j)	renvoie le j-ième élément de la i_ième liste des données
	affecterDonnées(i, j, d)	permet de mettre à jour le j-ième élément de la i_ième liste des données que porte le nœud
	ajouterArc(sens, arc)	ajoute un arc vers le nœud courant est le départ si le sens est "précédent", l'arrivée si le sens est "suivant".
	supprimerArc(sens, arc)	supprime le nœud de la liste des nœuds précédents si le sens est "précédent", ou de la liste des nœudsuivants si sens est "suivant"
	obtenirListeArcs(sens)	rend la liste des nœuds voisins suivants ou précédents selon "sens".
	obtenirListeLiaisons(sens)	renvoie la liste des liaisons auxquelles participe le nœud selon le sens
	affecterListeLiaison(sens, L)	affecte la liste L de liaisons au nœud

classe ARC	identificateur	spécification	type de donnée
Champs	identificateur	identificateur unique	chaîne de caractères
	type	type de l'arc	entier
	poids	poids de l'arc	réel
	IdNœudDépart	nœud de départ de l'arc	chaîne de caractères
	IdNœudArrivée	nœud d'arrivée de l'arc	chaîne de caractères

classe ARC	identificateur	spécification
Méthodes	nouvelArc	crée un nouvel arc
	obtenirIdentificateur	renvoie l'identificateur du nœud

	affecterIndentificateur(Id)	affecte l'identificateur Id à l'arc
	obtenirType	récupère le type de l'arc
	affecterType(t)	affecte le type t à l'arc
	obtenirPoids	récupère le type de l'arc
	affecterPoids(poids)	permet de régler le poids de l'arc
	obtenirNœudDépart	renvoie le nœud d'arrivée
	obtenirNœudArrivée	renvoie le nœud d'arrivée
	affecterNœudDépart(Id)	affecte le nœud d'identificateur Id comme nœud de départ
	affecterNœudArrivée(Id)	affecte le nœud d'identificateur Id comme nœud d'arrivée
	supprimerArc	supprime l'arc

classe LIAISON	identificateur	spécification	type de données
Champs	identificateur	identificateur du champ	chaîne de caractères
	typeLiaison	référence à un schéma de liaison qui guide la construction de cette relation.	entier
	nœudDepart	c'est le nœud de liaison, il est du type indiqué.	chaîne de caractères
	nœudArrivée	ce sont l'ensemble des nœuds des treillis qui sont liés par cette relation. Ils sont donnés dans un ordre précis.	chaîne de caractères

classe LIAISON	spécifications	représentation
Méthodes	nouvelleLiaison	crée une liaison
	supprimerLiaison	supprime la relation et l'ensemble des liens aux nœuds de structure impliquant cette relation.
	obtenirIdentificateur	renvoie l'identificateur
	affecterIdentificateur	permet de fixer l'identificateur
	obtenirType	renvoie le type de la liaison
	affecterType	modifie le type de la relation
	obtenirListeNœuds(sens)	renvoie la liste des nœuds selon le sens
	affecterListeNœuds(sens, L)	affecte la liste de nœuds de départ si sens est "départ", d'arrivée si sens est "arrivée"

classe SCHEMALIAISON	identificateur	spécification	type de données
Champs	identificateur	identificateur unique	chaîne de caractères
	typeLiaison	type de la liaison à créer	chaîne de caractères
	listeTypeDépart	liste des types des nœuds de départ	liste d'entiers
	listeTypeArrivée	liste des types des	liste d'entiers

		nœuds de d'arrivée	
	listeDimDonnées Arrivée	dimensions des tableaux de données	liste des couples d'entiers
	conditionsDépart	conditions sur les nœuds de départ	liste de couples (élément, valeur)
	conditionsArrivée	conditions sur les nœuds d'arrivée	liste de couples (élément, valeur)

classe SCHEMALIAISON	spécifications	représentation
Méthodes	nouveauSchéma	crée un nouveau schéma
	obtenirIdentificateur	renvoie l'identificateur
	affecterIdentificateur(Id)	permet d'affecter l'identificateur
	obtenirTypeLiaison	renvoie le type de la relation
	affecterTypeLiaison(t)	affecte t au type de la liaison que crée le schéma lorsqu'appliqué
	obtenirListeTypesNœud(sens)	renvoie la liste des types des nœuds d'entrée ou de sortie selon le sens
	affecterListeTypesNœud(sens, listeTypes)	affecte la liste des types des nœuds d'entrée ou de sortie selon le sens
	obtenirListeDimDonnées	renvoie la liste des couples de dimensions des données des nœuds à créer
	obtenirListeConditions(sens)	renvoie la liste des conditions d'entrée ou de sortie selon le sens
	affecterConditions sens, liste (éléments, conditions)	affecte la liste de conditions

3.2.3 Hierarchie "has-a" entre classes décrites

L'ensemble des classes décrites sont liées par des relations du type "est partie de". Ceci peut être représenté par un schéma du type suivant.

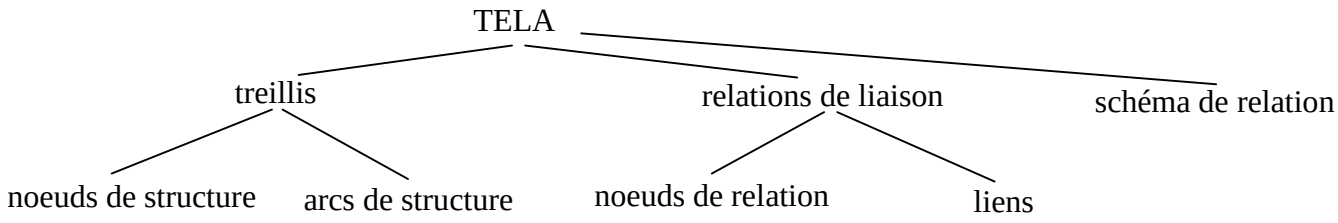


Figure 7 : Relation "Has-a" entre classes de TELA

Conclusion

Nous venons de donner une formalisation des structures TELA. Une description externe fondée sur les objets, et une définition du type de document XML a été donnée pour la manipulation et l'échange aisés, en particulier via Internet.

Chapitre 7

TELAM :
Spécialisation de TELA pour la Traduction Fondée sur la Mémoire

Contenu du chapitre

Introduction

Le chapitre 6 a présenté les structures TELA et TELAM de façon abstraite. Ce chapitre 7 propose une première implémentation informatique de TELAM. Le but du chapitre est de démontrer la faisabilité du modèle en l'implémentant algorithmiquement. L'analyse d'un segment XML en une structure TELAM est menée en prenant comme référence la présentation faite aux chapitres précédents. Pour atteindre ces objectifs, le paragraphe II pose quelques conventions et précise quelques outils de représentation et de manipulation des données. Le paragraphe final propose une évaluation de la complexité de la structure et des algorithmes mis en jeu. Les algorithmes qui suivent se basent sur la description des champs et méthodes d'objets présentés au Chapitre 6.

1. TELAM : application de TELA à la traduction fondée sur la mémoire

Introduction

Après une spécification formelle de la structure TELA dont, on l'a vu, les domaines d'applications sont divers, ce paragraphe propose un retour vers le problème de la représentation de l'information dans les systèmes de traduction fondés sur la mémoire. Il est montré ici comment une instance de la structure TELA convient au modèle nécessaire à la TFM, décrit au chapitre 5. Cette structure spécialisée sera dénommée TELAM : Treillis Etagés Liés pour les traitements Linguistiques Automatisés appliqués aux Mémoires de traduction.

1.1 Structures de données

1.1.1 Les nœuds de structure

Les tableaux suivants donnent la correspondance entre la description objet de la structure telle faite au chapitre 6, et celle découlant de l'approche empirique du chapitre 5. Cela donne les bases nécessaires à l'expression des algorithmes qui suivent. Nous reprenons chaque

tableau de champs de la partie III.B du chapitre 6, et montrons la correspondance avec les types de nœuds pour chaque étage du chapitre 5.

type t1	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t1	entier
	coordonnées	rangT, rang00, rang01	liste de couples d'entiers
	données	caractère représenté	liste d'une liste d'un élément
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

type t2	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t2	entier
	coordonnées	rangT, rang00, rang01, rang10	liste de couples d'entiers
	données	chaîne de caractères du mot	liste d'une liste d'un élément
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

type t3	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t3	entier
	coordonnées	rangT, rang00, rang01	liste de couples d'entiers
	données	[(nombre n de balises doubles représentées)[1] (code balise 1, rang00, rang01) .... (code balise 1, rang00, rang01]	liste de listes d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

[1] Le rang00 de début représente le rang00 du premier caractère de la balise de début, et le rang01 de début le numéro d'occurrence de la balise de début. De même pour le rang00 et rang01 de fin.

type t4	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t4	entier
	coordonnées	rangT, rang00, rang01	liste de couples d'entiers
	données	[(nombre n de monobalises représentées) (code balise 1, rang00, rang01) .... (code balise n, rang00, rang01)]	liste de listes d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

type t5	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t5	entier
	coordonnées	rangT	liste de couples d'entiers
	données	(lemme attributs lexicaux, attributs sémantiques)	liste d'une liste d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

type t6	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t6	entier
	coordonnées	rangT	liste de couples d'entiers
	données	[(partie de pivot) (conditions sur la partie)]	liste de deux listes d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

type t7	intitulé	donnée concrète	type de donnée
Champs	identificateur	estampille	liste de chaînes de caractères
	typeNœud	t7	entier
	coordonnées	rangT	liste de couples d'entiers

	données	(terme ∪ liste des attributs terminologiques)	liste de listes d'éléments
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons

1.1.2 Les arcs de structure

Les arcs de structures suivent exactement les spécifications générales. Dans un premier temps, bien que la nécessité des poids sur les opérations décrites au chapitre 5 ne soit pas immédiate, les poids seront conservés pour plus tard. On notera leur type ai, où i est numéro de type. Le type par défaut est "1".

1.1.3 Les schémas de liaison

Les schémas de liaisons sont du type général décrit au chapitre 6.

1.1.4 Les relations de liaison

Les liaisons suivent les spécifications du chapitre 6. Les liens entre le nœud de relation de liaison et les nœuds ainsi que leur ordre sont implicitement donnés par leur ordre dans la liste des nœuds liés. Les relations de liaisons sont instanciées à l'aide de *schémas de liaisons* vus ci-dessus par "*application d'un schéma de liaison*".

1.2 Opérations

1.2.1 Schéma linguistiques, schémas de liaison, et relations de liaison

Les schémas linguistiques tels qu'ils ont été introduits peuvent être vus sous le formalisme des schémas de liaison. Un ensemble de nœuds d'expression pivot est alors instancié par une relation de liaison. Le schéma linguistique de la page 58 du Chapitre 5, peut par exemple être vu comme l'application de ce schéma de traduction sur les nœuds du mots du deuxième étage. Ce schéma peut s'écrire comme suit :

S = [R26 //type de liaison
[(N1,t2)] //liste types entrée
[(N2, t6), (N3, t6), (N4, t6), (N5, t6), (N7, t6), (N8, t7)] //liste types sortie
[CondDépart : //conditions de départ
Nœuds : (N1.obtenirDonnées(1,1) = "et")]
[CondSortie : //condition de sortie
{Nœuds : (N2.Données(1,1) = "A")
(N3.Données(1,1) = "et")
(N4.Données(1,1) = "B")
(N5.Données(1,1) = "A'")
(N6.Données(1,1) = "and")
(N7.Données(1,1) = "B'") }
Arcs : {(T6.inf, N2, t_{a2}), (N2, N3, t_{a2}), (N3, N4, t_{a2}), (N4, T6.sup, t_{a2}),
(T6.inf, N5, t_{a2}), (N5, N6, t_{a2}), (N6, N7, t_{a2}), (N7, T6.sup, t_{a2})}
Liens : {(N1.listeNœudsPréc,N2), (N1,N3), (N1.listeNœudsSuiv,N2),
(N2,N5), (N1,N6), (N4,N7)}]
]

1.3 Outils de manipulation et de stockage

1.3.1 Identification des nœuds par des estampilles

triplets d'estampille

La notion d'estampille est proposée ci-dessous pour permettre d'étiqueter l'ensemble des nœuds d'une structure TELAM. L'estampille joue donc le rôle de l'identificateur dont le référentiel est la structure TELAM entière (voir Chapitre 6 pour la définition d'un référentiel d'identificateur). Ces estampilles doivent à la fois refléter la notion d'étage mais aussi, idéalement, la notion d'ordre partiel qui est inhérente à la structure des treillis. Pour atteindre cet objectif, un nœud est vu comme dépendant de trois paramètres :

- le treillis dont il fait partie (l'étage)
- le chemin du treillis auquel il appartient (le sous-étage)
- sa position relative au segment XML de départ.

La position relative d'un élément par rapport au segment de départ est :

- soit le couple des positions extrêmes de l'élément, s'il y a correspondance directe du nœud avec un tel élément : c'est le cas pour les nœuds de caractères du premier chemin du treillis des caractères, des nœuds de balises du premier chemin des treillis de balises, etc.
- soit le couple des positions extrêmes héritées par construction. Dans le cas suivant du nœud de "é", son rang00 est (3, 10) car il est construit avec l'ensemble des nœuds compris entre les nœuds de caractères "&" et ";", de rang00 respectifs (3, 3) et (10, 10)[2].

Chaque nœud est donc représenté par un triplet (étage, sous-étage, rang00). Voici la représentation par position de rang00 des nœuds d'un extrait de structure TELA.

ét	sé	nœud																									
0	0	0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	..
												0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	
1	1	inf1	L	'	&	e	a	c	u	t	e	;	p	i	n	e	t	t	e	_	<	h	i	1	>	e	..
	2'											é															..
2	1	inf2	L	'	épinette																					est	
3	1	inf3																			hi1						

Table 1 : Position de rang00 des différents nœuds.

La première colonne représente l'étage, la seconde le sous-étage. On notera que l'on réserve toujours une ordonnée ("(0, 0)" sur l'axe des abscisses) pour la borne inférieure des treillis de chaque étage. De même la dernière position est réservée à la borne supérieure. Dans l'exemple ci-dessus, voici les triplets d'estampilles de quelques nœuds :

nœud "L" : (1, 1, (1, 1))
nœud "u" : (1, 1, (7, 7))
nœud "é" : (1, 2, (3, 10))
nœud "épinette" : (2, 1, (3, 17))

[2] Un caractère est considéré comme ponctuel par rapport au référentiel du treillis, et n'a donc pas de "durée", ce qui exprimé par un instant ou la coordonnée de départ et celle d'arrivée sont les mêmes.

nœud "hi1" : (3, 1, (19, 23))

Ces estampilles nous permettent de définir un ordre partiel sur les triplets comme suit :

Définition 1 : ordre partiel sur les triplets d'estampilles

(a1, b1, (c1, d1)) < (a2, b2, (c2, d2)) ⇔	soit a1 < a2 soit a1 = a2 et b1 < b2 soit a1 = a2 et b1 = b2 et c1 < c2 soit a1 = a2 et b1 = b2 et c1 = c2 et d1 < d2

Il existe un morphisme d'ordre partiel entre l'ensemble des triplets munis de l'ordre défini ci-dessus, et l'ensemble des nœuds du treillis muni de l'ordre partiel correspondant. On vérifie bien par exemple :

"L"(1, 1, (1, 1)) < "u"(1, 1, (7, 7)) < "é"(1, 2, (3, 10)) < "épinette"(2, 1, (3, 17)) < "hi1"(3, 1, (19, 23))

estampille numérique

Pour ne pas avoir à manipuler des triplets, moins pratiques que des valeurs numériques, il est possible de ramener ces triplets à des entiers par le processus suivant. On choisit un ordre de grandeur k. Alors l'estampille numériquecorrespondant au triplet t = (a1, b1, c1) vérifie :

$$s = k^2 \times a1 + k \times b1 + d$$

Il suffit alors de choisir l'ordre de grandeur k assez important pour que deux estampilles s'appliquant à deux nœuds différents ne soient pas égales. Cela dépend du nombre de caractères que peut comporter le segment XML. Par exemple, si ce nombre ne dépasse pas 1000, on peut prendre $k = 1000 = 10^3$, et l'estampille numérique vérifie alors :

$$s = 10^6 \times a1 + 10^3 \times b1 + d$$

Les triplets d'estampilles de l'exemple précédent sont alors transformés en :

nœud "L" : 1001001
nœud "u" : 1001007
nœud "é" : 1002010
nœud "épinette" : 2001017
nœud "hi1" : 3001023

On peut vérifier que les tris précédents s'appliquent encore avec ces estampilles numériques. Le choix d'estampilles numériques est donc restrictif car si le nombre de caractères d'un des segments traités dépasse k, la bijection entre les nœuds et les estampilles peut ne plus être respectée. Ce choix peut cependant s'avérer pratique dans l'implémentation du modèle.

1.3.2 Identificateurs d'arcs de structure

Soit un arc AB entre les nœuds A et B, d'estampilles respectives s_A et s_B, calculées à l'aide de la base k. Alors l'identificateur s_{AB} de l'arc AB se base sur le fait qu'il ne peut y avoir qu'un seul arc entre deux nœuds, et que deux nœuds différents ne peuvent avoir la même estampille qui se calcule comme suit :

$$s_{AB} = k2\ s_A + s_B$$

1.3.3 Structures de données des nœuds (décorations)

Les structures de données requises dans la spécification des types de nœuds t1 à t7 donnés au Chapitre 6 vont du caractère à des listes de listes d'attributs. L'ensemble de ces données peut être supporté uniformément par plusieurs structures comme des listes de listes, ou des tableaux à deux dimensions, d'ordres variables.

Il est fait ici le choix (arbitraire) pour la suite de ce chapitre de tableaux à deux dimensions. Ainsi un caractère occupera une cellule d'un tableau à une dimension d'ordre (1 x 1), alors que l'enregistrement d'un lemme occupera plutôt un tableau à 3 lignes d'ordre (3 x n), une ligne étant affectée respectivement au lemme, aux attributs lexicaux, aux attributs sémantiques.

L'implémentation qui suit se basera donc sur un tableau T à deux dimensions d'ordre variable. Son ordre sera par convention supérieur à (1 x 2) pour gérer les données du nœud : les cellules T(0,0) et T(0,1) donnant les ordres des première et seconde dimensions. Ainsi si l'on a, par exemple, besoin d'un tableau à 3 lignes et 5 colonnes de données effectives, on aura : T(0,0) = 3+1 = 4, T(0,1) = 5+1. La ligne supplémentaire sert à gérer T(0,0) et T(0,1).

Cette technique a le désavantage d'utiliser des tableaux éventuellement creux, mais a l'avantage de posséder toute l'information dans le tableau même, qui est alors le seul objet à manipuler, et qui peut être défini dynamiquement.

1.3.4 Gestion du nombre d'objets de chaque éléments

Pour avoir une idée de l'ordre de grandeur de la structure TELAM, chaque treillis ainsi que la structure TELAM globale comportera un champ supplémentaire (dans le modèle objet) indiquant le nombre de nœud de structures, de nœuds de relations internes pour les treillis, externes pour TELAM, d'arcs de structures, et de liens. Voici une liste de ces champs et des méthodes associées par classe :

classe TELA

Champs

NbNœuds
NbArcs
NbRelationsExternes
NbLiens

Méthodes

obtenirNbNœuds
obtenirNbArcs
obtenirNbRelationsExternes
obtenirNbNLiens
affecterNbNœuds
affecterNbArcs
affecterNbRelationsExternes
affecterNbNLiens

classe Treillis

Champs

nbNeuds
nbArcs
nbRelationsInternes
nbLiens

Méthodes

obtenirNbNœuds
obtenirNbArcs

obtenirNbRelationsExternes
obtenirNbNLiens
affecterNbNœuds
affecterNbArcs
affecterNbRelationsExternes
affecterNbNLiens

2. Algorithmes

2.1 Algorithmes de base

2.1.1 Retour sur quelques méthodes des classes de TELA

Nous proposons d'abord une série de petites méthodes et procédures utiles pour les algorithmes qui suivront.

Methode nouvelleEstampille de la classe Nœud

Chaque nœud possède un identificateur unique, " l'estampille ", qui le caractérise par rapport à l'ensemble de la structure TELA, et qui a été présentée plus haut. Nous adopterons la version numérique de l'estampille, en prenant soin de laisser définissable la constante de base k. L'estampille demande donc un petit calcul qui dépendra de l'étage, du sous-étage, et du rang00 (rang00 = (c, d)) tel que vu plus haut. La méthode nouvelleEstampille est une méthode de la classe générale "nœud", dont héritent les sous-classes de nœuds. Nous considérerons que $k = 10^b$.

```
Méthode nouvelleEstampille(étage, sous-étage, (c, d))
Début
    s ← 10^2b x étage + 10^b x sous-étage + d
    rendre s
Fin
```

Méthode 1 : Calcul de l'estampille d'un nœud

Méthode rangT (rang de treillis) de la classe treillis

Le rangT d'un nœud est l'échelle dont le référentiel est le treillis, qui permet de savoir quel est le numéro d'occurrence du nœud dans le treillis. Celui-ci est calculé grâce à une méthode qui incrémente le champ nbNœuds du treillis. Cette échelle ne renseigne donc pas sur la position du nœud dans le treillis, contrairement à l'estampille.

```
Méthode nouveauRangT
    Début
        nbNœuds ← nbNœuds + 1
        rendre nbNœuds
    Fin
```

Méthode 2 : Calcul du nouveau rang d'un nœud dans un treillis (classe treillis)

Création d'un nouveau nœud pour la classe treillis

La création d'un nouveau nœud de structure pour TELAM, est assurée par la méthode "nouveauNœud" de la classe de nœud. Une implémentation de cette méthode est maintenant proposée. Cela permet de n'avoir qu'une seule méthode de la classe générale "nœud", dont héritent les sous classes t1 à t7.

La méthode nouveauNœud prendra, parmi ses arguments, le nombre de lignes m et de colonnes n du tableau des données du nœud. Le type t sera aussi pris en argument pour permettre de traiter de façon homogène la création de tout nouveau nœud (ceci est possible grâce, entre autres, à la flexibilité de la structure de données telle qu'elle a été décrite plus haut).

```
méthode nouveauNœud(t, m, n)
    Debut
        identificateur ← nil
        typeNœud ← t
        coordonnées ← nil
        T ← nouveauTableau (m, n)
        arcsPrécédents ← nil
        arcsSuivants ← nil
        relationsDeLiaison ← nil
    Fin
```

Méthode 3 : Création d'un nouveau nœud (classe treillis)

Gestion des données de la classe Nœud

Il a été proposé de stocker les données du nœud de façon homogène dans un tableau T à deux dimensions. La méthode "affecterDonnées" permet d'affecter les données d'un nœud de la façon suivante :

```
méthode affecterDonnées( T[ ] )
    Début
        Pour i ← 1 à T(0,0)
            Pour i ← 1 à T(i,0)
                T(i, j) ← dij
            Fin Pour
        Fin Pour
    Fin
```

Méthode 4 : Affectation des données d'un nœud (classe Nœud)

Une méthode de lecture des données permet de récupérer ces données :

```
méthode obtenirDonnées(i, j)
    Début
        Renvoyer T(i, j)
    Fin
```

Méthode 5 : Lecture des données d'un nœud (classe Nœud)

Gestion des arcs de structure

L'ajout d'un arc de structure au niveau d'un nœud est géré par la méthode "ajouterArc", et fait appel à trois procédures :

- la création d'un nouvel arc de structure (méthode nouvelArc de la classe Arc)
- l'ajout de cet arc au nœud de départ (méthode ajouterArcVoisin de la classe Nœud)
- l'ajout de cet arc au nœud d'arrivée (méthode ajouterArcVoisin de la classe Nœud)

Voici les algorithmes liés à ces méthodes :

```
méthode ajouterArc(type, poids, nœudDépart, nœudArrivée)
    Début
```

```
        A ← nouvelArc(type, poids)
        A.affecterType(type)
        A.affecterPoids(poids)
        A.affecterIdentificateur(k2 x nœudDépart.ObtenirIdentificateur
                                    + nœudArrivée.obtenirIdentificateur )
        A.affecterNœudDépart(nœudDépart)
        A.affecterNœudArrivée(nœudArrivée)
        nœudDépart.ajouterArc ("suivant", nœudArrivée)
        nœudArrivée.ajouterArc ("précédent", nœudDépart)
    Fin
```

Méthode 6 : ajout d'un arc de structure (classe TELAM)

Extraction des nœuds d'un chemin d'un des treillis d'une structure TELAM

Un treillis TTT étant donné, voici un algorithme générique d'extraction de la liste des nœuds de deux des chemins les plus usités d'un des treillis de TTT : le "premier chemin" et le "dernier" chemin créés. Dans l'algorithme suivant, "l'étage" est un entier représentant l'étage du treillis de TTT duquel le chemin sera extrait. Le "chemin", s'il est égal à "premier" représente le premier chemin créé dont les nœuds sont liés à l'aide d'arcs de premier type 1, qui correspondent à l'analyse de plus bas niveau, et au premier chemin créé. S'il est égal à "dernier", cela correspond au chemin qui code l'analyse de plus haut niveau. Ce chemin passe alors par les arcs de type t_{ak} le plus haut à chaque choix de parcours.

```
    Méthode extraireListeNœuds(étage, chemin) : liste de nœuds
    Début
        LL ← nil
        TT ← obtenirTreillis(étage)
        N ← TT.inf
        LL ← LL ∪ N
        Tant Que (N ≠ TT.sup) Faire
            L ← N.obtenirListeArcs("suivants")
            Si (chemin == "premier") Alors
                M ← Premier(L).obtenirNœudArrivée
            Sinon
                Si (chemin == "dernier") Alors
                    M ← Premier(L).obtenirNœudArrivée
                Fin Si
            Fin Si
        LL ← LL ∪ M
        N ← M
        Fin Tant Que
    Fin
```

Méthode 7 : extraction d'une liste de nœuds d'un treillis de TELAM (classe TELAM)

2.1.2 Traitements du segment XML

Introduction

La structure TELAM est construite à partir d'un segment XML. La première des procédures suivantes extrait les éléments de ce segment et les classe dans un tableau. La deuxième procédure permet de distinguer quel est le type d'un élément en examinant sa syntaxe.

Extraction et groupement des informations du segment dans un tableau (procédure segmentVersTableau)

SS regroupe les informations du segment XML. Cette procédure rend un tableau SST(i, j) à deux dimensions dont une ligne représente un élément du segment XML, sauf pour la première ligne qui comporte des informations générales. Les cellules de premier indice i = 0 sont réservées pour les informations sur les dimensions du tableau :

- SST(0, 1) ← nombre de caractères du segment
- SST(0, 2) ← nombre d'éléments du tableau
- SST(0, 3) ← nombre de balises (doubles ou monobalises) du segment

Puis pour i > 0, un i est affecté par élément du segment. Chaque ligne du tableau contient :

- SST(i, 1) : première coordonnée du rang00 de début du premier caractère de l'élément
- SST(i, 2) : deuxième coordonnée du rang00 de début du premier caractère de l'élément
- SST(i, 3) : première coordonnée du rang01 d'occurrence de l'élément dans le segment
- SST(i, 4) : deuxième coordonnée du rang01 d'occurrence de l'élément dans le segment
- SST(i, 5) : les code de l'élément, comme il est trouvé dans le segment XML.

Recherche du type d'un élément du segment XML

Quatre types d'éléments sont possibles dans un segment XML :

- 1 : un texte ou une référence d'entité (qui seront représentés à l'étage 2)
- 2 : la balise de début d'un couple de balises doubles
- 3 : la balise de fin d'un couple de balises doubles
- 4 : une monobalise.

La fonction tagType renvoie un de ces entiers pour signifier le type de l'élement.

```
Fonction tagType(entity)
// Donne le type de l'élément selon notre classification (cf. III.A.3), les types sont :
//     - texte et références : 1
//     - balise double de début: 2
//     - balise double de fin: 3
//     - monobalise : 4
Début
    c1 ← firstCharacter(entity)[3]
    c2 ← secondCharacter(entity)
    c3 ← penultimateCharacter(entity)
    c4 ← lastCharacter(entity)
```

[3] Nous supposons disposer d'une fonction "firstCharacter" qui rend le premier caractère d'une chaîne. Cette fonction est disponible dans tous les langages de programmation

```
        Si ((c1 == "<") et (c4 == ">"))
            Si ((c2 == "?") ou (c2 == "!"))
                tagType ← 4
            Sinon
                Si (c3 == "/")
                    tagType ← 4
                Sinon
                    Si (c2 == "/")
                        tagType ← 3
                    Sinon
                        tagType ← 2
                    Fin Si
                Fin Si
            Fin Si
        Fin Si
Fin
```

Algorithme 1 : Recherche du type d'une élément

2.1.3 Ajout d'un nœud dans un treillis

Introduction

La construction d'une structure TELAM consiste principalement à créer des nœuds et leurs arcs relatifs. La structure des nœuds étant identique à quelques détails près. Les différences entre les sept types de nœuds d'une structure TELAM portent essentiellement sur le type de données que porte chaque nœud et sur le nombre des étiquettes qui le qualifient, selon leur référentiel. Ayant introduit une procédure particulière pour généraliser la création des différents types de nœuds, il est possible de spécifier une procédure générale pour l'ajout ou l'insertion d'un nœud.

Ajout d'un nœud : méthode "ajouterNœud" de la classe Treillis

Il est supposé que les informations suivantes sont connues :

- le type de nœud : t_n
- le type d'arcs liant ce nœud à ses voisins : t_a
- rang00 et rang01 de l'entité à ajouter (pour construire l'estampille)
- nombre de lignes m et de colonnes n nécessaires pour stocker les données
- les données et le nombre de parties de l'entité sous forme d'un tableau T[ ][4]
- le nœud précédent N_{i-1}
- le nœudsuivant N_{i+1}

```
ajouterNœud(N, (listeNœudsPrécédents, listeTypeArcsPre), (listeNœudsSuivants,
                                    listeTypeArcsSuiv), rang00, rang01, T[ ])
Début
    N.affecterIdentificateur(nouvelleEstampille(1, 1, rang00))
    N.affecterCoordonées(1, rangT)
    N.affecterDonnées(T[ ])
    Pour (chaque Ni de listeNœudsPrécédents et tai de listeTypeArcsPre)
        ajouterArc(tai, Ni, N)
```

[4] T0, 0) donne le nombre de lignes, T[i, 0] donne le nombre d'éléments par ligne

```
        Fin Pour
        Pour (chaque Ni de listeNœudsSuivants et tai de listeTypeArcsSuiv)
            ajouterArc(tai, N, Ni)
        Fin Pour
    Fin
```

Méthode 8 : Ajout d'un nœud au treillis (classe Treillis)

Remarque :

- On notera que le nœud courant N, peut avoir plusieurs précédents et plusieurs successeurs. La liste des voisins précédents, et la liste des voisins suivants est donnée en paramètres, suivie de la liste des types des arcs qui relient ces nœuds à N.
- La méthode "ajouterNœud" ajoute un nœud B accompagné des arcs le reliant à ses voisins A et C. Si l'on veut construire un chemin de nœuds, il est toutefois nécessaire de supprimer l'arc préalablement créé entre A et C, comme le schéma suivant le montre :

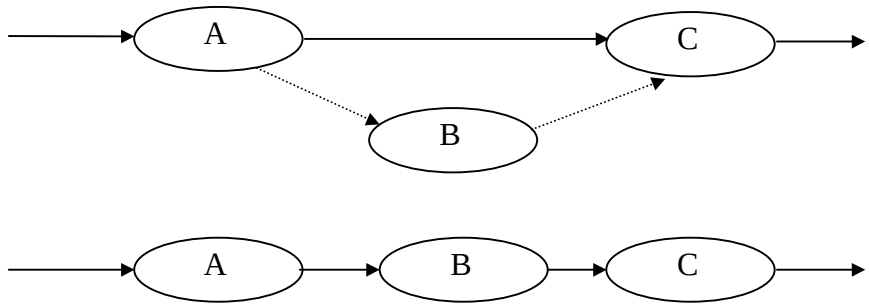


Figure 1 : Insertion d'un nœud : suppression de l'arc AC

Cela donne l'algorithme suivant résultant d'une légère modification de l'algorithme "ajouterNœud" :

```
insérerNœud(N, (listeNœudsPrécédents, listeTypeArcsPre), (listeNœudsSuivants,
                                        listeTypeArcsSuiv), rang00, rang01, (i, j, T(i, j)))
Début
    Premier(Premier(liste.NœudsPrécédents).obtenirListeArcs).supprimerArc
    ajouterNœud(N (listeNœudsPrécédents, listeTypeArcsPre), (listeNœudsSuivants,
                                        listeTypeArcsSuiv), rang00, rang01, (i, j, T(i, j)))
Fin
```

Méthode 9 : Insertion d'un nœud au treillis (classe treillis)

```
méthode supprimerArc(A)
Début
    N1 ← A.obtenirNœudDépart
    N1.supprimerArc(suivant, A)
    N2 ← A.obtenirNœudArrivée
    N2.supprimerArc(départ, A)
    A.supprimerArc
Fin
```

Méthode 10 : Suppression d'un arc

2.1.4 Ajout d'une relation de liaison

L'ajout d'une liaison est une opération qui a lieu soit au niveau du treillis, c'est alors une relation de liaison "interne" ne faisant intervenir que des nœuds du treillis ; soit au niveau de la structure TELA(M) pour les liaisons externes concernant des nœuds provenant de treillis distincts. Nous considérerons ici le cas général de la méthode "*ajouterLiaison*" appartenant à la classe TELA et englobant les deux possibilités précédentes.

```
méthode ajouterLiaison(type, listeNœudsDépart, listeNœudsArrivée)
    Début
                                                        // côté liaison
        R ← nouvelleLiaison
        R.affecter(nouvelIdentificateurLiaison)
        R.affecterType(type)
        R.affecterListeNœuds(départ, listeNœudsDépart)
        R.affecterListeNœuds(arrivée, listeNœudsArrivée)
                                                        // côté nœuds
        Pour chaque nœud N de la listeNœudsDépart
            N.ajouterLiaison(L)
        Fin Pour
        Pour chaque nœud N de la listeNœudsArrivée
            N.ajouterLiaison(L)
        Fin Pour
    Fin
```

Méthode 11 : ajouterRelation (classe Treillis)

2.1.5 Application d'un schéma de liaison

Les schémas de liaisons permettent par la suite de gérer plusieurs phénomènes tels que :

- les transcodages et transcriptions de caractères
- la fusion des deux parties des balises doubles
- les schéma linguistiques

L'application d'un schéma de liaison se fait en trois étapes :

- recherche d'applicabilité d'un schéma de liaison
- création d'un ou plusieurs nouveaux nœuds
- ajout d'une relation de liaison

```
méthode appliquerSchéma(schéma, listeNœudsDépart) : listeNœudsCréés
Début
    test ← vrai
    listeRef ← schéma.obtenirListeDépart
    Si (nombreEléments(listeRef) == nombreElements(listeNœudsDépart)) Alors
        Tant que (test)
            Ni ← suivant(listeRef)
            Mi ← suivant(listeNœuds)
            [test
                ET (Mi.obtenirType == Ni.obtenirType)
                ET ( suivant(schéma.obtenirListeConditions)(Mi) ) ]
```

```
            Fin Tant Que
        Fin Si
        Si (test) Alors
            L ← nil
            Lt ← schéma.listeTypeArrivée
            Ld← schéma.listeDimDonnéesArrivée
            Tant que (t<> nil)
                t ← suivant(Lt)
                D ← suivant(Ld)
                N ← nouveauNœud(t, D.a, D.b)[5]
                L ← L ∪ N
            Fin Tant que
        ajouterRelation(schéma, listeNœudsDépart, L)
        renvoyer L
        Fin Si
Fin
```

Méthode 12 : AppliquerSchéma (classe Treillis)

Le renvoi de L permet de travailler par la suite sur les nœuds de L, pour par exemple les affecter à un treillis, et les remplir.

[5] D.a représente le nombre de lignes du tableau des données du noeud, D.b représente le nombre de colonnes.

2.2 Algorithmes de construction d'une structure TELAM

Introduction

Dans le contexte de la traduction fondée sur la mémoire, la structure TELAM sert à la fois à représenter un segment source du document et à supporter la construction du segment cible correspondant, mais aussi à représenter les bi-segments de la mémoire (cf. Chapitres 8).

La méthode est la même pour la construction d'une structure représentant un segment source de document, ou un segment source ou cible de bi-segment de la mémoire, car leur structure d'échange est un segment XML. La série d'algorithmes suivants montre comment construire une telle structure TELAM sur la base d'un segment XML. La méthode servant à construire la partie cible par analogie sur le modèle du bi-segment de la mémoire ne sera pas abordée ici, mais plus tard (chapitre 8).

Procédures externes ou internes, données externes et internes

La construction de la structure TELAM consiste essentiellement en celle de ses différents treillis et liaisons. Les treillis sont eux-mêmes bâtis grâce à la création de nœuds et d'arcs.

L'élaboration de ces différentes parties de TELAM se fait sur la base d'informations externes à la structure TELA (comme les parties construites sur l'analyse du segment XML), ou sur l'utilisation de données internes déjà présentes dans la structure (comme le transcodage des caractères déjà représentés par des nœuds de caractères dans le treillis).

La manipulation de ces données internes ou externes peut être réalisée par des procédures externes (un module de lemmatisation par exemple), ou suivant des règles internes, supportées dans des schémas de liaison. Ainsi chaque chemin de nœuds sera le résultat de l'application :

- soit de procédures externes sur les données externes (type a)
- soit de procédures externes sur les données internes (type b)
- soit de procédures internes sur des informations internes (type c)

Méthode

La structure TELAM peut être calculée en cinq phases :

- la première phase regroupe la construction des étages 1, 3, et 4, et est basée sur le parcours séquentiel du segment XML (procédure de type a)
- la deuxième phase consiste en la construction de l'étage 2 après segmentation de l'ensemble des caractères en mots.
- puis la lemmatisation de la chaîne de mots s'effectue dans une troisième phase, pour obtenir une première analyse linguistique nécessaire à la construction de l'étage 5.
- dans la quatrième phase, la recherche, le choix et l'insertion de schémas linguistiques permettent de construire l'étage 6.
- une cinquième phase permet d'insérer les nœuds de termes extraits de la base terminologiques à l'étage 7.

Certaines phases intermédiaires peuvent intervenir. Entre les premières et secondes phases, par exemple, l'étage 1 des caractères peut être modifié par l'application de schémas de transcription ou de transcodage de caractères. D'autres étapes seraient possibles, menant à la construction de nouveaux étages, ou la modification de certains.

Variables globales

Un certain nombre de variables globales vont servir aux algorithmes suivants ; en voici quelques unes :

- dernierNœud(i) : dernier nœud créé pour un l'étage i
- nbNœuds(i) : nombres de nœuds de l'étage i

Présentation des informations provenant du segment

La construction de la structure TELAM se base dans un premier temps, sur les informations provenant du segment XML (données externes). On suppose que les informations contenues dans le segment SS sont regroupées dans un tableau SST par la procédure segmentVersTableau(SS) présentée plus haut. Chaque élément du segment (texte, référence d'entité, balise double ou monobalise) y occupe une ligne accompagné de ses attributs de SST(i, 0) à SST(i, 0). C'est donc une procédure externe.

2.2.1 Lecture du segment XML

Les étages 1, 3 et 4 peuvent être construits dans un premier temps par la lecture directe du segment XML. Soit SS un segment XML. la procédure segmentVersTableau(SS) transforme SS en SST, un tableau d'éléments. SST(0, 1) et SST(0, 2) et SST(0, 3), les premier et deuxième et troisième éléments du tableau, donnent respectivement le nombre total de caractères, d'éléments, et de balises du segment. Le nombre de nœuds horizontaux ne peut dépasser SST(0).

Les procédures suivantes sont le plus souvent explicites, et quelques remarques dans le code, ou en notes de bas de page complètent les informations.

```
Procédure segmentVersTELAM134 (SS : string) : TELAM
Début
    Si (Non segmentNul(SS))
                                                    // paramètres de départ
        SST ← segmentVersTableau(SS)[6]
        S ← nouvelleTELA
        T1 ← nouveauTreillis
        T3 ← nouveauTreillis
        T4 ← nouveauTreillis
        S.ajouterTreillis(1, T1)
        S.ajouterTreillis(3, T3)
        S.ajouterTreillis(4, T4)
        s1 ← nouvelleEstampille(1, 1, (SST(0, 1)+1))
        T1.sup.AffecterIdentificateur(s1)
        dernierNœud(1) ← s1
        s3 ← nouvelleEstampille(3, 1, (SST(0, 1)+1))
        T3.sup.AffecterIdentificateur(s1)
        dernierNœud(1) ← s3
        s4 ← nouvelleEstampille(4, 1, (SST(0, 1)+1))
        T4.sup.AffecterIdentificateur(s1)
        dernierNœud(1) ← s4
                                // Début parcours tableau d'information du segment
```

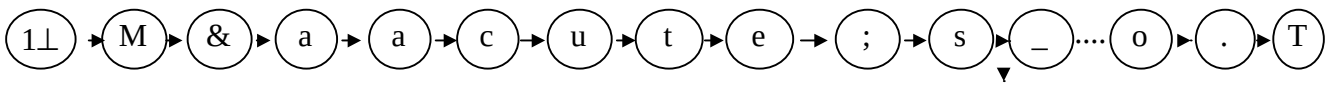
[6] Il est bien sûr possible de ne pas passer par un tableau, et de travailler directement sur le segment SGML. cette solution est choisie pour des raisons explicatives.

```
        Pour i=1 à SST(0, 2)

                                    // Construction des nœuds de caractères l'étage 1
            Si (tagType(SST(i, 5))="1") Alors
                rang00← (SST(i, 1), SST(i, 2))
                rang01 ← (SST(i, 3), SST(i, 4))
                Pour (chaque caractère c de SST(i, 5))
                    N ← nouveauNœud(t1, 1, 1)
                    T1.insérerNœud(N, ((dernierNœud(1)), (1)), ((T1.sup), (1)),
                                                            rang00, rang01, (1, 1, c))
                    N.affecterCoordonnées(2) ← rang00
                    N.affecterCoordonnées(3) ← rang01
                    dernierNœud(1) ← N.obtenirIdentificateur
                    rangT ← rangT + 1
                Fin Pour
            Fin Si

                                    // Construction des nœuds de balises doubles de l'étage 3//
            Si ((tagType(SST(i, 5))="2") ou (tagType(SST(i, 5))="3")) Alors
                rang00← (SST(i, 1), SST(i, 2))
                rang01 ← (SST(i, 3), SST(i, 4))
                N ← nouveauNœud (t3, 1, 1)
                T3.insérerNœud(N, (dernierNœud(3)),(ta1)), ((T3.sup),(ta1)),
                    SST(i, 1), SST(i, 3), [(1, 1 SST(i, 5)), (1,2, rang00), (1,3,rang01)])
                    N.affecterCoordonnées(2) ← rang00
                    N.affecterCoordonnées(3) ← rang01
                dernierNœud(2) ← N.obtenirIdentificateur

                                    // Construction des nœuds des monobalises de l'étage 4//
            Si (tagType(SST(i, 5))="4") Alors
                rang00← (SST(i, 1), SST(i, 2))
                rang01 ← (SST(i, 3), SST(i, 4))
                N ← nouveauNœud (t3, 1, 3)
                T1.insérerNœud(N, ((dernierNœud(4)), (ta1)), ((T4.sup), (ta1)),
                    SST(i, 1), SST(i, 3), [(1, 1 SST(i, 5)), (1,2, rang00), (1,3,rang01)])
                N.affecterCoordonnées(2) ← rang00
                N.affecterCoordonnées(3) ← rang01
            Fin Pour
        Fin Si
Fin
```

Procédure 1 : Prise en compte des données du segment XML

Voici une illustration de l'effet de cette première procédure sur un segment protoXML, pour la dicton espagnol "Más vale pájaro en mano que ciento volando" :

```
<s>M&aacute;s_vale_<hi1>p&aacute;jaro</hi1>_en_mano_que_ciento<mobj/>_volando.</s>
```



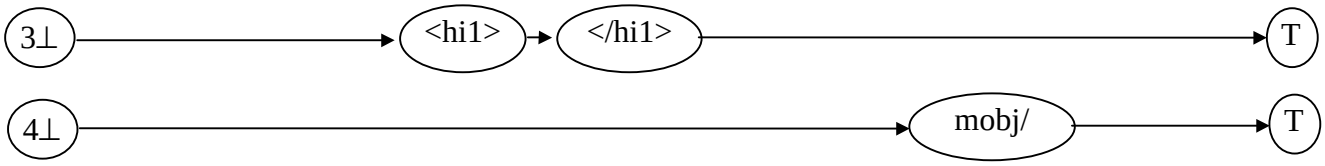


Figure 2 : extraction des informations du segment XML vers une structure TELAM

Tous les caractères ne faisant pas partie de balises sont représentés au premier étage par un nœud. le second étage n'existe pas encore. Les deux balises du couple de doubles balises sont représentées chacune par un nœudsur l'étage trois. Les monobalises (une seule ici) sont représentées à l'étage 4.

2.2.2 Transcodage et Transcription de caractères (phase intermédiaire)

La gestion du transcodage et de la transcription (voir chapitre 5) peut être assurée soit par des applications externes (transcodeurs par exemple), ou par le mécanisme des applications de *schéma de liaison* sur l'étage 1 entraînant la construction de *relations de liaison*. Le transcodage est alors vu comme l'application de schéma de liaisons binaires créant pour chaque nœud de l'étage 1, un nouveau nœud. L'ensemble des nouveaux nœuds forment un chemin différent (un sous-étage de plus). La figure ci-dessous illustre un transcodage des lettres de l'alphabet vers un codage numérique de l'alphabet.

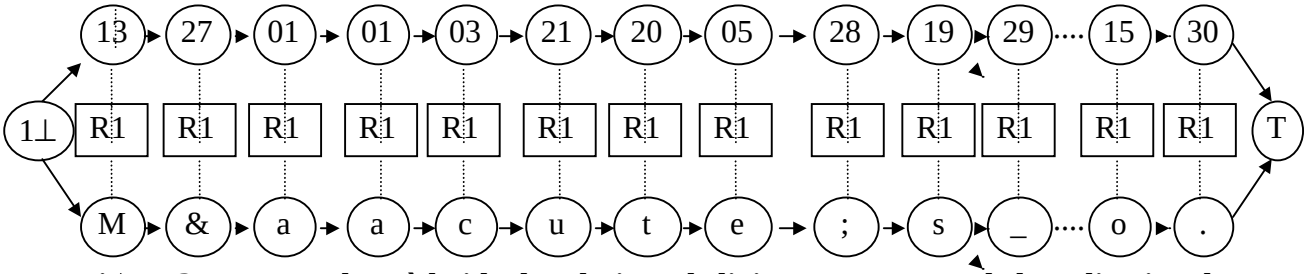


Figure 3 : Transcodage à l'aide de relations de liaisons provenant de l'application de schémas de liaisons

Le transcodage est une opération qui s'applique à tous les nœuds de caractères. Cela peut s'avérer une opération gourmande en mémoire pour les nombreuses relations de la figure ci-dessus, et l'on peut s'en tenir à la construction d'un sous-étage supplémentaire par une application extérieure, sans en garder la trace exprimée par des relations de liaisons.

La transcription qui peut être systématique, ne s'applique parfois que sur certains groupes de caractères. C'est en particulier le cas pour, par exemple la transcription des diacritiques sous un codage ASCII 7 bits. Dans ce cas l'application de schémas sur un ou plusieurs nœuds de l'étage 1 vers un ou plusieurs nœuds d'un sous-étage de l'étage 1, permet de gérer cette transcription.

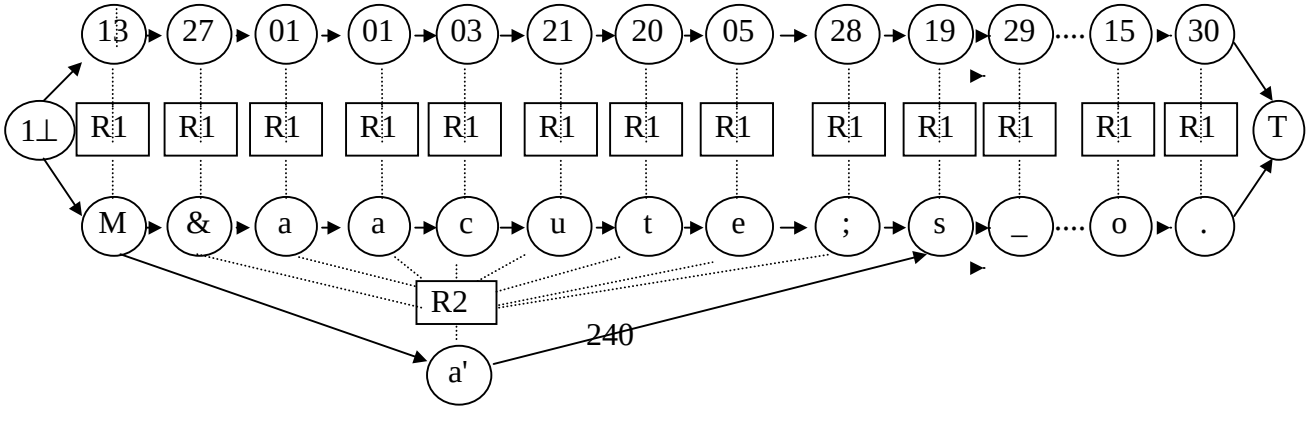


Figure 4 : Transcription par application d'un schéma de liaison instancié en une relation de liaison.

Cette procédure est assurée par la recherche de schémas de relations applicables, et l'application de ces schémas aux nœuds de caractères. On suppose que l'on possède une liste de schémas potentiellement applicables à des nœuds de caractères. Voici cette procédure :

```
Procédure SchemaSurCaractères(TTT : TELAM, listeSchémas) : TELAM
Début
    T1 ← TTT.obtenirTreillis(1)
    Pour chaque schémade listeSchémas
        lg ← nbEléments(s.obtenirListeTypesDépart)
        Pour (chaque liste L de nœuds de longueur lg)
            appliquerSchéma(s, L)
        Fin Pour
    Fin Pour
Fin
```

Procédure 2 : Application de schémas de liaison aux nœuds de caractères.

2.2.3 Procédure de fusion des balises de début et de fin des couples de balises doubles (procédure intermédiaire)

Une balise double est composée de deux parties : la balise de début, du type <B>, et la balise de fin, de type </B>. Un parcours du segment (ou plus exactement du tableau extrait du segment) permet dans un **premier temps** de construire une liste de balises de type <B> et </B> formant un premier chemin. La représentation choisie au chapitre 5 regroupe ces deux parties en un seul nœud. La **seconde** opération sur ces balises est donc un regroupement, créant un second chemin. Parmi les nœuds de balises de ce deuxième chemin, certaines contiennent une information du même type. Une **troisième** phase consiste à concentrer l'information portée par les balises du même type en une seule balise, pour créer un troisième chemin dans le treillis des balises doubles (étage 3).

Le nœud final possédera, pour chaque instance de couple de balises représentées, les informations suivantes :

- rang00 de début : numéro d'occurrence dans le segment XML du premier caractère de la balise de début
- rang00 de fin : idem pour la balise de fin
- rang01 de début : numéro d'occurrence de la balise de début, parmi les éléments du segment XML
- rang01 de fin : similaire pour la balise de fin

Voici une illustration de l'ensemble de ces trois procédures :

<s> <B1> <B2> </B1> </B2> <B3> </B3> <B2> </B2><B1> </B1> </s>

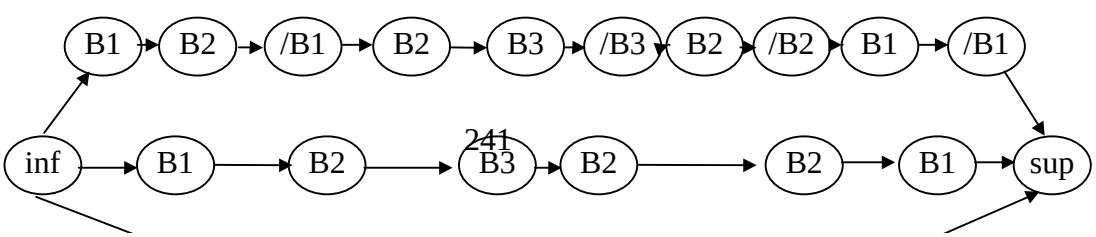


Figure 5 : Les trois étapes de la construction de l'étage 3

Méthode de calcul

```
Début
                                                    //Phase 1
    Pour chaque balise du segment
        Si c'est une balise double de départ <Bi> ou d'arrivée </Bi>
            créer un nœud N sur le chemin 1
        Fin Si
    Fin Pour
                                                    //Phase 2
    Pour chaque balise <Bi> du chemin 1
        Pour chaque balise suivante
            Si c'est </Bi>
                Créer une balise au chemin 2
            Fin Si
        Fin pour
    Fin Pour
                                                    //Phase 2
    Pour chaque balise <Bi> du chemin 2
        Pour chaque balise suivante <Bj>
            Si <Bi> et <Bj> sont du même type
                Créer une balise de ce type contenant les deux informations au chemin 3
            Fin Si
        Fin pour
    Fin Pour
Fin
```

On notera qu'en phase 1 l'on est obligé de chercher la balise correspondante car il est nécessaire de savoir quand elle se termine pour la suite. Cette méthode semble naturelle, mais elle est en O(n3), comme le prouve le calcul de complexité donné au paragraphe III.B.3..

Ceci étant, il y a rarement autant de balises que de mots dans les segments à traiter, ce qui réduit le temps de calcul.

2.2.4 Création de nœuds de mots

Pour créer les nœuds de mots, il faut appliquer un algorithme de séparation de mots sur la chaîne de caractère interprétée, c'est-à-dire récupérée par le parcours du chemin le plus récent du treillis des mots (qui reflète à la fois le transcodage et la transcription). La récupération de cette liste de mots se fait grâce à l'algorithme. On se reportera au paragraphe parlant de l'étage 2 du chapitre 5, pour se rappeler la méthode 7 présentée plus haut d'extraction des chemins.

2.2.5 Création de nœuds de lemmes de l'étage 5 (phase 3)

Les lemmatiseurs peuvent s'appliquer de plusieurs façons : soit il s'agit d'un lemmatiseur donnant toutes les solutions possibles pour une forme donnée (on l'appellera Lemmat1), et il n'a besoin en entrée que de cette forme (comme WordNet) ; soit il effectue une analyse entraînant une désambiguisation partielle et demandent alors l'ensemble des mots de la phrase à analyser (Lemmat2, comme les analyseurs d'ARIANE du GETA).

Dans le premier cas, il suffit d'appliquer l'algorithme Lemmat1 à chaque mot des nœuds de l'étage 2. Plusieurs solutions apparaissent en général par mot, et il faut alors créer autant de nœuds de lemmes à l'étage 5 que de solutions, pour un mot donné.

Dans le second cas, il faut d'abord extraire la chaîne (par l'algorithme de la méthode 7), puis appliquer le lemmatiseur à toute la chaîne de mots. Le lemmatiseur rend alors pour chaque mots les possibilités (moins que dans le premier cas, en général).

Dans les deux cas, les lemmatiseurs renvoient un ensemble de lemmes portant en général chacun les informations suivantes :

- lemme i,
- attributs lexicaux i
- attributs sémantiques i

2.2.6 Recherche et Insertion de schémas linguistiques pour l'étage 6 (phase 4)

Les schémas linguistiques peuvent s'appliquer soit à l'étage des mots, soit à l'étage des lemmes. Cet algorithme traite le cas de l'application d'un schéma de relation de type R26 sur l'étage des mots.

Il est donc supposé qu'un schéma S (voir p 16 du chapitre 6) est disponible. L'algorithme suivant prend en entrée une structure TELAM, et rend cette structure modifiée T1, plus une structure cible T2 dont seul l'étage 6 existe à ce niveau.

```
Procédure nœudsPivots(TTT: TELAM, S : schéma de liaison) : TTT1, TTT2 : TELAM
Début
TTT1 ← TTT
TTT1.ajouterTreillis(6)
TTT2 ← nouvelleTELAM
TTT2.ajouterTreillis(6)

L1 ← TTT1.extraireListeNœuds(2, "dernier")
L2 ← S.obtenirListeNœudsDépart

N2 ← premier(L2)
Pour chaque nœud N1 de L1
    Si ((S.conditionsDépart(N2)) (N1)) ← vrai) Alors
        Pour ((chaque nœud M1 suivant de L1) et (chaque nœud M2 suivant de L2))
            Si ((S.conditionsDépart(N2)) (N1)) ← vrai) Alors
                Appliquer (S.conditionsArrivée)
            Fin Si
        Fin Pour
    Fin Si
Fin Pour
Fin
```

Procédure 3 : Création des nœud d'expression pivot

2.2.7 Insertion de nœuds de terminologie (phase 5)

Une procédure Termes(), de recherche de terme dans la base de données donne, à partir d'une forme de l'étage 2. Il suffit alors de parcourir l'étage 2, Termes() donne les termes correspondant à chaque mot (en général un seul), et la construction de l'étage 7 des termes se fait en créant pour chaque nœud de mot, autant de nœuds que de termes trouvés. Cela ne pose pas de difficultés particulières.

3. Complexité

3.1 Complexité statique

3.1.1 Hypothèses de calcul

Estimation du nombre de nœuds

Au sein d'un segment XML, le nombre d'entités non textuelles dépasse rarement le nombre d'éléments textuels. En prenant comme référence le nombre d'éléments textuels m (des "mots" en général, nous prendrons comme hypothèse excessive que le nombre total n d'entités vérifie: n = 2m.

Estimation de l'espace nécessaire pour représenter les types de données élémentaires

Dans la suite, nous évaluons la complexité statique des structures en fonction de types de données élémentaires. Ces types sont l'entier, le réel, le caractère d'Unicode Les coûts utilisés en octets sont :

type	entier	réel	caractère	identificateur
coût (en octets)	1	2	2	10

Un identificateur est supposé être une chaîne de 10 caractères d'un octet.

Codage d'un nœud

Un nœud comporte toujours les mêmes champs. D'un étage à l'autre de la structure, les types varient peu. On peut considérer, au nombre de coordonnées d'échelles linéaires près, qu'un nœud comporte six champs de poids constant. Seul le champ des données varie nettement. Le tableau des champs d'un type de nœud de TELA (standard) est repris ici, avec le coût moyen du stockage d'un champ.

type standard	intitulé	donnée concrète	type de donnée	coût (octets)
Champs	identificateur	estampille	liste de chaînes de caractères	10
	typeNœud	t1	entier	1
	coordonnées	rangT, rang00, rang01	liste de couples d'entiers	3 x 2 x 1 = 6
	données	?	liste d'une liste d'un élément	?
	arcsPrécédents	liste des arcs qui arrivent du nœud	liste d'identificateurs d'arcs	20 (15)
	arcsSuivants	liste des arcs qui partent du nœud	liste d'identificateurs d'arcs	20 (15)
	listeLiaisons	liste des liaisons concernant ce nœud	liste d'identificateurs de liaisons	10
	Total sans données			67 ≈ 70 (40)

Il est considéré qu'en moyenne 2 (par excès ; 1.5 : par défault) nœuds arrivent ou repartent du nœud courant, et qu'une liaison est attachée à un nœud.

3.1.2 Complexité statique d'une structure TELA

Complexité de l'étage 0 : le segment XML

Cet étage ne comporte pas de nœuds, mais seulement un segment source (ou cible) de 2m éléments. On suppose qu'en moyenne une entité comporte 10 caractères, et que le code de caractère est au plus de 2 octets par caractères. Son "poids" est donc équivalent à :

2m x 10 x 2 octets = 40m octets

Nœuds de l'étages 1 (caractères)

Ce nœud comporte les champs de base plus un champ représentant un caractère. Le champ représentant le caractère prend 2 octets. Une entité est considérée de longueur moyenne 10 caractères. Un nœud de caractère coûte donc 70 (40) + 2 x 10 = 90 (60) octets. Il y a 2m entités dans le segment, ce qui donne en tout pour un chemin complet de caractères l'étage 1 : 2m x 90 (60) = 180m (120m) octets. En comptant 1 (1.5) encodage ou retranscription pour un étage en moyenne, cela donne 2 (1.5) listes, soit :

2 x 180m (1.5 x 120m) = 360m (180m) octets

Nœuds de l'étages 2 (mots)

Les nœuds de l'étage 2 représentent les mots. Un mot est supposé être composé de 10 caractères en moyenne, soit 20 octets de données par nœud. Cela donne en tout 70 (40) + 20 = 90 (60) octets par nœud de mot. Comme il a été considéré qu'un segment comporte m nœuds,

90m (60m) octets

sont nécessaires par chemin complet de nœuds de mots. Il n'y a généralement pas de chemin supplémentaire.

Nœuds de l'étage 3 et 4 (balises doubles et monobalises)

Ces nœuds représentent les entités complémentaires aux mots dans le segment XML. Il y en a donc un nombre m. Chaque nœud prend 70 (40) octets de champs communs. Les données d'un nœud de balise comprennent :

- le code XML de la balise : 10 caractères en moyenne soit 20 octets
- les rang00 et rang01 : 2 x 2 = 4 entiers, soit 4 x 1 = 4 octets

Un nœud de balise prend donc en moyenne 70 (40) + 20 + 4 = 94 (64) octets. Comme il y a m balises, un chemin complet de l'étage 3 plus un chemin complet de l'étage 4 prennent ensemble 94m (64m) octets. Le regroupement des balises doubles en deux passes forme deux chemins comportant l'un la moitié des balises, et l'autre moins de la moitié. On peut donc considérer qu'un deuxième chemin comportant le même nombre de nœuds est créé. Au niveau des monobalises, un deuxième chemin est rarement créé. Dans une approximation par excès, on peut donc estimer que les nœuds des étages 3 et 4 coûtent :

2 x 94m (64m) = 188m (128m) octets

Nœuds de l'étage 5 (lemmes)

Le poids de l'enregistrement de l'information linguistique se compose en première approximation pour chaque lemme de la liste suivante : lemme, forme, catégorie

grammaticale, genre, nombre, info 1, info 2. Cela donne en octets (20 + 20 + 3 + 1 + 1 + 2 + 3). Soit 50 octets. Un nœud de lemme prend donc 70 (40) + 50 = 120 (90) octets. Il y a m mots, supposant que 2 lemmes sont donnés en moyenne par forme, on obtient :

2m x 120 (90) = 240m (180m) octets.

Nœuds de l'étage 6 (nœuds d'expression pivot)
Les données contenues dans un nœud pivot sont la chaîne du mot pivot, et les conditions sur la partie. La chaîne, apparentée à un identificateur prendra 10 octets. Les conditions supposées exprimées par 10 égalités de deux identificateurs consomment : 10 x (10 + 1 + 10) = 210 octets. Un nœud peut donc être représenté avec 70 (40) + 10 + 210 = 290 (260) octets. Il y a au plus trois nœuds, par expression, soit :

3 x 290 (260) = 870 (780) octets.

Nœuds de l'étage 7 (termes)
Un nœud de terme ne comporte que le terme et son domaine en général, soit 40 octets de données. Cela donne 70 (40) + 40 = 110 (80) octets par nœud. Comme il y a en moyenne un nœud de terme provenant de la base terminologique par mot, cela donne :

110m (80m) octets

Complexité statique d'une structure TELA
L'ensemble de la structure a donc une complexité statique équivalente à la somme des coûts ci-dessus. Cela donne :

au maximum : 40m + 360m + 90m + 188m + 240m + 870 + 110m = 1898 m au minimum : 40m + 180m + 60m + 128m + 180m + 780 + 80m = 1448 m

Pour un segment de 10 mots, cela donne donc :

maximum : 19 Ko minimum : 14,5 Ko

3.2 Complexité dynamique

3.2.1 Construction des étages 1, 2, 4, 5, et 7

Les algorithmes de construction des étages 1, 2, 4, 5, et 7 ne requièrent qu'un parcours du segment XML. Les modules linguistiques appelés (segmentation de mots, lemmatiseur, etc.) ont leur propre coût, mais au niveau de la structure, la construction de ces étages reste linéaire par rapport au nombre d'éléments du segment XML.

3.2.2 Procédure de fusion des balises de début et de fin des couples de balises doubles

On se reportera au schéma donné avec l'algorithme au paragraphe II.B.4. pour référence de la structure de cet algorithme. Le premier parcours est linéaire. S'il y a n balises, il coûte donc n boucles, c1 = n.

La seconde phase demande cependant à chaque phase un double parcours du segment, a priori. Plus exactement, s'il y a n balises sur le chemin 1, pour chaque balise de début <B1>, on parcour (n-1) balises au plus, soit en tout :

$$c2 = \sum_{i=1}^{n} i(i-1) = \sum_{i=1}^{n} i^2 + -\sum_{i=1}^{n} i = (\frac{1}{3} n^3 + \frac{2}{3} n) + (\frac{1}{2} n^2 + \frac{1}{2} n) = \frac{1}{3} n^3 + \frac{1}{2} n^2 - \frac{7}{6} n$$

soit : $c2 = \frac{1}{3} n^3 + \frac{1}{2} n^2 - \frac{7}{6} n$

La troisième phase procède de la même combinatoire, mais le nombre d'éléments de départ est n/2. Cela donne donc :

$$c3 = \frac{1}{3}\left(\frac{n}{2}\right)^3 + \frac{1}{2}\left(\frac{n}{2}\right)^2 - \frac{7}{6}\frac{n}{2} = \frac{1}{24} n^3 + \frac{1}{8} n^2 - \frac{7}{12} n$$

Au total, la complexité de l'algorithme est donc :

$$C = c1 + c2 + c3 < n + (\frac{1}{3} n^3 + \frac{1}{2} n^2 - \frac{7}{6} n) + \frac{1}{24} n^3 + \frac{1}{8} n^2 - \frac{7}{12} n = \frac{9}{24} n^3 + \frac{5}{8} n^2 - \frac{9}{12} n$$

D'où une complexité en $O(n^3)$ pour cet algorithme naïf.

3.2.3 Conclusion

La concentration des informations au niveau des balises doubles n'est pas forcément une nécessité pour la bonne utilisation de la structure TELAM, et l'on peut décider de n'utiliser que le premier chemin dans les conditions standards d'utilisation de TELAM. De façon générale, il est donc possible de construire une structure TELAM avec des algorithmes linéaires, ce qui fait de TELAM une structure attractive pour son faible coût.

3.3 Expérimentations

Une première série d'expérimentations a été menée sur la base d'un prototype LISP. Les temps de réponses de construction des treillis sont effectivement linéaires comme le prouvent les résultats suivant.

Par exemple, la construction de la structure TELAM de cinq étages associée à un segment XML comprenant dix mots et dix balises, s'effectue en 0.04 secondes.

Conclusion

Avantages de la structure TELAM

- souplesse dans la transcription des objets non-textuels comme le format ou les objets imbriqués : pas besoin de table de l'ensemble des balises possibles : seule la syntaxe est importante : balises doubles et monobalises.

Perspectives

- La gestion des segments les uns par rapport aux autres, et en particularité l'extraction-ré-insersion des segments de traduction n'a pas été abordée. Il est important de gérer ceci au niveau de l'outil de traduction. Il serait intéressant de voir si une instance de TELA permettrait de faire ceci (cf exemple de l'application JAVA).
- conservation de l'information concernant l'analyse du segment :
  Le formalisme TMX est trop simple (il est fait pour être simple, ce n'est pas une critique !) pour pouvoir garder l'information provenant de l'analyse linguistique du segment par exemple (des arbres au moins). Garder cette information ne sert pas à grand chose entre logiciels qui ne peuvent se servir de la même information. Il peut cependant être intéressant de pouvoir conserver cette structure dans le cas ou, par exemple, une partie de l'analyse est faite sur un serveur distant. Dans ce cas, une structure plus complète du style de celle qui sera proposée au chapitre suivant pour représenter les données internes, pourrait gagner à être utilisée pour les données externes, pour la mémoire.
  Par exemple avec le langage externe de TELA.

Chapitre 8

Similitude

Contenu du Chapitre

Introduction

Trois opérations fondamentales sont utilisées dans le processus de traduction basée sur la mémoire :

- la comparaison du segment source provenant du document à traduire, et du segment source candidat de l'unité de traduction provenant de la mémoire des bi-segments.
- la mise en correspondance des éléments des segments sources et cibles de l'unité de traduction.
- la traduction par analogie qui utilise les deux étapes ci-dessus.

La première opération sert à déterminer la pertinence de l'utilisation de l'unité de mémoire choisie pour traduire le segment source du document, et les points de différence entre ces deux segments, qu'il faut gérer dans la suite de la traduction. Nous présenterons cette notion dans ce chapitre 8. La seconde est une mise en correspondance des éléments du segment source de l'unité de mémoire avec les éléments du segment cible de cette unité. Ceci sera abordé au chapitre 9. La troisième est la traduction du segment source par "transfert" des différences trouvées dans la première opération, via les correspondances trouvées à la seconde opération, traitée au chapitre 9.

Ce chapitre précise la notion essentielle de similarité qui guide la première opération sans laquelle les deux autres ne pourraient être appliquées. Il décrit formellement la notion de segment en se basant sur [Schreider 1975] de façon à pouvoir appliquer la notion de distance entre deux segments. La distance d'édition est alors proposée [Wagner & Fisher 1974], et l'on montre comment elle s'adapte à la recherche de similarité entre deux segments. Enfin on montre le lien avec les structures TELAM, et l'utilisation pratique de ces notions est développée.

1. Position du problème

1.1 Notion intuitive de similitude

Un segment S1 de document étant donné, le principe des outils de traduction fondée sur la mémoire est de trouver dans la mémoire une unité de traduction (S2, T2) telle que S2 soit similaire à S1. Les deux segments S1 et S2 peuvent être similaires sur plusieurs plans.

Il peut s'agir d'une similarité au niveau des mots (occurrences dans le texte), comme ci-après :

S1 : User logged in, proceed
S2 : User logged off, proceed

La similarité peut aussi mettre en jeu les flexions des mots :

S1 : User logged in, proceed
S2 : Users were logged in, proceed

On a également des similarités syntaxiques :

S1 : User logged in, proceed
S2 : Process passed by, continue

Ou encore sémantiques :

S1 : User logged in, proceed
S2 : User has been connected, you are through.

Il est donc nécessaire de trouver un moyen de formuler et de calculer cette similarité de façon automatique.

1.2 Approches actuelles

Introduction

Les outils de Traduction Fondée sur la Mémoire et les systèmes de Traduction Automatique Fondée sur l'Exemple, utilisent différents moyens de représenter et d'utiliser cette similarité. Nous présentons maintenant le principe général des techniques employées dans les OTFM1g, et deux exemples de techniques mises en œuvre dans deux prototypes de Traduction Automatique Fondée sur l'Exemple.

1.2.1 Recherche de similitude sur un flot hétérogène d'éléments

Les techniques des OTFM1g consistent à comparer les deux segments S1 et S2 au niveau de leur représentation primaire. Voici deux segments S1 et S2 dont la représentation primaire est codée en XML :

S1 : <s>User <idx id="001"/> logged <hi1>in</hi1>, proceed.</s>
S2 : <s>Users logged off, proceed.</s>

Il est nécessaire que les deux segments soient codés de la même façon. Cela n'est pas toujours vrai dans les OTFM1g, par exemple lorsque le format d'enregistrement du document dont est extraite la mémoire (S2) est différent du format d'enregistrement du document dont est extrait S1, et que la représentation interne n'est pas standard. Nous considérons ici que S1 et S2 sont codés de la même façon. Sous cette hypothèse, les techniques des OTFM1g consistent globalement à manipuler des fonctions de coût gérant plusieurs aspects des chaînes de caractères représentant S1 et S2, comme ceux qui suivent :

- comparaison des longueurs
- comparaison des mots présents ou absents
- comparaison du nombre de balises d'objets ou de mise en forme

Il est clair que les fonctions de coût fondées sur la comparaison des longueurs des représentations primaires de S1 et S2 mènent à des aberrations du fait de l'hétérogénéité de cette représentation : il suffit qu'il y ait un index de plus dans S1 pour que la fonction de coût change ! D'autre part, à supposer que seuls les caractères des mots du segment soient représentés (pas de balises), une comparaison fondée uniquement sur les caractères ferait apparaître la même différence entre S1 et S2 qu'entre S1 et S3, où :

S3 : <s>Users bogged off, proceed.</s>

Cela est bien sûr non désirable. Ces techniques introduisent donc du **bruit** dans la recherche de similarité. Ce bruit semble provoqué par :

- l'hétérogénéité de la structure
- le choix de la fonction de coût

D'autre part, les similarités syntaxiques et sémantiques ne sont pas détectées par les OTFM1g, d'où un **silence**. Cela est dû à ce qu'il n'y pas d'analyseurs linguistiques utilisés à ce niveau[1].

Il semble donc nécessaire d'une part d'appliquer de telles fonctions d'évaluation de la similitude sur chaque type de données, et pas sur un ensemble hétérogène, et d'autre part d'utiliser des données linguistiques à travers des "mesures" adaptées. Nous montrons maintenant quelques techniques utilisées dans certains prototypes de recherche.

1.2.2 Alignement de segments dans PanEBMT via des n-grammes

Introduction

Le système Pangloss [Niremburg 1995], conçu par le S. Niremburg de l'Université Carnegie Mellon, comprend trois modules. Un des modules est un système à transfert basé sur des règles, l'autre sur une représentation conceptuelle pivot (KBMT), et le troisième est basé sur la traduction fondée sur l'Exemple (PanEBMT). Les trois modules de traduction tournent en parallèle, et leurs résultats sont combinés dans une carte au niveau de sous-séquences de la phrase. Le choix de la meilleure traduction est effectué par un modèle statistique.

[1] Nous avons montré au chapitre 4 que Eurolang Optimizer par exemple possède un lemmatiseur. Mais il ne s'en sert que pour la recherche de terminologie.

Le principe du module fondé sur l'exemple consiste à traduire un texte d'entrée par recherche de similitude avec les parties sources d'un corpus bilingue [Brown 1996] et [Yang 1997]. Nous présentons cette technique ci-après.

Données

Les connaissances de base du système sont :

- *un corpus bilingue indexé*

C'est une collection de bi-textes (source et cible) alignés au niveau des phrases. Les mots et la ponctuation sont isolés, mais pas lemmatisés. Le texte est indexé : pour chaque item différent, l'index comporte une liste des occurrences de cet item. Une occurrence est caractérisée par l'identifiant de la phrase dans laquelle elle se trouve, plus le numéro de la place qu'elle occupe dans la phrase. Si un mot fait partie d'une des listes de mots particuliers du fichier de configuration (cf. plus bas), il est remplacé par sa classe (Ex : "5" est indexé par "<number>"). L'index est conçu pour permettre une incrémentation aisée.

- *un glossaire bilingue*

Il s'agit d'un dictionnaire bilingue de formes de mots.

- *une liste de racines et synonymes cibles*

Cela permet d'améliorer la recherche de correspondance entre les mots.

- *un fichier de configuration comportant des listes de mots particuliers*

Il s'agit de listes de mots comme celle des jours de la semaine, des chiffres et nombres, etc.. Ils permettent de généraliser un item en remplaçant un élément de cette liste par le nom de la liste, comme expliqué plus haut.

Processus

Une séquence est une suite connexe de mots (n-grammes). L'analyse de l'entrée se fait par recherche de séquences, et par leur comparaison avec les séquences relatives dans la partie source du corpus bilingue.

Soit une phrase d'entrée. Soit un mot "mot_i" de rang i dans cette phrase d'entrée. Soit la liste L_i des séquences incluant une occurrence de ce mot dans une des expressions sources du corpus bilingue. Soit F_{i-1} l'ensemble des séquences incluant le mot précédent. La procédure de recherche des séquences suit les étapes suivantes :

- s'il existe une séquence dans L_i qui étend une séquence de F_{i-1}, cette séquence est gardée dans F_i.
- sinon, s'il existe une séquence qui forme un bi-gramme avec mot_{i-1}, cette fréquence est incluse dans F_i.

On obtient ainsi l'ensemble E des séquences sources du corpus qui contiennent au moins deux mots consécutifs de la phrase d'entrée. Cet ensemble est élagué de telle façon à ne garder que les cinq dernières occurrences (dans le temps, car elles sont datées) de chaque séquence. Cela donne un ensemble de fréquences E'.

Adaptabilité à la Traduction Fondée sur l'Exemple

La similitude est donc ici évaluée sur les occurrence de mots ou de racines. Il s'agit donc d'une première introduction de connaissance linguistique. Dans ce cas, on notera que le problème de l'hétérogénéité des données ne se pose pas. Au niveau de la Traduction Fondée sur la Mémoire, cette technique n'est donc applicable qu'à des segments S1 et S2 dont les types de données ont été séparés.

Les techniques de recherche de n-grammes ont été largement utilisées dans des travaux de recherche de similarité pour les alignements de phrases ou l'appariement de groupes de mots ou de mots eux-mêmes et fournissent des classes de méthodes applicables pour la recherche de similitude.

1.2.3 La distance sémantique utilisée dans [Sumita et Iida 1992]

Introduction

[Sumita & Iida 1992] présentent la traduction par l'exemple comme la traduction d'une phrase d'entrée par similarité avec une phrase exemple pré-enregistrée. La solution résultante est alors fournie par l'adaptation de la traduction (pré-enregistrée) de la phrase exemple.

Exemple :

Soient les deux enregistrements des phrases suivantes et de leur traduction.

1. houchou wa kikeru → The kitchen knife cuts (le couteau de cuisine coupe)
2. kanojo wa kikeru → She is sharp (Elle est vive, éveillée, intelligente)

(houchou = kitchen knife = couteau de cuisine, kanojo = she = elle)

Ces deux phrases ont la même structure syntaxique en japonais. Cependant, selon leur sens, leur traduction en anglais donne une structure syntaxique anglaise complètement différente. Considérons la traduction de la phrase :

kachou wa kikeru → ?
(kachou = chief = chef)

La recherche de proximité sémantique (dans un dictionnaire adéquat) entre houchou/kachou, et kanojo/kachou montre que kanojo et kachou sont plus proches dans l'ontologie du dictionnaire car ils représentent tous deux des êtres humains. Ainsi le système choisit d'imiter la phrase 2, et la traduction retenue est :

kachou wa kikeru → The chief is sharp

La similarité entre segments ou phrases est calculée à l'aide d'une mesure de similarité, qui fait référence à une classification sémantique dans un dictionnaire.

Ce principe de traduction est illustré par son application à la traduction de syntagmes contenant la particule japonaise "no", réputée difficile pour les systèmes basés sur les règles. Le paragraphe suivant détaille cette implémentation.

Transfert de particules japonaises "adnominales" par Traduction Fondée sur l'Exemple

Introduction

La langue japonaise utilise à foison des particules "adnominales" pour exprimer l'appartenance, le lieu, ou d'autres relations. Une de ces particules est "no", et ses composés (heno, karano, deno,…) [Sumita & Iida 1992].

Exemple : Kyoto no tera ("un temple de Kyoto")

La traduction d'expressions vers l'anglais de type APB où A et B sont des noms, et P est une particule, est difficile par Traduction basée sur les règles. "No" peut en effet signifier différemment "in", "of", "from", sans raison syntaxique ou sémantique évidente. Par contre cette étude montre que la traduction par l'Exemple permet un taux de réussite moyen de 78%, qui peut être simplement amélioré par l'ajout d'éléments dans la base d'exemples.

Méthode

Le système, qui a pour but de traduire les syntagmes du type APB vus plus haut, comprend trois étapes :

- **Analyse morphologique** : le cas précis de cette étude se restreignant aux syntagmes APB, l'analyse syntaxique est absente. Les auteurs précisent cependant qu'en général, elle doit être bien sûr assurée.
- **Transfert fondé sur les Exemples** : des exemples similaires (voir plus bas) sont recherchés dans la base d'exemples, et les cinq plus semblables sont retenus. Puis le meilleur exemple (au sens d'une distance expliquée plus bas) et sa traduction correspondante sont retenus.
- **Génération** : les mots anglais correspondant au syntagme à traduire sont subtitués à l'exemple retenu à l'étape précédente. Cette génération est donc triviale, vu le type de syntagmes concernés.

et deux "bases de données" :

- **base d'exemples** : c'est une suite de paires (syntagmes japonais, syntagme anglais). Dans l'expérience, le corpus utilisé est extrait de celui de la base de données linguistique d'ATR, et comporte 2550 exemples du type APB, totalisant 270.000 mots.
- **thésaurus** : le thésaurus, basé sur dictionaire le "Everyday Japanese" [Ohno et Hamanishi, 1984], est hiérarchisé selon un arbre sémantique non pas binaire mais décimal (chaque nœud a 10 fils), de profondeur 3, et possède 60.000 mots courants.

Distance au sein du thésaurus

Soient deux nœuds A et B représentant deux mots a et b. Soit AB la borne supérieure de A et de B au sens de la hiérarchie arborescente (AB est "le nœud ancêtre commun le plus proche" pour A et B). Si AB est placé au niveau k de la hiérarchie, qui comporte n+1 niveaux, la distance de A à B est :

$$d(A, B) = k/n$$

Formule 1

Exemple:

La distance entre Kaigi (conférence) et Taizai (rester) est de 2/3 (cf. schéma suivant) car la borne supérieure de Kaigi et Taizai est Koudou (action), et se trouve sur à la deuxième couche (en partant du bas), parmi 3 niveaux (en fait quatre, le niveau du bas étant compté comme la couche "0").

1.2.4 Distance entre la phrase à traduire et les exemples de la base

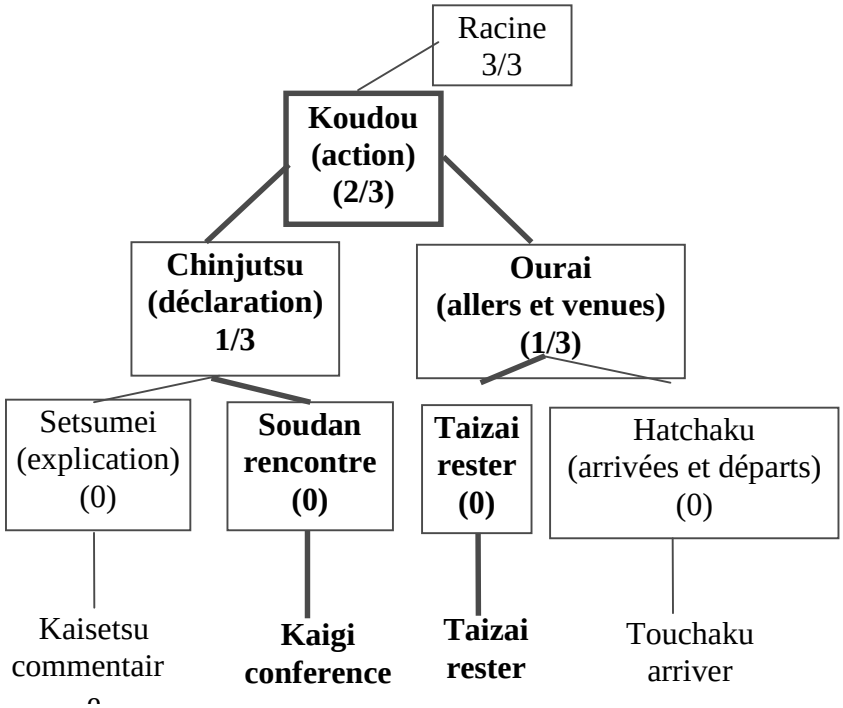


Figure 1 : Distance sémantique dans le dictionnaire "Everyday Japanese".

Soit maintenant un syntagme japonais I ("input"), comportant les mots I_i, et l'exemple japonais E auquel on veut comparer l'entrée I, comportant les mots E_i

La distance d(E,I) entre les deux syntagmes est définie comme suit.

d(E,I) = $\sum_i d(Ii, Ei) * wi$, où
d(Ii, Ei) est la distance définie plus haut pour les mots pleins du thésaurus,
si Ei et Ii sont des particules adnominales, d(Ii, EI) = 0 si Ei=Ii, 1 si Ei≠Ii.
w_i est un poids provenant de la fréquence d'utilisation des mots (cf plus bas).

Le poids w_i

Dans la formule suivante, la deuxième colonne représente le pourcentage f d'utilisation de l'exemple, par rapport au sous-ensemble des patrons retenus pour la phrase I à traduire, cela donne :

$$w_i = \sqrt{\sum_{patrons} (f, pourEi = Ii)^2}$$

Ainsi pour comparer le syntagme $I=I_1I_2I_3$, à $E = E_1E_2E_3$, le w_i doit successivement sommer le carré des fréquences des patrons dans lesquels A est utilisé, puis P (la particule adnominale), puis B. Ce poids a été proposé dans [Stanfill and Wantz 1986].

Exemple :

Le poids pour la comparaison entre "Kyoto no kaigi" et "Timei deno soudan" est :

Patron de tr.	fréquence	Patron de tr.	fréquence	Patron de tr.	fréquence

E_1 = timei (lieu)		E_2 = deno (à)		E_3 = soudan (meeting)	
B in A	12/27	B in A	3/3	B	9/24
AB	4/27			AB	9/24
B from A	2/27			B in A	2/24
BA	2/27			A's B	1/24
….	…			…	…
B to A	1/27			B on A	1/24

Dans ce tableau, les patrons (exemple) de traduction de trois mots japonais pour lesquels le deuxième mot est E_2="deno", sont tous traduits par "in" en anglais. Il n'y a donc pas d'ambiguïté, et la distance est 1.

Par contre, il y a plusieurs possibilités de traduire les tri-grammes commençant par "timei", et la fréquence d'utilisation de chaque possibilité est fournie dans la colonne de droite. La distance est alors une combinaison de ces fréquences, selon la formule ci-dessus :

$$w_1 = \sqrt{(12/27)^2 + (4/27)^2 + ... + (1/27)^2} = 0.49$$

$$w_2 = \sqrt{(3/3)^2} = 1.0$$

$$w_3 = \sqrt{(9/24)^2 + (9/24)^2 + ... + (1/24)^2} = 0.54$$

Application à la Traduction Fondée sur l'Exemple

Cette technique montre comment l'on peut évaluer une similarité sémantique à l'aide d'une distance sémantique fondée sur une classification sémantique des mots. La même remarque que pour la technique précédente s'applique : il est nécessaire d'avoir séparé les données des segments S1 et S2 par types. D'autre part, il est évidemment nécessaire de disposer d'une classification sémantique des mots. Enfin, ces algorithmes sont très coûteux. Cela a d'ailleurs obligé cette équipe à envisager un traitement par processus parallèles, comme cela est expliqué dans [Sumita & Iida 1995]. Nous en concluons que l'idée d'utiliser une distance est intéressante, mais que celle-ci devra être optimisée si l'on veut atteindre de meilleures performances.

1.3 Données utilisables pour la recherche de similitude

Les méthodes de recherche de similarité décrites ci-dessus sont fondées sur l'utilisation de données précises. Dans les OTFM1g, ce sont presque exclusivement le flot des caractères du segment primaire. Dans les deux autres méthodes ce sont une classification sémantique des mots de la phrase, et/ou un glossaire bilingue de ces mots.

Reprenant le cadre qui a été donné dans les chapitres précédents, c'est-à-dire l'expression d'un segment de traduction en deux parties (un segment XML et une structure TELAM), nous examinons maintenant les données que nous pouvons utiliser pour effectuer cette recherche systématique de similitude.

Ils s'agit des données représentées sur une structure TELAM, y compris la forme primaire qui est considérée comme l'étage 0 d'une telle structure.

- *les caractères de base de la forme primaire* : ils sont donnés à l'étage 0, et se présentent comme une suite hétérogène.
- *les caractères du texte* : ces caractères d'une part ne comprennent plus de données hors texte qui introduisent du "bruit", et sont d'autre part exprimés en un codage et une transcription adaptables au traitement que l'on veut appliquer (étage 1).
- *suite de mots* : proviennent de la séparation du flot de caractères du texte en mots.
- *suite de lemmes* : ils proviennent de la lemmatisation de la suite de mots, et sont représentés à l'étage 5. Plusieurs lemmes peuvent être associés à un mot donné.
- *suite de catégories grammaticales* : elle provient d'une analyse morpho-syntaxique et associe à chaque mot les catégories grammaticales possibles. Ces catégories sont soit indiquées au niveau des lemmes (étage 5), soit à un étage dédié.
- *suite d'autres catégories* : on pourrait imaginer une autre sorte de "balisage" ("tagging" en anglais), qui associe aux mots d'autres classes ou catégories.
- *suite des termes provenant d'une base terminologique* : ces termes sont gérés au niveau d'une base terminologique. A chaque mot est associé un (ou plusieurs) termes, cela donne une suite de termes. (étage7).
- *catégorisation sémantique des mots (ou lemmes)* : cela consiste en une suite de traits sémantiques par mot (ou lemme), et est codé à l'étage des lemmes ou sur un étage séparé.
- *suite des balises doubles* : extraites du segment XML, elles sont représentées à l'étage 3. Elles peuvent avoir une importance dans la recherche de similarité si l'on admet que par exemple la mise en forme, dont les zones d'applications sont déterminées par des balises, est importante.
- *suite des monobalises* : les monobalises représentent des objets non textuels. Si les segments que l'on traite sont issus d'un document hypertexte contenant donc des hyperliens qui seront retranscrits par des monobalises, et si le travail de traduction est en fait une mise à jour de cet hypertexte, il est alors crucial de pouvoir tester la présence de monobalises. Elles sont représentées à l'étage 4.

Il reste donc à savoir comment utiliser ces données de façon systématique pour une recherche effective de similarité. Nous allons maintenant donner un cadre formel de définition de la similitude entre deux segments.

2. Similitude basée sur une distance

Introduction

Nous avons vu que la notion de similitude dépend beaucoup de sa formalisation. Suivant l'exemple de [Sumita et Iida 1992], et d'autres aussi, nous baserons le calcul de la similitude entre deux segments sur une distance.

Nous donnons d'abord une interprétation mathématique des textes (et donc des segments) de façon à construire clairement les ensembles qui, munis d'une distance, donnent les espaces métriques que nous utiliserons pour ces calculs de distance entre segments. Puis la notion de distance est rappelée et une discussion sur les éléments de base des espaces métriques, et sur les distances applicables est menée.

2.1 Définitions

Nous empruntons la notion de texte à [Schreider 1975] qui fonde lui même ses travaux sur ceux de [Arapov & Borshchev 1970]. La définition d'un texte se base sur celle de schéma syntaxique que nous présentons ici.

2.1.1 Texte unidimensionnel

Définition 1 : Schéma syntaxique

Un schéma syntaxique est la donnée d'un ensemble M muni des relations A_1,..., A_n, et est noté : S = <M; A_1,..., A_n>. M est appelée "l'ensemble support"[2].

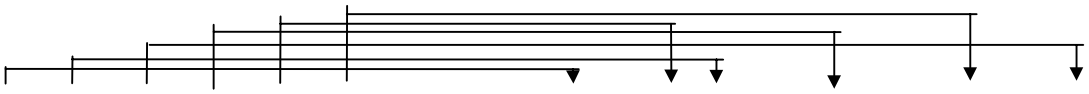
Dans les exemples qui suivent M sera un sous-ensemble de N, et gère les "places" des unités de texte.

Définition 2 : Texte (unidimensionnel[3])

Un texte T est défini par deux éléments : un schéma syntaxique S, et une fonction[4] $\varphi : M \rightarrow \sum$, où $\sum$ est un ensemble d'éléments appelé "alphabet". Il est noté T = <S, φ>.

Exemple :

Soit $\sum$ l'ensemble des formes des mots du français. Soit M l'ensemble N des entiers naturels, muni d'une seule relation A_1, celle d'ordre total, notée "<". Dans cette formalisation, un texte est la donnée d'un sous-ensemble de N ("d'indices") ordonnés, et d'une application qui à chaque indice i associe une forme de mots de $\sum$.



{1, 2, ,3 ,4 ,5 ,6 } {ceci,..., de, est,..., exemple,..., texte,...,..un,....}

Figure 2 : modélisation d'un texte selon [Schreider 1975]

Dans la figure ci-dessus, l'application φ est représentée par les flèches. A l'ensemble des entiers de 1 à 6 sont associés des formes de mots français. L'ordre total dont N est muni permet de retranscrire la "succession" des mots dans la phrase.

[2] "Carrier set" en anglais.

[3] Nous ajoutons ce qualificatif à la terminologie de [Schreider 1975]

[4] une application univoque est aussi une fonction [Berge 1959, p21]

Un segment est alors un texte particulier dont la "longueur" est donnée par le dernier indice de M. Si $\sum$ est l'ensemble des caractères ASCII, alors il peut s'agir de la longueur du segment XML, si $\sum$ est l'ensemble des mots de la phrase, il s'agira de la longueur de la phrase en termes de mots, etc.

Remarque :

[Schreider 1975] définit un ensemble de cinq relations A_i élémentaires : la relation de succession, de contrôle grammatical, de concordance, d'homogénéité, et d'occurrence dans un constituant. Selon son hypothèse, elles permettent, par composition, de caractériser l'ensemble des formalisations linguistiques. Nous ne retiendrons de ces relations élémentaires que celle de succession (ordre total sur N), comme on l'a déjà vu dans les exemples ci-dessus. Nous n'utilisons pas en effet de notion de théorie linguistique forte d'une part, et laissons à cet auteur la responsabilité de ses affirmations sur la puissance de ses relations élémentaires.

2.1.2 Distance

Nous considérons que la similitude entre deux segments S1 et S2 s'estime par l'utilisation d'une **distance** d(S1, S2). Si d(S1,S2) est "faible", S1 et S2 étant deux segments sont alors considérées comme "proches". Plus exactement, un seuil η étant fixé, on décide que S1 et S2 sont "assez proches" si $d(S1, S2) < \eta$.

Définition 3 : Distance

Une application d'un ensemble E dans R+ est une distance si et seulement si, pour tous S1, S2, et S3 de E :

$d(S1, S2) \geq 0$
$d(S1, S2) = 0 \Leftrightarrow S1 = S2$
$d(S1, S2) = d(S2, S1)$
$d(S1, S3) \leq d(S1, S2) + d(S2, S3)$

Nous venons donc définir:

- l'espace M des segments S
- la distance d sur les éléments de l'espace M

2.1.3 Distance élémentaire et distance composée

Soit $\sum$ un alphabet. Soit δ une distance sur $\sum$. On dit que $[\sum,\delta]$ forme un "espace métrique". Soient maintenant S1 et S2 deux segments vus comme des suites finies d'éléments de $\sum$: $S1 = \sigma_{11}....\sigma_{1m}$, $S2 = \sigma_{21}.... \sigma_{2n}$. En général, on cherche à définir une distance sur les chaînes de caractères ($\sum$*) de façon inductive à partir des $\delta(\sigma_{1i}, \sigma_{2j})$.

2.1.4 Distance sur un texte

Soit donc un texte T = <S, φ>, où φ : M → V. La définition d'une distance d se fera comme expliqué ci-dessus sur l'ensemble du vocabulaire V (=$\sum$*) de façon à pouvoir estimer la distance d'un élément de V à un autre.

2.2 Choix des espaces et des distances

2.2.1 Espaces

Pour choisir des espaces convenables et adaptés à la recherche de similitude entre segments, nous reprenons les types de données listées plus haut, et montrons quelques espaces métriques candidats basés sur ces données.

V l'ensemble des caractères $\sum$

Les segments sont traités comme des suites de caractères, sur l'espace métrique [$\sum$, δ].

V est l'ensemble des mots $\sum$*

Soit V $\subseteq \sum$*. les segments alors sont vus comme des suites finies d'éléments de V, soit encore des suites finies de *mots*, éléments de V = $\sum$*. Cela forme l'espace métrique [$\sum$*, d].

Soient S1 et S2 deux segments composés de mots. La plus petite unité est le mot. La comparaison de S1 et S2 se fonde alors sur la distance entre mots. Il est possible de continuer à voir un mot comme une succession de lettres et de baser la distance sur la distance entre les lettres des mots. On peut tout aussi bien considérer d'autres types de distances entre mots. Ces distances peuvent être définies à partir d'un classement des mots comme on le verra à l'aide des lemmes associés, de catégorie grammaticale, sémantique, terminologique, ou d'autres comme cela sera expliqué aux points suivants.

Comparaison de classes de mots

Soit deux segments S1 et S2 de mots. Supposons que nous possédons un analyseur donnant pour chaque mot une classe c, parmi un ensemble fini C de classes. Alors il est possible de voir le segment comme une suite finie de classes S1 = c_{11} c_{12}.....c_{1n}, et
S2 = c_{21} c_{22}.....c_{2m}. Il est alors possible d'appliquer le même principe de comparaison par une distance d sur l'espace métrique M = [C*, d], pourvu que la distance $\delta(c_{1i}, c_{2j})$ entre deux instances de classes soit définie. C'est ainsi que la distance appliquée précédemment sur les segments vus comme des suites de mots indivisibles est aussi applicable dans ce cadre. Voici une liste non exhaustive de classes de mots qui peuvent servir à calculer la distance entre les deux segments :

- catégories grammaticales
- domaines
- classes de cooccurrences
- classes sémantiques

Le raisonnement mathématique est le même pour toutes ces classes. Nous allons donner une idée plus précise de ce que cela peut donner sur les catégories grammaticales des mots.

Comparaison de vecteurs de classes grammaticales

Deux segments peuvent sembler similaires, bien que les mots qui les composent ne soient pas identiques, comme dans les exemples suivants [5].

[5] Ces exemples ont été trouvés "à la volée" dans l'aide en ligne de Microsoft Word qui a servi à écrire cette étude.

S1	S2
Add borders to paragraphs	Add shading to paragraphs
Click Database, on the Insert Menu	Click Get Data, under Data Source
Click Table AutoFormat	Click Insert Data

Table 1 : Exemple de segments à structure syntaxique similaire

Cette ressemblance provient clairement de la similitude des constructions syntaxiques de ces segments. En traduction automatique, cela est généralement géré par des règles d'analyse et de génération inscrites au sein des moteurs de traduction. Nous ne possédons pas ces puissants moteurs, et notre but n'est pas non plus de leur substituer une procédure. Pourtant dans une première approche, nous soutenons que la succession des catégories grammaticales des mots formant les segments est un indicateur de la similitude de ces segments souvent efficace. Reprenant les segments de l'exemple ci-dessus, le tableau suivant montre les catégories grammaticales des mots[6] qui les forment :

S1	S2
V N P N	V N P N
V O, P Art, O	V O, P O
V O	V O

Table 2 : Vecteurs de catégories grammaticales correspondant aux segments précédents

La notion "d'objet" suit ce que nous avons déjà vu au chapitre 4 : certains termes spéciaux de terminologie du domaine sont employés en fait comme des noms dans les manuels techniques. Dans les fichiers de ressources de programmes, ces objets peuvent être des variables comme "%1". Nous les dénoterons O dans la suite. Lors de l'analyse du segment qui donne la structure TELAM, une recherche sur les mots de l'étage 1 est conduite ; si de tels objets sont trouvés, ils sont en général représentés à l'étage 3 de la structure TELAM.

Nous allons maintenant préciser et formaliser cette notion de ressemblance pour l'utiliser dans la recherche d'un segment de mémoire similaire au segment à traduire.

Définition des vecteurs de catégories grammaticales

L'analyse du segment en une structure TELAM prévoit l'indication des attributs linguistiques, et parmi eux la catégorie grammaticale. L'ordre de succession des catégories grammaticales est un indicateur de la structure grammaticale du segment.

Définition 4 : Vecteur de catégories grammaticales

Soit un segment S de texte constitué de p mots. Soit L la liste ordonnée des p lemmes correspondants. Le "vecteur de catégories grammaticales" V^p associé à S et L est le vecteur de dimension p des catégories grammaticales ordonnées des éléments de L.

Exemple :

S = "Press the red buttons"

L = (press, the, red, button)

V^p = (verbe, article, adjectif, nom)

[6] Nous utilisons les abréviations suivantes : V (verbe), N (nom), Adj (adjectif), Adv (adverbe), P (préposition), Art (article), LINK (conjonctions, locutions conjonctives), O (objet).

Remarque :

- Nous étendons la notion de catégorie grammaticale aux ponctuations, à la notion d'objet vue en introduction, et à la notion de pivot (cf. Chapitre 5).
- Pour un exemple de telle entité, on peut se reporter à la Figure 15 du Chapitre 2 (paragraphe I.1.b.). Le mot "ENTER" est en fait un objet de ce type car il fait partie de la liste des commandes du logiciel. Le vecteur associé au segment "Click a color and press ENTER" est : (V, Art, N, LINK, V, O). Ce même vecteur qualifie "Click a color and press %1".

Comparaison sémantique de deux mots

Il est possible d'attacher une suite ordonnée de traits sémantiques à chaque mot m_{ki}. Soit alors une telle qualification sémantique sem() telle que : sem(m_{ki}.) = t^k_{i1}... t^k_{i1}. Alors en supposant que l'on a une distance entre les t^k_{il} [7], il est possible de mener les mêmes raisonnements que précédemment, déduire une distance sémantique pour les mots, et pour les segments. On pourra se reporter ci-dessous pour un exemple de distance sémantique.

2.2.2 Difficulté de définition des distances

Pour tester la proximité de deux segments représentés par une suite de caractères d'un vocabulaire V, il est classique de fonder une distance d sur une fonction des distances entre caractères (alors [V*, d]).

Exemple :

Si V = (alphabet français) = {a, b, c, ..., z} ∪ {_}[8], alors "bleuet", "lys", et "coquelicot" sont des éléments de V*. Soient m1=m_{11}.....m_{1n} et m2=m_{21}....m_{2m}, deux mots de ce type.

Soit la fonction δ définie entre deux "lettres" m_{1i} et m_{2j} telles que :

δ(m_{1i}, m_{2j})= 0 si m_{1i},= m_{2j}, 1 si m_{1i},≠ m_{2j}.

On a bien δ(a, a)= 0, δ(a, b) = δ(b, a) = 1, et δ(a, c) ≤ δ(a, b) + δ(b, c)[9]. Il est alors possible de définir une fonction de coût positive entre un mot m1 et m2 de la manière suivante :

d'(m1,m2) = $\frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{m}\delta(m_{1i}, m_{2j})$, ainsi :

d'(bleuet, lys) = 5/6 = 0.83
d'(bleuet, les) = (1 + 0 + 0 + 1 + 0 + 1) / 6 = 3/6 = 0.5
d'(les, bleuet) = 1/3 = 0.33
d'(bleuet, bleu) = (0+0+0+0+0+1) /6 = 1/6 = 0.17
d'(bleue, bleu) = (0 + 0 + 0 + 0 + 0) / 5 = 0

[7] On peut penser à la profondeur du plus petit attribut commun dans le treillis des attributs sémantiques, comme dans thésaurus de [Ohno et Hamamashi, 1984], cité dans [Sumita et Iida 1992].

[8] Il s'agit de l'espace, séparateur de mots en français.

[9] Trivial si a = b = c, si a = b ≠ c : δ(a, c) = 1 ≤ δ(a, b) + δ(b, c) = 0 + 1, si a ≠ b = c idem, si a ≠ b ≠ c, δ(a, c) = 1 ≤ δ(a, b) + δ(b, c) = 1 + 1 = 2, donc l'inégalité triangulaire est bien vérifiée.

Cette distance correspond en fait à la proportion de lettres du premier mot absentes du second. Cette fonction ne vérifie pas l'axiome de symétrie, car l'on voit par exemple que d'(bleuet, les) ≠ d'(les, bleuet). Symétrisons d en calculant la moyenne des distances calculées "dans les deux sens" :

d(m1, m2) = ½[d'(m1,m2) + d'(m2,m1)], cela donne pour les calculs précédents :

d(bleuet, lys) = d(lys, bleuet) = ½(5/6 + 2/3) = 0.75
d(bleuet, les") = d(les, bleuet) = ½(3/6 + 1/3) = 0,43
d(bleuet, bleu) = d(bleu, bleuet) = ½(1/6 + 0) = 0,08
d(bleue, bleu)= d(bleu, bleue) = ½(0 + 0) = 0

L'axiome de symétrie est maintenant vérifié. D'autre part, on a bien d(m1, m1) = 0. Mais il existe des mots différents m1 et m2 tels que d(m1, m2) = 0, comme par exemple pour d'(bleue, bleu). Le deuxième axiome n'est pas vérifié. Pour obtenir ceci, il faudrait introduire un comptage de mots, ou un repérage de place de mot...

Quant à l'inégalité triangulaire, cette fonction de coût ne la vérifie pas non plus. Il suffit pour s'en convaincre de prendre les trois mots (théoriques) suivants :

S1 = abcd, S2 = cd, S3 = cdefghij
d(S1, S2) = ½ (2/4 + 0/2) = 1/4
d(S2, S3) = ½ (0/2 + 2/4) = 1/4
d(S1, S3) = ½ (4/4 + 6/8) = ½ (14/8) = 14/16 = 0.875

et d(S1, S2) + d(S2, S3) = 1/2 < 0.875 = d(S1, S2)

Nous en concluons que la recherche d'une fonction qui possède toutes les propriétés d'une distance n'est donc pas triviale. Nous avions en effet avec d une fonction de coût qui avait l'air adaptée à la mesure de similarité, mais qui ne vérifie pas pour autant les axiomes d'une distance.

2.3 Distance généralisée

2.3.1 Définition mathématique d'un texte multidimensionnel

Nous inspirant de la formalisation mathématique qui a défini plus haut un texte unidimensionnel, nous donnons maintenant celle de texte multidimensionnel.

Définition 5 : Texte multidimensionnel

Soit S = <M; A_1,..., A_n> un schéma syntaxique. Soit $\{\Sigma_1,...,\Sigma_p\}$ un ensemble de p alphabets. Soient les fonctions $(\varphi_j : M \rightarrow \Sigma_j)_{1\leq j \leq p}$ qui aux éléments de M associe des éléments de Σ_j. Un texte multidimensionnel est la donnée de $T = <S, (\varphi_j)_{1\leq j \leq p}>$.

Segment de texte

Les segments de texte vus dans les chapitres précédents peuvent alors se concevoir comme un texte multidimensionnel. Soit M l'ensemble des entiers naturels. Soit A_1, la relation d'ordre total sur N. Soient les alphabets suivants :

Σ_0: {ensemble des 128 caractères ASCII 7 bits}
Σ_1: {ensemble des 256 caractères ASCII 8 bits}
Σ_2: {ensemble des formes de mots français : "a", "abaca", "abacas", "abacule", ...}
Σ_3: {ensemble des balises doubles XML : <hi1>, </hi1>, ...}
Σ_4: {ensemble des monobalises XML : <monobj/>, ..}
Σ_5: {ensemble des lemmes du français : "a", "abaca", "abacule", ...}
Σ_6: {ensemble des catégories grammaticales augmentées : Art, Adj, Adv, N, Prep,...}

Soient les fonctions $(\varphi_j)_{1\leq j \leq 6}$ associées telles que $\varphi_j : M \rightarrow \Sigma_j$. Alors $T = <S, (\varphi_j)_{1\leq j \leq p}>$ est un "segment" tel qu'on le trouve au niveau des segments XML vus aux chapitres précédents.

Dans ce cas on voit que φ_0 représente le codage XML d'un texte, et par exemple $\varphi^{-1}_0 \bullet \varphi_1$ correspond à la fonction qui fait passer de l'étage 0 à l'étage 1 d'une structure TELAM représentant un segment qui est lui-même le résultat de l'application de l'application successives des fonctions φ_1, φ_3, et φ_4, de M dans $\Sigma_1 \cup \Sigma_3 \cup \Sigma_4$. L'ordre total de M est alors isomorphe à l'échelle linéaire définie sur l'ensemble des caractères de l'étage 0.

2.3.2 Espace métrique multidimensionnel et distance généralisée

Soit un texte multidimensionnel $T = <S, (\varphi_j)_{1\leq j \leq p}>$, où il existe j applications φ_j telles que $\varphi_j : M \rightarrow \Sigma_j$. Nous avons vu au paragraphe précédent comment associer une distance d_j à l'un des alphabets Σ_j. Supposons donc que nous disposons des p espaces métriques (Σ_j, d_j).

Considérons maintenant l'espace Σ comme le produit cartésien des Σ_j.:

$$\Sigma = \Sigma_1 \text{ x } \Sigma_2 \text{ x....x } \Sigma_p = \prod_{j=1}^{p} \Sigma_j$$

Alors le produit $\prod_{j=1}^{p} (\Sigma_j, d_j)$ admet comme distance naturelle[10], en particulier :

[10] Voir par exemple, [Berge 1959], p 90-91.

d((x_{11}, ...x_{1p}), (x_{21},...,x_{2p})) = $\sqrt[n]{\sum_{j=1}^{p} d_j{}^n(x_{1j}, x_{2j})}$, où n est une puissance quelconque.

En prenant n = 2, et (α_1,..., α_p) une suite de cœfficients positifs, on obtient une nouvelle distance produit[11] :

$$d((x_{11}, ...x_{1p}), (x_{21},...,x_{2p})) = \sqrt[n]{\sum_{j=1}^{p} \alpha_j d_j{}^n(x_{1j}, x_{2j})}$$

2.3.3 Interprétation

Reprenons les hypothèses précédentes, et considérons que le multitexte T= <S, $(\varphi_j)_{1\le j\le p}$> représente l'ensemble des étages de la structure TELAM d'un segment S1. Plus exactement, T est le produit cartésien des textes correspondants à la concaténation d'un champ de la décoration des nœuds du "dernier chemin" (voir chapitres 5, 6, et 7) de chaque étage.

Exemple :

φ_{11}(M) = {concaténation des caractères de chaque nœud du dernier chemin de l'étage 1}

...

φ_{15}(M) = {concaténation des lemmes de chaque nœud du dernier chemin de l'étage 5}

...

Alors on considère le segment S1 comme représenté par (φ_{11}(M),..., φ_{1p}(M)), qui est à valeurs dans $\prod_{j=1}^{p} \Sigma_j$. Prenons la même représentation pour S2 : (φ_{21}(M),..., φ_{2p}(M)), alors on considérera que la distance entre S1 et S2 est donnée par :

$$d(S1, S2) = \sqrt{\sum_{j=1}^{p} \alpha_j d_j{}^2(\varphi_{1j}(M), \varphi_{2j}(M))}$$

De façon graphique, l'on peut comprendre cela comme l'évaluation de la distance entre deux colonnes représentant chacune S1 et S2, dans un plan vertical :

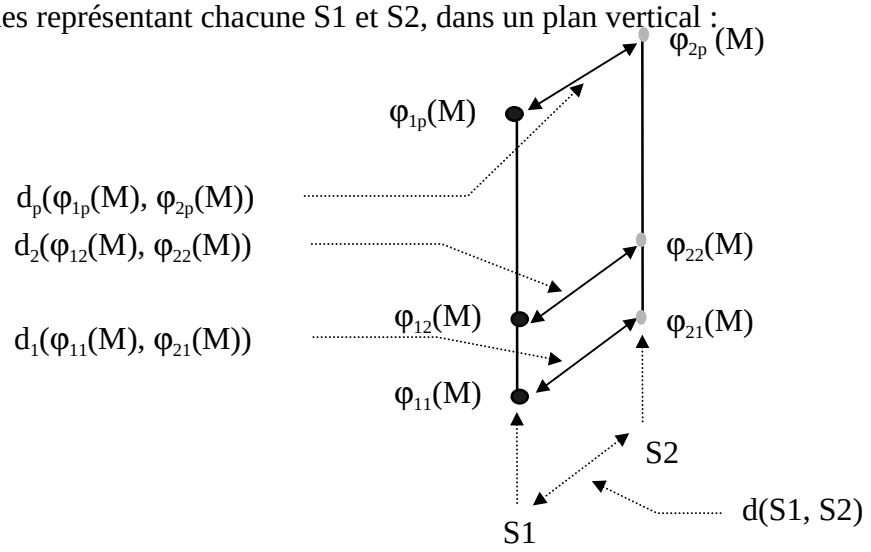


Figure 3 : distance produit d(S1, S2) entre deux segments S1 et S2

[11] En posant d'(x_{1i}, x_{2j}) = $\sqrt[p]{\alpha_j}$ d(x_{1j}, x_{2j}), la preuve est immédiate.

La colonne est constituée de coordonnées verticales qui correspondent à chacun des $\varphi_{kl}(M)$. A chaque "étage", la similitude est calculée par une distance d_j adaptée à l'étage (étage des lemmes si l=5 par exemple). La similitude entre les segments S1 et S2 est vue comme la distance produit entre vecteurs coordonnées de S1 et S2. La réduction de la similitude est alors équivalente à la diminution de la surface du rectangle dessiné en perspective. En toute rigueur, on devrait voir un polyèdre, car les distances à chaque étage ne sont pas forcément égales.

Conclusion

Dans ce paragraphe 1, nous avons d'une part formalisé la notion de texte unidimensionnel et multidimensionnel, et montré le lien avec les structures TELAM. D'autre part nous avons opté pour la représentation de la similitude entre deux segments par une distance appliquée à l'ensemble des dimensions du segment, via une métrique produit.

3. Similitude basée sur une distance d'édition

Introduction

Nous rappelons ici la définition d'une distance d'édition et les algorithmes associés, présentés par [Wagner & Fisher 1974], et utilisés dans [Lepage 1996]. Il s'agit de définir la distance entre deux chaînes S1 et S2, comme coût des éditions élémentaires que sont la suppression, l'insertion ou la substitution de symboles, nécessaires au passage de S1 à S2.

3.1 Distance d'édition

3.1.1 Opérations d'édition

Nous rappelons d'abord ce qu'est une opération d'édition élémentaire.

Définition : Opération d'édition élémentaire

Soit V un vocabulaire donné (un ensemble symboles). Soit $V^* = V^+ \cup \{\varepsilon\}$ l'ensemble des chaînes finies de caractères, où $\Lambda=\{\varepsilon\}$ représente la chaîne vide.

Une opération d'édition élémentaire est un couple $(a, b) \neq (\Lambda,\Lambda)$ de $V^+ \times V^+$, où a et b sont deux chaînes élémentaires de longueur 0 (Λ est la chaîne nulle), ou 1 (composée d'un seul caractère). Elle est notée $a \rightarrow b$.

Alors la chaîne B est le résultat de l'application de l'opération d'édition sur la chaîne A, s'il existe des chaînes σ et τ de V* telles que $A = \sigma a \tau$, et $B = \sigma b \tau$. On note ceci $A \Rightarrow B$.

On considérera ici trois opérations élémentaires :

Définition : Opérations d'éditions élémentaires classiques :

$a \rightarrow b$ est une opération de **substitution** si $(a, b) \neq (\Lambda,\Lambda)$
$a \rightarrow b$ est une opération d'**insertion** si $a = \Lambda$ et $b \neq \Lambda$
$a \rightarrow b$ est une opération de **suppression** si $a \neq \Lambda$ et $b = \Lambda$

La substitution peut être stricte avec $a \neq b$, ou identique si $a = b$.

Exemple :

Soit V = {a, b, ..., z} = alphabet français.
On passe de A = " sens " à B = " sans ", par une substitution $e \rightarrow a$.
On passe de A = " sens " à B = " seins ", par une insertion $\Lambda \rightarrow i$.
On passe de A = " sens " à B = " ses ", par une suppression $n \rightarrow \Lambda$.

Définition : S-Dérivation

Soit une séquence d'édition S, exprimée comme une suite d'opérations élémentaires d'édition $S = s_1, s_2, ..., s_m$. Une S-dérivation de la chaîne A vers la chaîne B, est une suite de chaînes $A_0, A_1,..., A_m$ telles que $A = A_0$, et $B = A_m$, et telles que $A_{i-1} \Rightarrow A_i$ via s_i, $\forall\ 1 \leq i \leq m$.

Exemple :

[Schreider 1975] rappelle une vieille plaisanterie estudiantine selon laquelle " white " est égal à " black ", et qui repose sur une S-Dérivation dont les opérations élémentaires ne sont que des substitutions :

S_1 = white, while, whale, shale, shave, stave, stove, store, stork, stock, stack, slack, black.

Le passage de "white" à "while" par exemple, se fait par une substitution t→l.

3.1.2 Distance d'édition

Définition

Un coût est assigné à chacune de ces opérations. La distance d'édition entre deux chaînes S1 et S2 est alors vue comme la somme minimale des coûts des opérations élémentaires d'une S-dérivation nécessaire au passage de S1 à S2.

Définition : Distance d'édition sur les chaînes

Soit une fonction de coût positive γ associée à chacune de ces éditions élémentaires s_i et vérifiant : $\gamma(a \rightarrow a) = 0$, $\gamma(a \rightarrow b) = \gamma(b \rightarrow a)$, $\gamma(a \rightarrow b) + \gamma(b \rightarrow c) \geq \gamma(a \rightarrow c)$.

Soit l'extension de cette fonction de coût à une dérivation S telle que $\gamma(S) = \sum_{i=1}^{m} \gamma(s_i)$ [12].

Alors on définit la distance entre deux chaînes comme :

$d(A, B) = \min \{\gamma(S), S$ étant une S-dérivation de A vers $B\}$.

Exemple :

La distance de Levenshtein (rappelée dans [Stephen 1994]) propose de considérer les trois opérations de base que sont la substitution, la suppression et l'insertion, et de prendre comme valeurs pour γ :

$\gamma(a \rightarrow b) = 0$ si a = b (identité), 1 si $a \neq b$ (substitution stricte)
$\gamma(a \rightarrow \varepsilon) = 1$ (suppression)
$\gamma(\varepsilon \rightarrow a) = 1$ (insertion)

Sous ces hypothèses, dans l'exemple du passage de "black" à "white" ci-dessus, $\gamma(S_1) = 13$, car il y a treize substitutions. Cependant les mots de départ et d'arrivée comportant cinq lettres différentes chacun, le nombre minimal d'opérations pour passer de A à B est cinq opérations élémentaires pour passer de l'un à l'autre à l'aide cinq substitutions. La distance de A = " white " à B = " black " est donc d(A, B) = 5.

[12] Si m = 0, $\gamma(S) = 0$.

Calcul

Examinons maintenant ce qui se passe lors d'une opération élémentaire d'une chaîne $S_1=\sigma_{11}... \sigma_{1m}$ à une chaîne $S_2=\sigma_{21}... \sigma_{2n}$. Soit $S_1(1,i) = \sigma_{11}... \sigma_{1i}$, $S_2(1,j) = \sigma_{21}... \sigma_{2j}$, avec $i \leq m$ et $j \leq n$, et $d_{ij} = d(S_1(1,i), S_2(1,j))$.

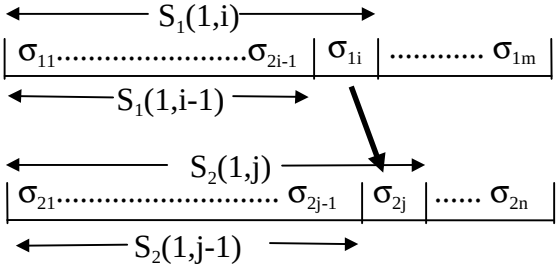


Figure 4 : Passage de $S_1(1,i)$ à $S_1(2,j)$

Nous considérons que l'édition de S1 à S2 est une opération incrémentale, que nous avons déjà transformé $S_1(1,i-1)$ et $S_2(1,j-1)$, et que la distance entre ces deux dernières chaînes est d'_{i-1j-1}. Alors la transformation de $S_1(1,i)$ en $S_2(1,j)$ peut résulter de l'une des trois opérations élémentaires insertion, suppression, ou substitution. Cela donne trois évolutions pour la distance d_{ij}, que nous présentons dans le tableau suivant :

	σ_{1i}	σ_{2j}	d'_{ij}
suppression	ε	σ_{2j}	$d'_{ij} = d_{i-1j} + \gamma(\sigma_{1i} \rightarrow \varepsilon)$
insertion	σ_{1i}	ε	$d'_{ij} = d_{ij-1} + \gamma(\varepsilon \rightarrow \sigma_{2j})$
substitution	σ_{1i}	σ_{2j}	$d'_{ij} = d_{i-1j-1} + \gamma(\sigma_{1i} \rightarrow \sigma_{2j})$

Figure 5 : Evolution des distances de dij

Comme la distance est la somme des opérations élémentaires entre les chaînes, si l'on passe de $S_1(1,i)$ à $S_2(1,j)$ par la suppression de σ_{1i}, la distance d'_{ij} entre ces deux chaînes peut être vue comme la distance entre $S_1(1,i-1)$ à $S_2(1,j)$, soit d_{i-1j} , plus le coût de la suppression, soit $\gamma(\sigma_{1i} \rightarrow \varepsilon)$. Le raisonnement est similaire pour les deux autres situations.

On a ainsi :

$$dij = \min (d_{i-1j} + \gamma(\sigma_{1i} \rightarrow \varepsilon), d_{ij-1} + \gamma(\varepsilon \rightarrow \sigma_{2j}), d_{i-1j-1} + \gamma(\sigma_{1i} \rightarrow \sigma_{2j}))$$

Cela donne une relation de récurrence permettant de calculer d_{ij} de proche en proche, pourvu que l'on fournisse les conditions initiales que voici :

$$d_{00} = 0,$$

$$d_{i0} = \sum_{k=1}^{i} \gamma(\sigma_{1i} \rightarrow \varepsilon), \ 1 \leq i \leq m$$

$$d_{0j} = \sum_{k=1}^{j} \gamma(\varepsilon \rightarrow \sigma_{2j}), \ 1 \leq i \leq m$$

Dans le cas de distance de Levenstein, où $\gamma(\sigma_{1i} \rightarrow \varepsilon) = \gamma(\varepsilon \rightarrow \sigma_{2j}) = \gamma(\sigma_{1i} \rightarrow \sigma_{2j}) = 1$, cela donne :

$$d_{00} = 0,$$

$d_{i0} = i$,
$d_{j0} = j$

Le calcul par récurrence peut être représenté par un tableau du type suivant, qui aux indices i et j fait correspondre les distances incrémentales d_{ij}, depuis la distance entre deux chaînes vides à celle entre "analogie" et "paradoxe".

	j	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
			p	**a**	**r**	**a**	**d**	**o**	**x**	**e**
i										
0		0	1	2	3	4	5	6	7	8
1	**a**	1	1	1	2	3	4	5	6	7
2	**n**	2	2	2	2	3	4	5	6	7
3	**a**	3	2	2	3	2	3	**4**	5	6
4	**l**	4	4	3	3	3	3	4	5	6
5	**o**	5	5	4	4	4	4	3	4	5
6	**g**	6	6	5	5	5	5	4	4	5
7	**i**	7	7	6	6	6	6	5	5	5
8	**e**	8	8	7	7	7	7	6	6	5

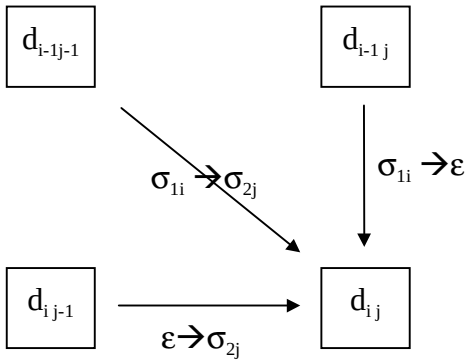


Figure 6: Calcul récurrent de dij

Le tableau ci-dessus est tiré de [Stephen 1994]. Nous avons ajouté le schéma de droite permettant d'interpréter le passage d'une case à l'autre. Ce tableau se lit par groupe de quatre cellules. Pour l'exemple des quatre cellules entourées ci-dessus, le "4" inférieur droit (en gras) correspond à d_{36}, qui donne la distance entre "ana" et "parado" Cette distance est calculée en évaluant le minimum des sommes suivantes :

d'_{36} = d(ana, parado) = d(ana, parad) + $\gamma(\varepsilon \rightarrow$'o') = d_{35} + 1 = 3 + 1 = 4 ($\rightarrow$)
d'_{36} = d(ana, parado) = d(an, parado) + γ('a' $\rightarrow\varepsilon$) = d_{26} + 1 = 5 + 1 = 6 ($\downarrow$)
d'_{36} = d(ana, parado) = d(an, parad) + γ('a'$\rightarrow$'o') = d_{25} + 0 = 4 + 0 = 5 ($\searrow$)

Le minimum est donc atteint dans ce cas pour l'insertion, et donc d_{36} = 4, nombre qui reste inscrit dans le tableau à la position (3, 6). La distance finale entre "analogie" et "paradoxe", soit d_{88}, peut être lue dans la case inférieure droite, et vaut 5.

Cette méthode permet de déterminer une dérivation S_0 rendant minimale la somme $\gamma(S)$, ce qui permet de connaître la distance d(S_1, S_2). La chaîne des opérations élémentaires constituant la dérivation S_0 est choisie à chaque itération. L'opération la moins coûteuse est choisie. Dans le cas où deux au moins des opérations possibles (insertion, suppression, substitution) ont le même coût, l'opération élémentaire choisie est arbitraire (toujours la substitution par exemple, et il y a plusieurs solutions pour S_0.

Il serait possible de garder la mémoire de toutes les possibilités, mais cela entraînerait une augmentation non négligeable de la complexité de l'algorithme de Wagner et Fisher, que nous souhaitons garder efficace (voir plus loin la complexité).

Pour caractériser cette dérivation, [Wagner & Fisher 1974] introduisent la notion de "trace", dont voici la définition :

Définition : Trace de A vers B

Soient deux chaînes S_1 et S_2. Une trace de S_1 vers S_2 est un triplet (T, S_1, S_2) où
* T est un ensemble ordonné de paires (i, j)
* $1 \leq i \leq |S_1|$ et $1 \leq i \leq |S_2|$, où $|S_1|$ est la longueur de S_1.
* quelles que soient deux paires distinctes $(i_1, j_1) \neq (i_2, j_2)$ de T :
  a) $i_1, \neq i_2, j_1 \neq j_2$
  b) $i_1, < i_2$ si et seulement si $j_1 < j_2$

Graphiquement, les paires (i, j) représentent des lignes joignant un symbole de chaque chaîne A et B comme suit :

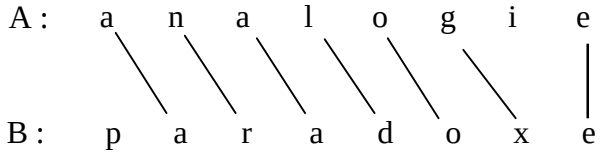


Figure 7 : Une trace de "analogie" vers "paradoxe".

La condition 1 restreint dûment le domaine d'action des lignes aux seuls caractères des chaînes. La condition (a) assure que les symboles joints sont différents. La condition (b) assure le non croisement des lignes.

Pour la définition suivante, nous convenons que si S_1 et S_2 sont des éléments de V^+, σ_{1i} le i-ième caractère de la chaîne S_1, et S(i,j) l'ensemble des caractères contenus entre σ_{1i} et σ_{1j} compris[13].

Définition : Coût d'une trace

Soit T une trace, soient I et J les ensembles d'indices de S_1 et S_2 qui ne participent pas aux couples de T[14]. Alors le coût de la trace est défini comme suit :

$$\text{coût(T)} = \sum_{(i,j)\in T} \gamma(\sigma_{1i} \rightarrow \sigma_{2j}) + \sum_{i\in I} \gamma(\sigma_{1i} \rightarrow \Lambda) + \sum_{j\in J} \gamma(\Lambda \rightarrow \sigma_{2j})$$

Wagner et Fisher prouvent qu'alors :

Théorème:

d(A, B) = min {coût(T) | T est une trace}

Cela correspond alors à la méthode décrite ci-dessus.

3.1.3 Algorithmes

[Wagner & Fisher 1975] donnent deux algorithmes : l'algorithme X pour calculer la distance entre deux chaînes A et B, et l'algorithme Y qui donne la trace d'une des dérivations S_0 minimisant la fonction $\gamma(S)$.

[13] Si i > j, A(i,j) = ε.
[14] C'est à dire graphiquement qui ne sont pas touchés par une ligne.

Voici maintenant l'algorithme X. Il est divisé en deux parties : la fixation des conditions initiales, correspondant à la première colonne et à la première ligne du tableau précédent, et le calcul récurrent des d_{ij}. |A| représente la longueur de la chaîne A.

Algorithme X :

```
Début
    d[0, 0] ← 0
    Pour i = 1 à |S1|
         d[i, 0] ← d[i-1, 0] + γ(σ1i → Λ)
    Fin Pour
    Pour j = 1 à |S2|
         d[0, j] ← d[0, j-1] + γ(Λ → σ2j)
    Fin Pour
    Pour i = 1 à |S1|
     Pour j = 1 à |S2|
         m1 ← d[i-1, j-1] + γ(σ1i → σ2j)
         m2 ← d[i-1, j] + γ(σ1i → Λ)
         m3 ← d[i, j-1] + γ(Λ → σ2j)
         d[i, j] = min (m1, m2,m3)
     Fin Pour
    Fin Pour
    Rendre le vecteur d[ ]
Fin
```

L'algorithme X calcule la distance entre deux listes de symboles S_1 et S_2. Cet algorithme a un coût en $O(|S_1| \times |S_2|)$, puisque les deux premières boucles coûtent respectivement $|S_1|$ et $|S_2|$, et la troisième $|S_1| \times |S_2|$. La sortie de l'algorithme est le tableau précédent, dont la lecture donne pour l'exemple précédent d("analogie", "paradoxe") = 5.

Algorithme Y

L'algorithme dit " Y ", produit à partir du vecteur renvoyé par l'algorithme X une trace de coût minimal.

```
Début
    i ← |S1|
    i ← |S2|
    T = {∅}
    Tant que ((i≠0) et (j≠0))
     Si d[i, j] = d[i-1, j] + γ(σ1i → Λ) Alors i ← i - 1
     Sinon   Si d[i, j] = d[i, j-1] + γ(Λ → σ2j) Alors j ← j - 1
         Sinon
             T ← T ∪ (i, j)
             i ← i - 1
             j ← j- 1
         Fin Si
     Fin Si
    Fin Tant que.
    Renvoyer T
Fin
```

Le principe consiste à "remonter" le tableau précédent depuis la case inférieure droite, et à tester à chaque pas quel est le moindre coût, pour connaître la nature de l'opération.

Pour le passage de "analogie" à "paradoxe", l'algorithme Y donne une trace T telle que :

T = (1 2) (2 3) 3 4) (4 5) (5 6) (6 7) (8 8)

Cela s'interprète de la façon suivante : (1 2) montre que la première lettre de "paradoxe" est insérée, puis les lettres se substituent (identiquement ou strictement, l'algorithme ne le précise pas) en diagonale. (2 3) montre par exemple que premier "a" de "analogie" correspond au premier "a" de paradoxe, etc..

L'algorithme X calcule la similarité, et Y localise les substitutions ce qui permet ensuite de procéder une traduction par analogie (nous le montrons plus loin). L'algorithme Y nous donne presque la solution, à un détail près : dans le cas de substitutions, la trace n'indique pas s'il s'agit d'une substitution identique, ou d'une substitution stricte.

Pour remédier à ce manque, nous proposons une légère modification de l'algorithme X qui consiste à noter la nature de l'opération, de manière à obtenir non pas un tableau de cellules de distances, mais un tableau de cellules de couples (distance, opération). Nous nommons cet algorithme X'.

Algorithme X':

```
Début
   d[0, 0] ← 0
   Pour i = 1 à |S1|
        d[i, 0] ← d[i-1, 0] + γ(σ1i → Λ)
   Fin Pour
   Pour j = 1 à |S2|
        d[0, j] ← d[0, j-1] + γ(Λ → σ2j)
   Fin Pour
   Pour i = 1 à |S1|
    Pour j = 1 à |S2|
        m1 ← d[i-1, j-1] + γ(σ1i → σ2j)
        m2 ← d[i-1, j] + γ(σ1i → Λ)
        m3 ← d[i, j-1] + γ(Λ → σ2j)
        Si (σ1i = σ2j) Alors d[i, j] = (m1, '=')
        Sinon   Si ((m1 < m2) et (m1 < m3)) Alors d[i, j] = (m1, '%')
                Sinon   Si (m2 < m3) Alors d[i, j] = (m2, '-')
                        Sinon d[i, j] = (m3, '+')
                        Fin Si
                Fin Si
        Fin Si
    Fin Pour
   Fin Pour
   Rendre le vecteur d[ ]
Fin
```

Le tableau comporte ainsi l'information de l'opération élémentaire qui a donné la distance minimale. La signification des signes est la suivante :

'=' : substitution identique
'%' : substitution stricte
'-' : suppression
'+' : insertion

Nous donnons ci-après l'exemple du tableau donné pour "analogie" et "paradoxe".

	j	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
i			**p**	**a**	**r**	**a**	**d**	**o**	**x**	**e**
0		0	1+	2+	3+	4+	5+	6+	7+	8+
1	**a**	1 -	1%	1=	2+	3=	4+	5+	6+	7+
2	**n**	2%	2%	2%	2%	3%	4%	5%	6%	7%
3	**a**	3 -	2%	2=	3%	2=	3+	4+	5+	6+
4	**l**	4 -	4%	3 -	3%	3 -	3%	4%	5%	6%
5	**o**	5 -	5%	4 -	4%	4%	4%	3=	4+	5+
6	**g**	6 -	6%	5 -	5%	5%	5%	4 -	4%	5%
7	**i**	7 -	7%	6 -	6%	6%	6%	5 -	5%	5%
8	**e**	8 -	8%	7 -	7%	7%	7%	6 -	6%	5=

Figure 8 : Résultat de l'algorithme X'

La place mémoire prise par ce tableau est à peu près double de celle du premier tableau. Pour des chaînes de petites longueurs (comme nous l'envisageons pour la suite), cela reste raisonnable.

Algorithme Y'

Voici maintenant la version modifiée de l'algorithme Y qui produit la trace commentée.

```
Début
    i ← |S1|
    j ← |S2|
    T ← {∅}
    Tant que ((i≠0) et (j≠0))
     Si (d[i, j].y[15] = '-') Alors i ← i - 1
     Sinon   Si (d[i, j].y = '+') Alors j ← j - 1
          Sinon
               Si (d[i, j].y = '%') Alors T ← T ∪ (i, j, '%')
               Sinon T ← T ∪ (i, j, '=')
               Fin Si
               i ← i - 1
               j ← j- 1
          Fin Si
     Fin Si
    Fin Tant que.
    Renvoyer T
Fin
```

La trace renvoyée est composée de triplets (i, j, "opération"). L'opération "=" est une substitution identique, et opération "%" une substitution stricte.

Si l'on reprend l'édition de "analogie" en "paradoxe", la trace se présente comme suit :

T = (1 2 =) (2 3 %) (3 4 =) (4 5 %) (5 6 =) (6 7 %) (8 8 =)

Cette trace montre par exemple que seule la première substitution est identique (elle correspond au remplacement de "a" en "a'", et que la deuxième est stricte ("n" par "p").

[15] d[i, j].y représente la seconde composante de d[i, j], soit le type de l'opération qui a fait passer à cette cellule dans l'algorithme X.

On notera que l'algorithme X' est toujours en $O(|S_1| \times |S_2|)$, et Y' en $O(|S_1| + |S_2|)$.

Conclusion

Ce paragraphe vient donc de donner une "bonne" métrique, la **distance d'édition**, adaptée à la recherche de similarité sur une chaîne (un segment) composée d'éléments. Ces éléments pourront être d'un des types de données que porte une structure TELAM, une des dimensions de l'espace métrique où la distance de base sera la distance d'édition.

Nous avons montré comment l'algorithme X' permet de calculer la distance entre deux segments de longueur respective m et n, en un temps en O(mn), et comment l'algorithme Y' peut, à la suite de X' déterminer complètement l'une des séquences d'éditions qui minimalisent cette distance, à l'aide d'une "trace".

On se rappellera que l'algorithme Y' donne **une** trace qui minimalise la distance entre les deux segments. Dans le cas où **une seule opération a eu lieu**, il n'y a qu'une trace minimale possible, et l'algorithme Y' donne donc **la** trace.

S'il y a plus d'une opération, alors il peut y avoir plusieurs traces possibles, et l'algorithme Y ne donne pas forcément la bonne. Cela a une conséquence importante pour la suite :

On n'utilisera la trace donnée par l'algorithme Y' pour retrouver une édition au niveau des mots entre deux segments S1 et S2, que dans le cas où d(S1, S2) = 1.

3.2 Application à la recherche de similitude entre deux segments

Introduction

La recherche de segments sources dans la mémoire des bi-segments, équivalents au segment source à traduire, est maintenant considérée comme une recherche de similitude dans le cadre décrit ci-dessus. Le passage d'un segment source S1 à un segment source S2 se fait donc par l'intermédiaire des trois opérations d'édition de base (suppression, insertion, substitution) sur, par exemple, les caractères, les mots ou les classes de mots des segments. Nous utilisons les algorithmes X' et Y' pour évaluer cette similarité, et localiser les éléments du segment à traduire qui ont changé par rapport au segment de la mémoire

Il reste à utiliser cette distance au niveau de chaque type de donnée, et à évaluer le seuil à partir duquel les segments seront considérés comme proches. Pour ce faire, le paragraphe qui suit montre comment utiliser la distance d'édition et un seuil relatif sur chacune des dimensions d'un texte multidimensionnel représenté par une structure TELAM, suivant les buts escomptés. Le second précise quels seuils sont utilisés par dimension pour chaque tâche à réaliser. Le troisième montre la généralisation du concept de distance appliquée à deux structures TELAM.

3.2.1 Distance d'édition par type de données

Introduction

Nous avons listé au paragraphe 1.1.3., dix types de données utilisables pour la reconnaissance de similitude entre deux segments. La manipulation de certains types de données étant identique, nous réduirons cette liste à cinq sous-classes :

1. les caractères du texte
2. les mots, lemmes, ou termes
3. les classes de mots (catégories grammaticales ou autres)
4. la représentation sémantique des mots du segment
5. les balises

Nous montrons maintenant comment utiliser chacune de ces sous-classes pour conduire notre recherche de similarité. On notera que les caractères du segment primaire ne sont pas utilisés.

Distance d'édition sur les caractères du texte

Considérons $[\Sigma, \delta]$, où δ est une distance d'édition sur l'ensemble Σ des caractères du texte.

Exemple :

On passe du segment S1 au segment S2 par une insertion de un caractère à la fin.

S1 = "Click_on_the_right_button."

S2 = "Click_on_the_right_buttons."

Le calcul de la distance selon l'algorithme X donne "1", et l'algorithme Y' donne la trace (1 1 =) (2 2 =) .....(24 24 =) (26 26 =)[16], indiquant qu'une insertion à la fin du segment a eu lieu puisque le second segment a 27 caractères.

[16] Les espaces notés '_' et la ponctuation font partie de Σ.

Le passage du segment S1 au segment S3 (présenté ci-après) se fait via une substitution non identique ("o" en "i"), quatre suppressions (trois lettres et un espace) et une insertion de caractères. La distance entre les deux segments vus chacun comme une suite de caractères est donc 6, comme l'indique l'algorithme X'.

S1 = "Click_on_the_red_button."
S3 = "Click_in_the_buttons."

L'algorithme Y' donne la trace suivante :

(1 1 =) (2 2 =).....(6 6 =) (7 7 %) (8 8 =).....(13 13 =) (18 14 =).....(24 20 =)

qui indique qu'il y a eu suppression des caractères 14 à 17 de S1, et insertion du caractère 21 de S3. Le triplet (7 7 %) indique que le "o" a été substitué en "i".

La distance d'édition nous permet de choisir le segment le plus proche au niveau des caractères, et aussi de connaître l'endroit où les modifications ont eu lieu, ce qui est important pour la procédure de traduction qui viendra compléter la recherche d'analogie, et qui est expliquée plus loin.

Distance d'édition de mots, lemmes ou termes

Dans ce cas, l'espace métrique est [V, d], avec V = ∑*. Un segment est alors vu comme une suite de mots, de lemmes ou de termes insécables. Le passage d'un segment à l'autre s'effectue encore par les mêmes d'opérations d'édition (insertion, suppression, substitution) sur les mots, les lemmes ou les termes.

Exemple :

Reprenons les segments S1, S2 et S3 précédents, plus une nouvelle phrase S4, et considérons le segment comme une suite de mots :

S1 = "Click_on_the_right_button"
S2 = "Click_on_the_right_buttons"
S3 = "Click_on_the_buttons."
S4 = "Click_on_the_button."

Le calcul de la distance selon l'algorithme X' donne respectivement :

d(S1, S2) = 1
d(S1, S3) = 1
d(S1, S4) = 1
d(S2, S4) = 2

et l'algorithme Y' donne les traces suivantes :

T(S1, S2) = (1 1 =) (2 2 =) (3 3 =) (4 4 =) (5 5 %)
T(S1, S3) = (1 1 =) (2 2 =) (3 3 =) (5 4 %)
T(S1, S4) = (1 1 =) (2 2 =) (3 3 =) (5 4 =)
T(S2, S3) = (1 1 =) (2 2 =) (3 3 =) (5 4 =)
T(S2, S4) = (1 1 =) (2 2 =) (3 3 =) (5 4 %)

d'où la déduction des opérations sur les mots.

Distance d'édition de classes de mots

Catégories grammaticales augmentées

Comme montré au paragraphe 1.2.3, un segment peut être vu comme la suite des catégories grammaticales de ses mots, plus les occurrences d'une catégorie particulière que nous avons appelé "objet pseudo linguistique", et qui est notée "O". Il existe un différence entre les objets "pseudo linguistiques" comme les variables des fichiers de ressources de programmes, et les objets non-linguistiques comme les marques d'index. Les premiers participent à la structure linguistique du segment, et son considérés comme les instances d'une catégorie syntaxique particulière "O". Les autres ne font pas partie de la structure syntaxique de la phrase et sont donc évalués à part.

Parmi les objets de type "O", on peut voir par exemple des "indices" d'alignement dont on se sert dans [Takahashi 1997]. Dans ces travaux, les valeurs numériques et les noms propres sont utilisés avec succès comme indicateurs d'alignement d'articles de journaux spécialisés entre les langues japonaise et anglaise.

Le même principe de recherche de similarité s'applique cette fois sur [C, d], où C est un ensemble de catégories grammaticales fixé (nom, verbe, adjectif, etc.) plus la catégorie "O". Dans le cas où une seule catégorie grammaticale change, il est encore possible de savoir où se situe ce changement, et de quel type il s'agit.

Exemple :

S1 = (V, Prep, Art, Adj, N)
S2 = (V, Prep, Art, PP, N)
S3 = (V, Prep, Art, Adj, N, Adj)
S4 = (V, Prep, PP, N)

Alors les distances d'éditions sur les catégories grammaticales donnent :

d(S1, S2) = 1
d(S1, S3) = 1
d(S1, S4) = 2
d(S2, S4) = 1

et les traces fournies par l'algorithme Y' :

T(S1, S2) = (1 1 =) (2 2 =) (3 3 =) (4 4 %) (5 5 =)
T(S1, S3) = (1 1 =) (2 2 =) (3 3 =) (4 4 =) (5 5 =)
T(S1, S4) = (1 1 =) (2 2 =) (3 3 %) (5 4 =)
T(S2, S4) = (1 1 =) (2 2 =) (4 3 =) (5 4 =)

Il peut arriver qu'une ambiguïté d'analyse donne plusieurs analyses possibles pour un segment donné. Supposons par exemple que S1 soit à comparer à S2, et que S1 possède deux analyses possibles, et S2 en possède trois.

S1a = (V, Prep, Art, Adj, N) ou S1b = (V, Prep, Art, PP, N)

S2a = (V, Art, Adj, N), S2b = (N, Prep, Art, Adj, N), ou S2c = (V, Prep, Art, N)

Alors la distance entre S1 et S2 sera calculée (par exemple) comme une moyenne des distances. Il est en effet important de ne pas prendre la meilleure des distances car cela peut mener à une perte d'information.

$$D(S1, S2) = 1/6 \sum_{\substack{\alpha=a,b \\ \beta=a,b,c}} d(S1\alpha, S2\beta)$$

Classification en domaines

S'il est difficile de trouver un analyseur qui donne l'ensemble des attributs syntaxiques ou sémantiques d'un lemme, il est par contre assez courant de posséder une base de données terminologiques dans laquelle les termes sont caractérisés par des domaines. Nous pouvons alors envisager la même approche que pour les classes grammaticales. A chaque mot, est affecté un domaine. Ainsi on peut envisager un segment comme la concaténation de l'ensemble des domaines possibles par mot, avec éventuellement des domaines non définis pour certains mots.

Exemple :

S1 = "Open the "Save Menu", and save the file".
S1 : (computer command) ∅ (menu) ∅ (computer command) ∅ (computer)
: (computer command) menu (computer command) computer

S2 = "Open the "Style Menu", and choose the font"
S2 : (computer command) ∅ (menu) ∅ (computer command) ∅ (computer)
: (computer command) menu (computer command) computer

Alors on obtient la distance nulle, ce qui montre que l'on se trouve dans le même domaine, pour la base terminologique.

Autres classes

Ces classes peuvent cependant être définies tout a fait autrement, par convention, comme dans le cas des catégories grammaticales, ou par apprentissage machine, comme par exemple dans [Brown & Della Pietra 1993].

Classification sémantique

Introduction

Au cinquième étage d'une structure TELAM figurent (pourvu qu'un module de classement sémantique soit disponible) les lemmes et l'information sémantique qui leur est attribuée. Nous considérons à partir de maintenant que cette information est donnée sous forme d'un ensemble ordonné de traits sémantiques, par lemme. Si l'on dénote la catégorie sémantique d'un lemme comme une fonction Sem() :

Sem(lemme1) = (t_{11}, …., t_{1p})

Distance d'édition sémantique entre deux lemmes

Nous considérons que le premier trait (trait11) est le plus général, et que le traits sont classés dans une hiérarchie (un arbre ou un treillis). Alors la distance d'un lemme à un autre peut être vue comme une **distance d'édition sur les traits composants la représentation sémantique du lemme**.

Exemple :

Le sens 2 du nom "button" dans Wordnet [Wordnet 1997] a les hyperonymes suivants (que nous considérerons ici comme les attributs sémantiques) :

Sem(button) = (entity, object, artifact, instrumentality, device, mechanism, control, switch, push button)

De même le sens 2 du nom "screen" est qualifié par :

Sem(screen) = (entity, object, artifact, instrumentality, device, electronic device, display, screen)

Ainsi la distance d'édition au sens sémantique entre "button" et "screen" sera 4 car il faut au moins quatre substitutions pour passer d'une liste d'attributs à l'autre.

Soient les deux segments S1 et S2 :

S1 = "He pointed the button"
S2 = "He pointed the screen"

Alors la distance sémantique est celle existant entre "button" et "screen", soit 4.

Exemple de généralisation

Soit S1 et S2 deux segments composés de mots m^k.

$Sk = m^k_1....m^k_n$

Soit une (il y en a plusieurs en général) lemmatisation de chaque mots :

$Sk = l^k_{11}....l^k_{1n}$

et les p attributs sémantiques (du plus général au plus spécialisé) d'un sens et les q attributs d'une autre sens (il y en a aussi plusieurs en général) du lemme de l^k_{ij} tel que :

$l^k_{ij1} = t^k_{ij11}.... t^k_{ij1p}$, $l^k_{ij} = t^k_{ij21}.... t^k_{ij2q}$

Alors on considère λ la distance entre cette suite ordonnée de sens. Cela peut par exemple être (pourvu que l'on commence bien par le plus spécifique à gauche), la distance d'édition sur ces attributs sémantiques, basée sur la distance élémentaire σ entre deux traits sémantiques :

$\sigma (t^k_{ij11}, t^k_{ij21}) = 1$ si $t^k_{ij11} \neq t^k_{ij21}$, 0 si $t^k_{ij11} = t^k_{ij21}$

$\lambda(l^k_{ij}, l^k_{op}) = \min \sigma (t^k_{ij\omega\xi}, t^k_{op\psi\zeta})$

Connaissant ainsi $\lambda(l^k_{ij}, l^k_{op})$ pour tous i, j, o, p, la distance sémantique entre deux segments peut s'exprimer comme la distance d'édition des lemmes considérés comme semmes.

Alors la distance sémantique entre deux segments S1 et S2 peut être vue comme le minimum des distances trouvées en faisant varier les lemmes possibles et les traits sémantiques possibles.

Distance d'édition sur les objets non linguistiques représentés par les balises
Certains objets, comme les variables ont un comportement pseudo-linguistique au sein du segment, et peuvent donc associés aux mots pour la recherche de similarité, comme on l'a vu précédemment (catégorie "O").

Il existe cependant toute une série d'objets, représentés par des balises doubles ou des monobalises dans un segment XML, qui ne s'inscrivent pas dans le cadre précédent. Dans le cas où ces objets sont gardés en mémoire, ils peuvent être de précieux indicateurs de similarité de segment (mais ils sont surtout importants pour la recherche de correspondance entre segments de langue différente, comme nous le verrons ci-après). Ces objets sont représentés soit à l'étage des balises doubles, soit à l'étage des monobalises.

Pour rechercher la similarité de deux segments dans lesquels de tels objets sont représentés, on forme la suite S1 de ces objets dans le segment source du document, la suite similaire S2 dans le segment de la mémoire. Le passage de l'un à l'autre étant considéré comme une édition sur [B, d], où B est l'ensemble des balises. On recherche alors la distance d'édition entre les deux.

Exemple :

S1 = <h1> <h2> <h3>
S2 = <h1> <h2> <h4>
d(S1, S2) = 1

Les segments ci-dessus représentent les doubles balises. Les deux parties d'une balise double classique sont ici concentrées en une balise, mais il n'est pas évident que ce soit la meilleure des solutions. Dans ce cas la distance d'édition sera 1 car seule la dernière balise a été substituée.

3.2.2 Distance et seuil

La distance d'édition d définie précédemment est une fonction entière positive. Nous avons convenu que si $d(S_1, S_2)$ est "petit" alors S_1 et S_2 sont proches. Encore faut-il être plus précis, et donner un seuil (entier) à respecter. Rappelons-nous pour cela les buts que nous voulons assigner à un Outil d'aide à la Traduction Fondée sur la Mémoire :

but 1
Retrouver la traduction de toutes les phrases *exactement* présentes en mémoire.

but 2
Retrouver les phrases *exactes linguistiquement*, mais dont la *mise en forme diffère*, et donner la traduction de la mémoire avec la mise en forme correspondante

but 3
Aller vers une composition de segments exacts de la mémoire ou "presque exacts" pour traduire un segment formé de parties correspondant à ces différents segments dans la mémoire.

but 4
Pour les phrases *presque exactes*, proposer une solution légèrement générée par analogie, plus proche de la solution que la recopie stricte de ce qui est en mémoire

but 5
Donner le plus possible de segments *raisonnablement proches* de la traduction à effectuer au traducteur comme *modèles*.

Voici maintenant, les seuils proposés pour les distances de chaque type de données, qui caractérisent chacune des situations précédentes. Ces seuils correspondent à la limite supérieure des distances $d(S_1, S_2)$ entre le segment S_1 à traduire, et le segment S_2 provenant de la mémoire. Dans le cas de la composition, si l'on tente de traduire S_1 avec k segments S_2^1 S_2^2...S_2^k, il s'agit de la distance entre S_1 et la concaténation des k segments S_2^1 S_2^2...S_2^k.

		exact	forme diff	composition	presque exact	proches
1	**caractères base**	nil	nil	nil	nil	nil
2	**caractères texte**	0	0	>0	>0	>0
3	**mots**	0	0	≤ k+1	1	α_m
4	**lemmes**	0	0	≤ k+1	≤ 1	α_l
5	**termes**	0	0	≤ k+1	1	α_t
6	**catégories gr.**	0	0	0	0	α_{gr}
7	**autres catégories**	0	0	≤ k+1	≤ 1	α_c
8	**sémantique**	0	0	≤ 3	≤ 3	α_s
9	**balises doubles**	0	≥ 0	≥ 0	≥ 0	≥ 0
10	**monobalises**	0	≥ 0	≥ 0	≥ 0	≥ 0

Figure 9 : distances d'édition par type de donnée et par but

Cela appelle plusieurs commentaires :

- Nous avons vu dans la première partie de cette étude que la recherche menée sur l'ensemble des caractères du code primaire peut mener à des aberrations. Nous ne nous servirons donc pas de ce niveau, mais plutôt des données par type. Aucune généralité n'est perdue car tous les types de données sont examinés. Nous obtenons au contraire une plus grande maîtrise du poids de chaque donnée, et restons libres de la combiner dans la distance produit.
- Un segment traduit par un segment de la mémoire dont la forme diffère a une distance nulle au niveau de tous les types de données, sauf pour celles qui gèrent la mise en forme, c'est-à-dire les balises.
- la composition de segments se fera à l'aide de k segments de la mémoire. Pratiquement nous proposons pour k = 2 la notion d'expression pivot. L'expression pivot n'étant pas trouvée dans la concaténation des 2 segments à partir de laquelle elle sera traduite, la distance aux niveau des mots, catégories grammaticales, et lemmes, est celle de ce pivot manquant, soit 1 (si le pivot est composé de p mots, ce sera p). **Mais on impose que la place du mot manquant soit exactement à la jonction des deux segments à utiliser**.
- "presque exact" signifie donc pour la suite qu'exactement les mêmes catégories grammaticales sont retrouvées aux mêmes places, mais qu'une catégorie grammaticale est instanciée par un autre mot (ou lemme). Si en plus les lemmes sont tous égaux, alors il s'agit d'un changement de flexion. Il est intéressant de voir ce qui se passe pour une distance supérieure à 1 pour les mots ou lemmes, tout en

conservant une distance nulle pour les catégories grammaticales (notion d'expressions à "trous", proches des PKB de [Takeda 1994]).

- Dans le cas des segments proches de la mémoire, comme ces segments ne sont qu'un indicateur pour le traducteur, il est important de laisser à celui-ci le soin de gérer lui même les seuils qu'il désire. (d'où les α). Nous considérons qu'une recherche de proximité concerne en soi la proximité linguistique. Si le traducteur pense que l'occurrence d'objets non linguistiques est importante (cf. [Sukehiro 1995] ou [Takahashi 1997]), alors il faut lui laisser le loisir de définir un seuil de balises adapté.
- Les paramètres ci-dessus gagnent à être testés sur des corpus significatifs, et éventuellement adaptés. On sera par exemple peut être amené à restreindre la distance sémantique de 3 à 1 (suivant la hiérarchie utilisée) pour caractériser des segments presqu'exacts. On pourra se rendre compte qu'une distance de 2 et pas de 1 au niveau des mots reste raisonnable. On pourra encore être amené à restreindre le cœfficient α_{gr} caractérisant la proximité syntaxique, pour ne pas autoriser l'utilisateur du système à donner des valeurs non significatives.

3.2.3 Distance d'édition généralisée

Deux segments S1 et S2 étant donnés, ces segment étant vus chacun comme un texte multidimensionnel $T = <S, (\varphi_j)_{1\leq j \leq p}>$, avec $(\varphi_j : M \rightarrow \sum_j)_{1\leq j \leq p}$, nous avons vu plus haut qu'une fois les distances d_j définies sur chacun des vocabulaires $\sum_j$, il est possible de définir une distance d produit sur le segment entier, qui vérifie :

$$d((x_{11}, ...x_{1p}), (x_{21},...,x_{2p})) = \sqrt[n]{\sum_{j=1}^{p} \alpha_j d_j{}^n (x_{1j}, x_{2j})}$$

En prenant p = 10, et les $\sum_j$ les alphabets relatifs à chaque type de donnée décrit plus haut, et en choisissant par exemple n = 1 et $\alpha_j = 1$ (la plus simple des possibilités sur n) :

$$d((x_{11}, ...x_{1\,10}), (x_{21},...,x_{2\,10})) = \sum_{j=1}^{p} \alpha_j d_j\ (x_{1j}, x_{2j})$$

Exemple :
Soient les segments S1 et S2 sous leur segment XML :

S1:<s>There_was_a_time_when_<hi1>James_Brown</hi1><mobj1/>_would_have_said_it.</s>
S2 : <s>There_was_a_time_when_James_Clark_would_have_uttered_it.</s>

Voici maintenant présentées les distances pour chaque type de données, et la distance produit pour tous les α_j égaux à 1. Comme convenu plus haut, la distance sur la forme de base n'est pas incluse dans la distance produit.

p		$d_p(S1,S2)$
1	***caractères base***	*29*
2	**caractères texte**	10
3	**mots**	2
4	**lemmes**	2
5	**termes**	2
6	**catégories gr.**	0
7	**autres catégories**	0
8	**sémantique**	2
9	**balises doubles**	2
10	**monobalises**	1
	distance totale	21

Figure 10 : distance entre deux segments, par étage, et globale

Remarques :

- Certains des α_j peuvent être nuls. Cela peut être utile dans le cas ou l'analyseur de la j-ième donnée n'est pas disponible (pas de lemmatiseur par exemple), ou si l'une des données n'est pas discriminante.
- Les segments sont vus comme le produit du chemin le plus récent de chaque étage d'une structure TELAM. Mais l'on pourrait généraliser la notion de distance d'édition sur des treillis, comme Lepage le fait sur des arbres.
- L'utilisation de la distance d'édition est robuste pour les cas (fréquents) où les analyses (lemmatisation, analyse sémantique, catégorisation) ne sont pas uniques. En supposant que dans le cas ou deux segments sont proches, le même analyseur donne les mêmes ambiguïtés sur les parties communes, nous avons l'option de concaténer l'ensemble des analyses possibles par mot, et demander l'évaluation de la distance d'édition sur l'ensemble. Comme les mêmes ambiguïtés devraient apparaître dans les deux chaînes à comparer, la distance devrait être gardée.

3.3 Premiers résultats

Une première série de tests est en cours de réalisation au moment de la rédaction de ces lignes. Nous espérons pouvoir montrer rapidement l'efficacité de nos considérations.

3.3.1 Implémentation de X' et Y'

Voici en guise de conclusion l'affichage du résultat de l'application des programmes X' et Y' en MAC COMMON LISP sur "analogie" et "paradoxe"[17].

```
? (describe (X' "analogie" "paradoxe"))
#2A(((0) (1) (2) (3) (4) (5) (6) (7) (8))
   ((1) (1 . %) (1 . =) (2 . +) (3 . =) (4 . +) (5 . +) (6 . +) (7 . +))
   ((2) (2 . %) (2 . %) (2 . %) (3 . %) (4 . %) (5 . %) (6 . %) (7 . %))
   ((3) (3 . %) (2 . =) (3 . %) (2 . =) (3 . +) (4 . +) (5 . +) (6 . +))
   ((4) (4 . %) (3 . -) (3 . %) (3 . -) (3 . %) (4 . %) (5 . %) (6 . %))
   ((5) (5 . %) (4 . -) (4 . %) (4 . %) (4 . %) (3 . =) (4 . +) (5 . +))
   ((6) (6 . %) (5 . -) (5 . %) (5 . %) (5 . %) (4 . -) (4 . %) (5 . %))
   ((7) (7 . %) (6 . -) (6 . %) (6 . %) (6 . %) (5 . -) (5 . %) (5 . %))
   ((8) (8 . %) (7 . -) (7 . %) (7 . %) (7 . %) (6 . -) (6 . %) (5 . =)))
Type: (CCL::COMPLEX-ARRAY T (9 9))
Class: #<BUILT-IN-CLASS ARRAY>
Element type: T
Flags: #x10402
Displacement: 0
Displaced to: #<SIMPLE-VECTOR 81>
Dimensions: (9 9)
? (Y' (X' "analogie" "paradoxe"))
(((1 . 2) . =) ((2 . 3) . %) ((3 . 4) . =) ((4 . 5) . %) ((5 . 6) . =) ((6 .
7) . %) ((8 . 8) . =))
?
```

Voici maintenant la même chose pour "bonjour" et "bonsoir".

```
? (describe (X' "bonjour" "bonsoir"))
#2A(((0) (1) (2) (3) (4) (5) (6) (7))
   ((1) (0 . =) (1 . +) (2 . +) (3 . +) (4 . +) (5 . +) (6 . +))
   ((2) (1 . -) (0 . =) (1 . +) (2 . +) (3 . =) (4 . +) (5 . +))
   ((3) (2 . -) (1 . -) (0 . =) (1 . +) (2 . +) (3 . +) (4 . +))
   ((4) (3 . -) (2 . -) (1 . -) (1 . %) (2 . %) (3 . %) (4 . %))
   ((5) (4 . -) (3 . =) (2 . -) (2 . %) (1 . =) (2 . +) (3 . +))
   ((6) (5 . -) (4 . -) (3 . -) (3 . %) (2 . -) (2 . %) (3 . %))
   ((7) (6 . -) (5 . -) (4 . -) (4 . %) (3 . -) (3 . %) (2 . =)))
Type: (CCL::COMPLEX-ARRAY T (8 8))
Class: #<BUILT-IN-CLASS ARRAY>
Element type: T
Flags: #x10402
Displacement: 0
Displaced to: #<SIMPLE-VECTOR 64>
Dimensions: (8 8)
? (Y' (X' "bonjour" "bonsoir"))
(((1 . 1) . =) ((2 . 2) . =) ((3 . 3) . =) ((4 . 4) . %) ((5 . 5) . =) ((6 .
6) . %) ((7 . 7) . =))
?
```

[17] Ces programmes sont dus à Christophe Chenon (stagiaire DEA) qui en est expressément remercié ici.

3.3.2 Comparaison de structures TELA et affichage

Un prototype a été programmé en LISP, avec pour but de démontrer la faisabilité de la comparaison de structures TELA par étages. Le prototype effectue les opérations suivantes :

- on choisit la comparaison sur un étage donné
- pour un segment S0 donné, construction de sa structure TELA associée
  ⇒ l'analyse linguistique se fait par l'appel d'un dictionnaire ad hoc
- pour chaque segment du corpus (50 phrases)
  ⇒ sa structure TELA est crée
  ⇒ la comparaison avec S0 suivant l'étage choisi s'effectue
- le résultat est une liste des trois segments les plus proches au sens de l'étage choisi
- les différentes structures sont alors affichées

Ce prototype montre qu'en effet, il est possible de comparer deux structure TELA. Les temps de réactions sont rapides, malgré la non optimisation du prototype. On attend par exemple moins d'une seconde pour l'ensemble des opérations décrites ci-dessus pour un segment de 15 mots, y compris son affichage et l'affichage du segment proposé.

Conclusion

Ce chapitre a mis en place un formalisme et les méthodes associées pour exprimer et calculer la notion de similarité entre segments. Nous nous fondons pour cela sur la notion de texte telle que [Schreider 1975] l'a exprimée, et celle de distance d'édition formalisée par [Wagner & Fischer 1974].

Il nous est maintenant possible de comparer deux segments sources avec l'algorithme X', et de trouver une séquence d'édition complètement caractérisée par l'algorithme Y'. Dans le cas ou une seule édition a eu lieu, la solution est unique et caractérisée par les algorithmes X' et Y'.

Nous avons aussi montré comment utiliser ces algorithmes sur les données extraites de la représentation en structure TELAM de deux segments sources S1 et S2, pour calculer leur similarité et spécifier leurs différences. Nous avons montré comment différents seuils appliqués à chaque type de donnée permettent de caractériser différentes situations de traduction, de l'identité de S1 et S2, à la similitude proche.

Chapitre 9

Similitude et Traduction Fondée sur la Mémoire

Contenu du Chapitre

Introduction

L'introduction du chapitre précédent rappelle que la traduction fondée sur l'exemple se base sur trois opérations qui sont la recherche de similitude entre deux segments sources S1 et S2, la recherche de correspondance entre S2 et T2 (la traduction de S2), et enfin le transfert des différences entre S1 et S2, via les correspondances entre S2 et T2, de telle façon à pouvoir générer T1, la traduction de S1.

La premier paragraphe montre comment chercher le segment S2 d'une unité de mémoire, le plus proche de S1. Il montre ensuite comment localiser les différences entre S1 et S2. Ces deux opérations se font sur la base de la notion de similitude exposée au chapitre 8. Le deuxième paragraphe formalise dans un premier temps la notion de correspondance grâce à celle de traductibilité, et celle de d'analogie. Il montre ensuite de façon séparée les heuristiques qui nous permettent de trouver les correspondances entre segments source et cibles. Le troisième paragraphe montre enfin comment générer la traduction T1 du segment S1 d'entrée.

Pour conclure, le paragraphe 4 suggère comment pourraient s'organiser les notions développées tout au long de cette étude, pour donner naissance à une nouvelle génération d'Outils de Traduction Fondée sur la Mémoire.

1. Application de la similitude

Introduction

Ayant montré comment la similarité peut être calculée sur les différents types d'information que l'on peut trouver dans un segment, nous allons maintenant donner la procédure complète de comparaison des segments basée sur une structure TELAM.

1.1 Méthode

1.1.1 Du segment à la structure TELAM

Principe

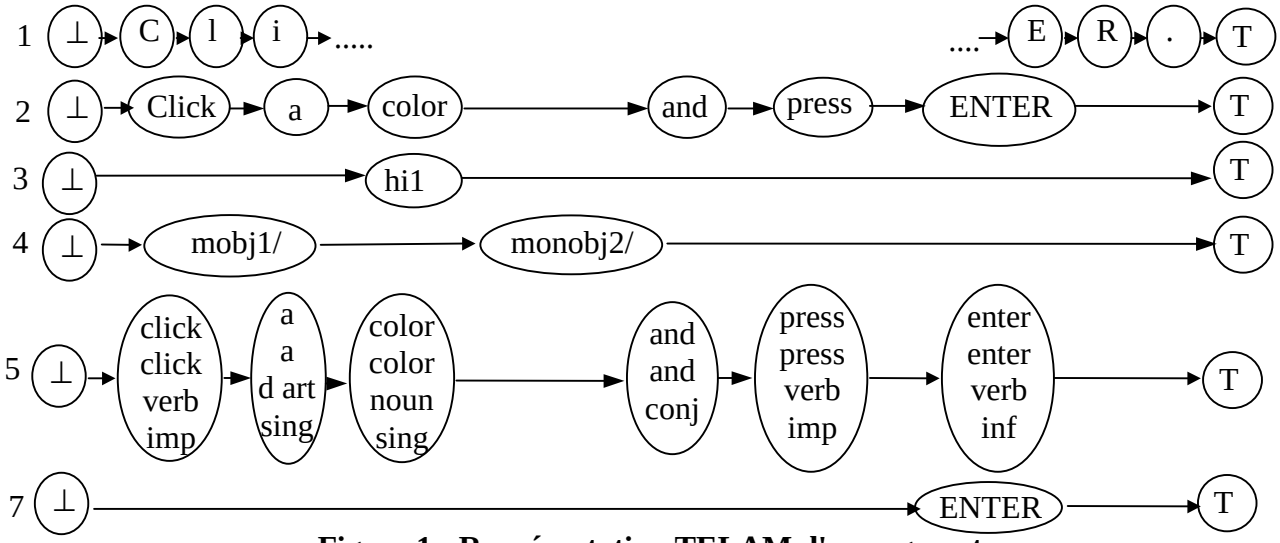


Figure 1 : Représentation TELAM d'un segment

Pour traduire un segment source de document, il peut exister plusieurs unités de mémoire candidates. Le choix de la meilleure unité de mémoire consiste à opter pour l'unité de mémoire dont la distance avec le segment à traduire est la plus petite possible. Les comparaisons se basent sur les structures TELAM correspondantes aux segments à comparer.

Avant toute comparaison, la première opération est donc d'analyser des segments en une structure TELAM comportant 7 étages (nous montrerons plus tard comment calculer le sixième comportant les expressions pivots). Voici par exemple un segment et sa structure TELAM associée :

Codage et transcription des deux segments de caractères

Il est essentiel que la comparaison se fasse sur une représentation homogène des deux segments. L'un des segments provient du document à traduire, l'autre de la partie source de l'unité de traduction utilisée. Les deux segments sont analysés en une structure TELAM dont le premier étage représente l'ensemble des caractères du texte traité. C'est à ce stade que l'on s'assure que les représentations TELAM des deux segments se fait exactement sous le même codage et la même transcription (cf. Chapitre 5). Si ce n'est pas le cas, une étape supplémentaire de transcodage et/ou transcription sera appliquée à l'étage 1 pour créer un nouveau chemin codé et transcrit de façon adéquate.

1.1.2 Indexation et balisage des catégories grammaticales des segments sources de la mémoire

Les segments de la mémoire doivent être indexés pour autoriser une recherche rapide. Une indexation simple peut consister à grouper les occurrences de mots dans un ordre lié à la

langue. Cet ordre peut être alphabétique dans le cas de l'anglais, ou plus élaboré (phonétique plus distinctif d'un des trois systèmes d'écriture plus reposant sur un classement des signes empruntés au chinois, dans le cas du japonais). Nous prendrons le cas de l'anglais (le plus simple) pour la suite, mais pourvu que l'on ait un indexage, le principe est a même. L'indexation se fait en trois étapes :

- affectation d'un identificateur à chaque segment
- balisage des mots par leur catégorie grammaticale (ou une autre classe, mais elle devra être la même que celle utilisée pour l'analyse du texte d'entrée).
- création de l'occurrence des mots. Un mot donné correspond à une entrée de l'index. Sous cette entrée est enregistrée chaque occurrence de ce mot dans les segments de la mémoire. Chaque occurrence est caractérisée dans l'index par l'identificateur du segment qui contient l'occurrence, et sa catégorie grammaticale.
- Structuration en arbre des entrées de l'index (création d'un index sur l'index des mots) de façon que la recherche soit efficace.

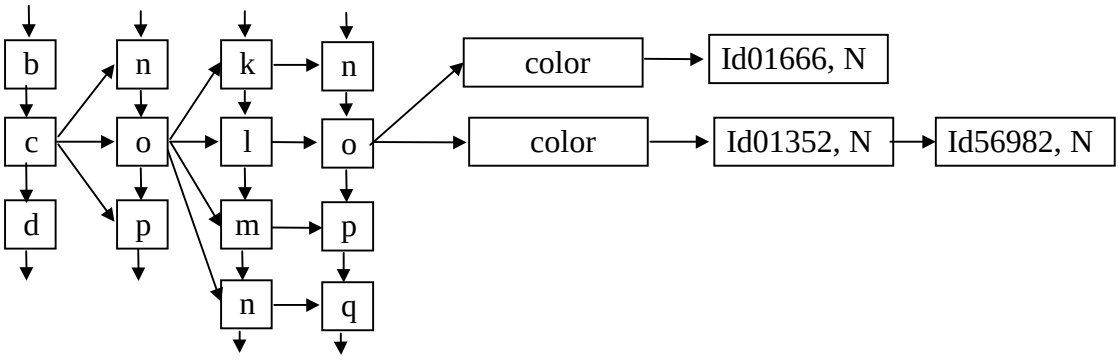


Figure 2 : indexage des mots des segments de la mémoire

Le schéma ci-dessus représente un index d'une mémoire de mots anglais dans laquelle un prè-index sur les lettres de profondeur quatre permet de retrouver "color" en 4n pas, où e "n" est le nombre de mots commençant par "colo" en anglais dans l'index (n vaut à peu près 70 dans le dictionnaire Anglais-Français Harraps Standard).

Dans le cas où l'on possède un lemmatiseur, il est bien sûr préférable de lemmatiser les phrases avant de les indexer, et d'indexer non plus les formes, mais les lemmes. Cela représente un gain de place non négligeable, surtout pour les langues à flexions importantes comme le russe ou l'allemand qui sont des langues à cas.

Dans l'index ne figurent pas un ensemble de mots vides comme les articles ou les prépositions. L'ensemble de ces mots vides doit être fixé. Il sera réutilisé lors de la recherche de segments équivalents.

1.1.3 Recherche de la meilleure unité de traduction

La recherche se fait de façon graduelle pour réduire à chaque étape la complexité de l'opération. Il y a deux grandes phases dans la recherche du meilleur segment : la première est basée sur l'utilisation de l'index et vise à réduire le nombre de phrases candidates pour ne pas effectuer la seconde phase sur l'ensemble du corpus des bi-segments de la mémoire. La seconde consiste à appliquer en couche les recherches de similarité selon différentes données.

1.2 Implémentation de la recherche de la meilleure unité de traduction

1.2.1 Réduction du nombre de bi-segments candidats par recherche basée sur un index

Considérons un segment S1 à traduire. Nous recherchons donc les unités de mémoire dans le corpus des bi-segments indexés, dont le segment source est proche de S1. Soit V_{s1} l'ensemble des mots du segment source s1[1]. Alors la procédure de pré-sélection des unités de traduction consiste à ne garder comme première liste C de bi-segments candidats que les segments du corpus comportant plus qu'un seuil η de mots contenus dans V_{s1}. Soit p le nombre de mots de V_{s1}. Voici une esquisse de l'algorithme :

```
Début
    paramétrer η                                    // c'est une proportion
    C ← {∅}
    Pour i←1 à p
       Rechercher m_i dans l'index
       soit P_mi la liste des (identificateurs des) phrases contenant m_i trouvés dans l'index
       C ← C concat P_mi                            // il faut garder toutes les occurrences
    Fin Pour

    Si (C ≠ {∅})
       CC ← Trier (C)
    Fin Si

    CCC ← {∅}
    c ← Premier(CC)
    Pour suivant(c) de CC
       nb ← 0
       Tant que ((cc ← suivant (CC)) = c)       //CC est triée, on peut donc utiliser "Tant que"
              nb ← nb +1                        //dans l'index, la phrase n'apparaît qu'une fois par mot
       Fin Tant Que                             //nb indique donc le nombre de mots communs de c
       Si (nb/p ≥ η) Alors (CCC ← CCC concat c) Fin Si
       c ← cc
    Fin Pour

    Renvoyer CCC
Fin
```

Algorithme 1 : Sélection des unités de traduction avec l'index

La première partie consiste à faire l'union des identificateurs des segments qui comportent au moins un mot commun, par consultation de l'index du corpus des bi-segments.

La deuxième permet de ne retenir que les segments comportant un taux de mots supérieur au seuil η choisi. Pour cela, on trie d'abord la liste des identificateurs. Si un segment possède plusieurs mots communs avec le segment à traduire, il y aura alors dans cette liste triée autant d'occurrences d'identificateur de ce segment. En les comptant, on peut évaluer le taux de mots communs, et on ne retient que les segments qui ont ce taux supérieur à η.

[1] On a intérêt à n'indexer que les mots non vides au sens de Tesnière. Les mots "vides" comme les articles se répétant beaucoup et n'étant pas spécifiques à tel ou tel segment, leur utilisation alourdit l'index et n'amène pas de discrimination supplémentaire.

Remarque :

- La sélection se fait sur les mots dans l'algorithme précédent. Dans le cas où l'index de la mémoire des bi-segments comporte les catégories grammaticales, il peut être intéressant d'analyser d'abord le segment S1 de façon à obtenir les catégories grammaticales des mots qui le composent, et à faire la sélection sur les catégories grammaticales. La procédure est exactement la même. Cela doit permettre à la fois un nombre de phrases candidates plus élevé, et un silence moins grand.
- Une lemmatisation systématique du corpus des segments bilingues, et une indexation par lemmes, et non pas par mots réduit la taille de l'index et diminue le silence. Une sélection basée sur les lemmes sera donc plus efficace.

1.2.2 Comparaison des segments en cascade

Principe

Ayant réduit le nombre le segments candidats, il est maintenant adéquat d'appliquer les algorithmes de comparaison de distance d'édition sur les segments candidats restants.

Il s'agit maintenant d'appliquer la techniques de recherche de similitude présentée au paragraphe 3.2 du chapitre 8. La comparaison se base donc sur neuf types de données, rappelées dans le tableau 8.

Stratégie

L'importance de chacune des similarités que l'on vient de voir dépend des paramètres suivants :

- *disponibilité des données sur lesquelles se basent ces similitudes* : les modules d'analyse peuvent être absents : il n'est pas évident de trouver un lemmatiseur du serbo-croate par exemple. Les bases de données terminologiques ne sont pas systématiquement constituées dans les projets de traduction.
- *complétude de ces données* : si le module existe, il est très souvent incomplet. Cela entraîne alors un taux de confiance en l'analyse résultante plus ou moins grand.
- *type de document traité* : pour certains documents, certaines données seront plus importantes et d'autres insignifiantes. Les "variables" (faisant partie de la pseudo catégorie syntaxique "O") seront hautement significatives dans le cas des fichiers de ressources, les balises et monobalises seront importantes si dans les mémoires sont conservées les mises en pages.

Une classification adaptative D de l'importance de chaque type de donnée sera donc à effectuer. D est une liste ordonnée de type de données. Ce classement étant défini, la méthode va consister à réduire l'ensemble des unités de traduction candidates à chaque recherche de similarité basée sur un type de donnée. On veillera cependant à garder la mémoire des ensembles successifs obtenus pour pouvoir faire éventuellement un retour en cas de liste finale vide. L'algorithme global donnant l'ensemble des solutions possibles classées par ordre de pertinence se présente alors de la façon qui suit. C est l'ensemble des unités de traduction trouvées précédemment grâce à l'index.

```
Début
    S ← C     // C est l'ensemble des segments candidats trouvés par l'algorithme 1.
    Pour chaque type de donnée di de D
        Pour chaque unité de traduction U de S
        Construire la structure TELAM de la partie source de U
        Extraire les données de l'étage de TELAM correspondant aux données di
        Calculer l'ensemble ordonné Si des unités de traductions proches selon la
                similarité basée sur di, calculée à l'aide l'algorithme X'
```

```
        Garder en mémoire les traces données par l'algorithme Y'
        S ← S ∩ Si
    Fin Pour
    Retourner S
Fin
```

Algorithme 2 : Principe de la recherche de similarité en cascade

Cet algorithme rend donc un ensemble S d'unités de mémoire ordonnées, possédant pour chacune des unités une trace donnée par l'algorithme Y' donnant le nombre minimal de substitutions, d'insertions et de suppressions permettant de passer du segment source à traduire au segment candidat.

Exemple de classification des données

Voici un exemple de classification de l'importance des données. Cette classification proviendra soit de l'intuition (ce cas-ci), soit de l'expérience humaine, soit de l'apprentissage machine. Les différents ensembles P indicés, utilisés ci-après, représentent l'ensemble des unités de traduction sélectionnées, et classé par ordre de similarité pour chaque type de donnée.

```
Début
    Si ∋ catégories grammaticales Alors
        rechercher P_cg
        P ← P_cg
    Sinon
        rechercher similarité sur les caractères P_car
        P ← P_car
    Fin Si
    Si ∋ lemmatiseur Alors
        rechercher P_lem, P ← P ∩ P_lem
    Sinon
        rechercher P_mot, P ← P ∩ P_mot
    Fin Si
    Si ∋ base terminologique Alors
        rechercher P_term, P ← P ∩ P_term
    Fin Si
    Si ∋ autre classification Alors
        rechercher P_class, P ← P ∩ P_class
    Fin Si
    Si ∋ indices non linguistiques Alors
        rechercher P_nl, P Y P ∩ P_nl
    Fin Si
Fin
```

Algorithme 3 : Exemple de recherche de similarité

Remarque :

La distance produit définie au chapitre précédent n'est donc pas directement utilisée pour la recherche de segments candidats. Son utilisation directe amènerait à un calcul de distance entre tous les segments issus de la première sélection au niveau de l'index. L'utilisation en "cascade" des distances permet d'éliminer progressivement des candidats et d'augmenter la

vitesse de la procédure. Cela peut par contre amener du silence, si la classification D précédente n'est pas optimale.

1.2.3 Choix de la meilleure unité de traduction

Le résultat de la phase précédente est donc un ensemble S d'unités de traduction classées par distance croissante par rapport au segment à traduire. Alors soit S est vide, et il n'y a pas de traduction par mémoire possible, soit S est non vide. S'il y a une seule unité de traduction solution, c'est elle qui sera prise en considération par la suite ; si il y en a plusieurs, on garde celle dont la partie source est à la distance la plus faible du segment à traduire.

Conclusion

Efficacité et robustesse de l'approche

L'approche en cascade permet à la fois de réduire à chaque étape le nombre d'unités de traduction à vérifier tout en réduisant le temps de calcul, mais il permet aussi une approche robuste puisque la seule conséquence de la non présence d'une donnée est la non réduction de l'ensemble des unités de traduction (et donc l'augmentation du temps de réponse). Il n'y a cependant pas d'échec de l'algorithme. Nous avons vu que notre approche garantit un silence minimal. Dans le pire des cas, aucune information extérieure n'est disponible. L'algorithme fonde alors sa recherche de similarité sur les caractères et les mots, comme dans les OTFM1g actuels.

1.3 Localisation de la différence entre S1 et S2

Introduction

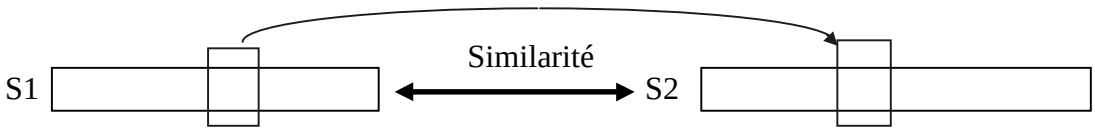


Figure 3 : Localisation des différences entre segments sources.

Ayant trouvé deux segments S1 et S2 proches, nous abordons maintenant la question de savoir où se trouvent exactement les points de différence des deux segments, dans le cas où ils ne sont pas égaux.

1.3.1 Rappel des différents scénarios

Nous reprenons les différents scénarios établis au paragraphe 3.2.2. du chapitre 8, et montrons quelles sont les actions relatives en ce qui concerne la recherche de différence entre S1 et S2. Nous distinguons donc cinq cas d'utilisation du segment S1 sélectionné précédemment comme le plus proche de S1.

Cas 1

S1 = S2. Dans ce cas, il n'y a rien à faire. Il suffit de donner à S1 la traduction T2 de S2.

Cas 2

S1 correspond à S2 linguistiquement, c'est à dire $di(s_{1i}, s_{2i}) = 0$, pour $1 \leq i \leq 8$ (en se référant au tableau 8 du chapitre 8). Il est alors nécessaire de connaître les correspondances entre S1 et S2 pour pouvoir transférer la position des balises, mais cela est immédiat entre S1 et S, puisqu'ils sont égaux.

Cas 3

Une expression pivot pouvant s'appliquer à S1, celui-ci est séparé en deux sous-segments unis par l'expression pivot. Chacun de ces sous-segments est dans l'un des cas 1 ou 2 précédents. On se ramène donc aux cas précédents.

Cas 4

S1 et S2 sont très proches mais non égaux. Leur distance au niveau des mots ou des lemmes est de 1. Le mot qui change d'un segment source à l'autre (de S1 à S2) devra aussi être changé dans T2 pour donner T1 cherché. Il est donc nécessaire de connaître précisément la correspondance entre S1 et S2. Cette correspondance est donnée par l'algorithme Y', puisqu'il n'y a qu'une opération d'édition, et que justement dans ce cas Y' donne exactement les correspondances par la trace. C'est le cas le plus intéressant que nous développerons au niveau du transfert.

Cas 5

La distance entre S1 et S2 étant supérieure à 1 pour les catégories grammaticales, on considère que S2 n'est pas exploitable de façon sûre. On le laisse donc en tant qu'indicateur, traduction approchée pour le traducteur qui jugera de l'utilisation à en faire.

1.3.2 Localisation des différences entre S1 et S2

Nous nous plaçons donc dans le cas 4 décrit précédemment, les autres cas ne demandant pas de localisation de différences entre S1 et S2.

Situation classique

		but 4
		presqu'exact
1	**caractères base**	nil
2	**caractères texte**	>0
3	**mots**	1
4	**lemmes**	≤ 1
5	**termes**	1
6	**catégories gr.**	0
7	**autres catégories**	≤ 1
8	**sémantique**	≤ 3
9	**balises doubles**	≥ 0
10	**monobalises**	≥ 0

La distance entre les catégories grammaticale est nulle. Les catégories de tous les mots se correspondent donc. Il y a donc le même nombre de mots dans S1 et S2. Seules la distances au niveau des lemmes, mots et termes est 1. On en déduit qu'il s'agit d'une substitution. La trace donnée par l'algorithme Y' donne alors l'endroit exact où se situe cette substitution.

Exemple :

S1 : "Press on the right button"

S2 : "Press on the left button"

Trace au niveau des mots : (1 1 =) (2 2 =) (3 3 =) (4 4 %) (5 5 =) montre qu'il s'agit d'une substitution entre "right" et "left".

Une flexion a changé

Un sous-cas particulier est donné pour une distance nulle au niveau des lemmes, alors que la distance au niveau des mots est toujours 1. Il s'agit alors d'un changement de flexion du mot : un passage au pluriel ou un changement de conjugaison.

Exemple :

S1 : "Press on the right buttons"

S2 : "Press on the right button"

Trace au niveau des mots : (1 1 =) (2 2 =) (3 3 =) (4 4 =) (5 5 %)

Un mot a été supprimé ou ajouté

Une évolution de ce cas pourrait permettre par exemple de gérer les adjectifs ajoutés ou supprimés dans un segment. Dans ce cas, la distance au niveau des lemmes est aussi égale à 1 puisque l'on perd ou l'on gagne un mot dans la phrase. Les algorithmes X' et Y' appliqués aux catégories grammaticale permettent de situer avec la trace de la catégorie qui est changée (supprimée ou ajoutée). En limitant ces catégories aux adjectifs ou adverbes par exemple, on devrait pouvoir arriver à construire des phrases linguistiquement correctes en transférant cette différence dans le segment cible.

Exemple :

S1 : "Press on the right button"

S2 : "Press on the button"

Trace : (1 1 =) (2 2 =) (3 3 =) (5 4 =)

1.3.3 Conclusion

L'utilisation conjointe de la distance fournie par l'algorithme X', la trace fournie par l'algorithme Y', et des longueurs des segments et les comparaisons de segments permettent de savoir exactement où se situent les éditions d'un segment S1 à S2, et de quelles éditions il s'agit (insertion, suppression ou substitution non identique).

Il reste maintenant à transférer ces différences sur les parties cibles pour pouvoir générer de nouvelles traductions. C'est ce que les paragraphes suivants proposent.

2. Traductibilité, analogie et mémoires de traduction

La notion de similarité a été décrite au chapitre précédent. L'introduction de ce chapitre rappelle que la traduction fondée sur l'exemple se base aussi sur deux autres : la traductibilité et l'analogie. La traductibilité formalise la notion de correspondance entre les segments source et cible. L'analogie est exposée comme une combinaison des notions de similarité et traductibilité. Nous exposons ces deux notions dans le premier paragraphe.

Dans la pratique, ces notions théoriques mais non constructives (contrairement à celle de distance d'édition) ne nous permettent pas de calculer les correspondances. Nous présentons pour cela des heuristiques au second paragraphe.

2.1 Traductibilité et analogie théorique

2.1.1 Traductibilité

Nous avons montré dans le chapitre précédent comment un segment peut être vu comme un texte T = <S, $(\varphi_j)_{1\le j\le p}$>, représenté par vecteur $(\varphi_1(M)_{,...,}\ \varphi_p(M))$ de l'espace produit des alphabets $\prod_{j=1}^{p}\Sigma_j$, où chaque coordonnée correspond à un "type de données" portées par un étage de la structure TELAM.

Considérons maintenant une unité de traduction, composée d'un segment source S2, et d'un segment cible T2. Ces segments peuvent donc être vus comme des vecteurs $(\varphi^S_{21}(M^S)_{,...,}\ \varphi^S_{2p}(M^S))$ et $(\varphi^T_{21}(M^T)_{,...,}\ \varphi^T_{2p}(M^T))$ représentés chacun par une colonne sur le schéma ci-dessous.

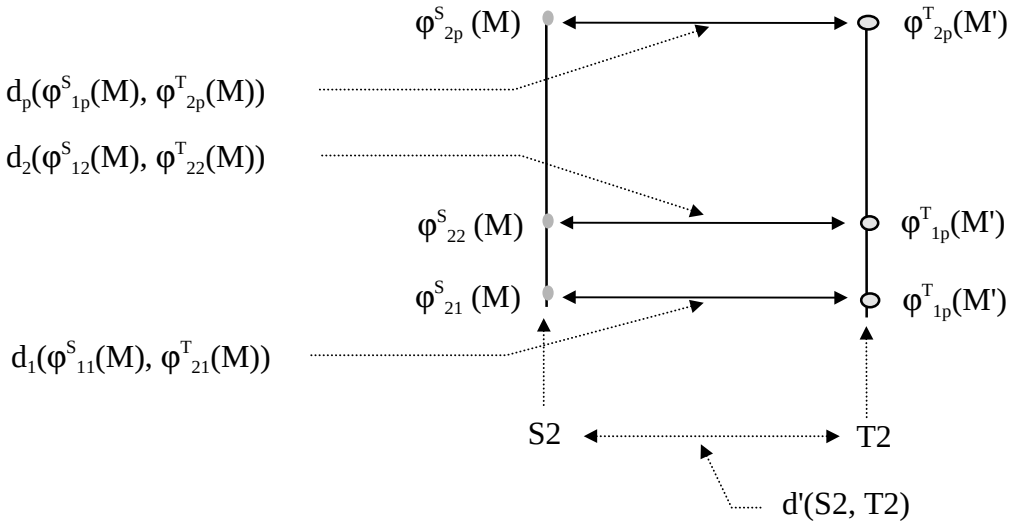


Figure 4 : Distance entre un segment source et un segment cible

De la même façon que la similarité entre deux segments sources est considérée comme le fait que leur distance d soit inférieure à un seuil, la *traductibilité* d'un segment source en un segment cible sera vue comme la distance δ entre les deux segments qui les représentent.

Définition 1 : Traductibilité d'un segment source par un segment cible

Soit un segment source S2 = <S^S, $(\varphi^s_j)_{1\le j\le p}$>, où S^S = <M^S; A^S_1,..., A^S_n> et $(\varphi^S_j : M^S \rightarrow \Sigma^S_j)_{1\le j\le p}$ et un segment cible T2 = <S^T, $(\varphi^T_j)_{1\le j\le p}$>, avec S^T = <M^T; A^T_1,..., A^T_n>, $(\varphi'_j : M^T \rightarrow \Sigma^T_j)_{1\le j\le p}$. Soit l'espace produit des vocabulaires sources $\Sigma^S = \prod_{j=1}^{p} \Sigma^S_j$, et l'espace produit des vocabulaires cibles $\Sigma^T = \prod_{j=1}^{p} \Sigma^T_j$. Soit d' une distance sur Σ^S x Σ^T, et δ un réel positif. La traductibilité de S2 en T2 est définie par : d'(S2, T2) < δ

Intuitivement, cela correspond à la "proximité" des deux colonnes que représentent les vecteurs S2 et T2 dans le plan vertical dessiné sur la figure précédente.

2.1.2 Définition de l'analogie selon Lepage

[Lepage 1997] définit l'analogie de deux couples d'expressions textuelles, (u, v) et (w, x), comme la vérification de trois égalités portant sur une distance d'édition "dist" :

Définition 2 : Analogie de deux couples d'expressions textuelles

$$u : v = w : x \leftrightarrow \begin{cases} dist(u, v) = dist(w, x) \\ dist(u, w) = dist(v, x) \\ dist(v, w) = dist(u, x) \end{cases}$$

On peut énoncer cette analogie de la façon suivante : "v est à u ce que x est à w", et l'on peut interpréter cette analogie comme le respect des contraintes de distances inhérentes à un rectangle, en imaginant que l'analogie consiste pour les expressions textuelles à se placer aux sommets d'un rectangle comme suit :

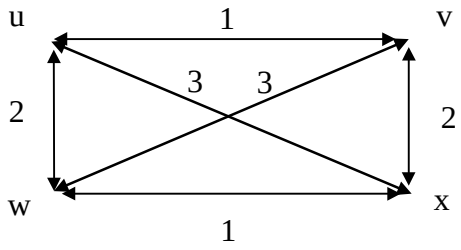


Figure 5 : Illustration de l'analogie selon [Lepage 1997]

Si les trois distances en jeu (les largeurs, les longueurs et les diagonales) sont égales deux à deux (cela correspond aux équations ci-dessus), alors le quadrilatère est un rectangle et l'analogie est vérifiée. Un cas trivial d'analogie est l'identité des couples, qui est illustrée par la réduction du rectangle à un segment, où u = v, et v = w. Le cas extrême est celui où les mots u, v, w, et x sont égaux, réduisant le rectangle à un point.

[Lepage 1996, 1997a, 1997b, 1997c, 1997d, 1998a, 1998b] propose l'utilisation de la notion mathématique de l'analogie pour à la fois mesurer cette analogie, mais aussi pour la générer. Il se base alors sur la distance d'édition introduite plus haut.

Si cette formalisation est non seulement analytique, mais aussi générative dans le cas où u, v, w, et x appartiennent à la même langue (voir les travaux cités ci-dessus), nous ne pensons pas qu'elle soit suffisamment générative pour u et w appartenant à une langue, et v et x à une autre. Nous ne considérerons cette notion d'analogie que de façon analytique.

2.1.3 Analogie généralisée

Dans ce cadre analytique, nous pouvons étendre la notion précédente au cas où u et w appartiennent à une langue, et v et x à une autre. On peut aussi considérer que (u, v) = (S2, T2) est une unité de traduction, et que (w, x) = (S1, T1) en est une autre. Chaque sommet est alors vu comme un segment composé de plusieurs dimensions. La réunion de la figure 1 précédente représentant la distance entre un segment source et un segment cible, et de la figure 3 du chapitre précédent représentant la distance entre deux segments d'une même langue donne alors la figure 3 suivante, évolution naturelle de la figure 2 précédente. Alors l'analogie entre (S1,T1) et (S2,T2) se définit par :

Définition 3 : Analogie de deux unités de traduction

$$(S1, T1) \equiv (S2, T2) \Leftrightarrow \begin{cases} d(S1, S2) = d(T1, T2) \\ d'(S2, T2) = d'(S1, T1) \\ d''(S1, T2) = d''(S2, T1) \end{cases}$$

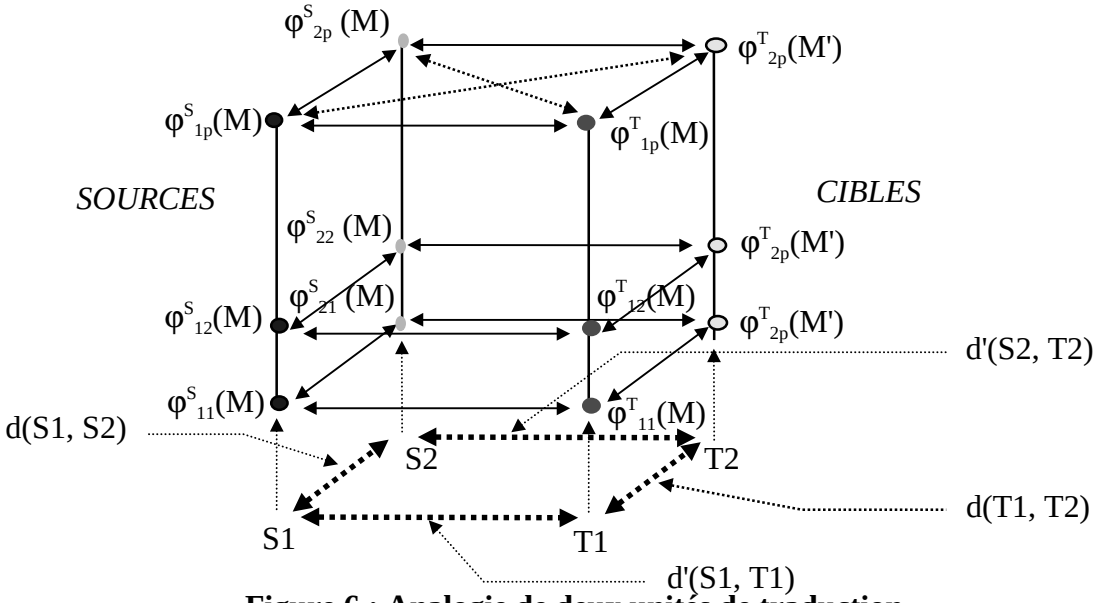


Figure 6 : Analogie de deux unités de traduction

Cette définition appelle plusieurs commentaires. Tout d'abord le choix de la distance d, subordonné à celui des distances par "étage" dont elle est le produit, n'est pas trivial. Si l'on voit en effet comment définir une distance sémantique entre S1 et T1 (avec sur une distance fondée sur une hiérarchie sémantique par exemple), il est plus difficile d'établir une distance entre deux chaînes de mots exprimées en langue différentes.

D'autre part la même distance d est employée pour deux segments sources ou deux segments cibles. Il s'agit d'un abus de notation. Ces distances sont simplement "équivalentes". Dans le cas du français de l'anglais par exemple, les alphabets ne sont pas tout à fait les mêmes, et la distance sera définie sur l'alphabet adéquat.

Enfin, nous insistons sur le fait que la notion d'analogie définie ici n'est qu'analytique et nous servira de cadre théorique, sans donner un cadre constructif. La construction d'une unité de traduction à partir d'une autre se fera plutôt par heuristiques, comme expliqué ci-après.

2.2 Méthodes heuristiques de recherche de correspondance

2.2.1 Principe

Généralités

La recherche d'unités similaires parmi les bi-segments de la mémoire de traduction donne un ensemble S de solutions, qui, s'il est non vide, fourni la meilleure des solutions (S2, T2). Nous nous plaçons maintenant dans le cas où ce segment solution existe.

S1 étant le segment à traduire, on a vu comment localiser les différences entre S1 et S2. Pour pouvoir générer une traduction de S1 à l'aide de S2, le segment le plus proche de la mémoire, nous devons maintenant rechercher les correspondances entre S2 et sa traduction T2.

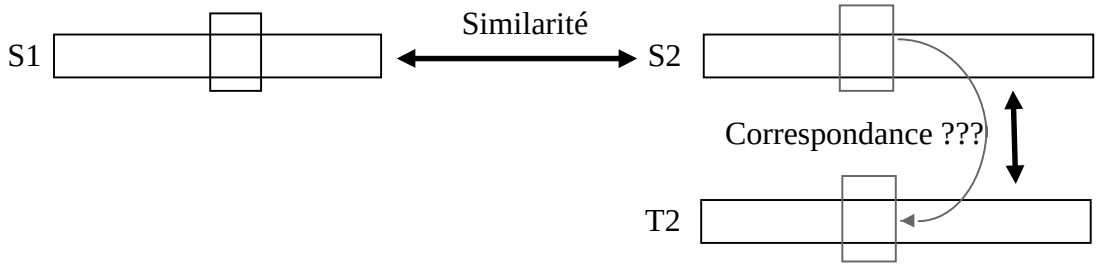


Figure 7 : Recherche de correspondance entre les parties sources et cibles de l'unité de traduction sélectionnée

Nous allons montrer comment trouver ces correspondances à l'aide des différentes données que l'on possède. Ces données sont de deux types : linguistiques et non linguistiques. Dans le cas d'objets non linguistiques, le repérage ne posera pas de problèmes car les objets sont les mêmes en source ou en cible, ou obéissent à une des règles fixes de correspondance (cas des dates par exemple).

Construction de la structure TELAM du segment cible de l'unité de mémoire

Pour rechercher cette correspondance, nous analyserons d'abord S2 et T2 en une structure TELAM. Nous appelons $TELAM_{S1}$, $TELAM_{T1}$, $TELAM_{S2}$ et $TELAM_{T2}$ les structures correspondant à S1, T1, S2 et T2. Elles ne comportent pas encore de sixième étage pour gérer les expressions pivots. Nous montrerons ceci plus loin, aux niveaux des transferts.

Souvent, les segments de la mémoire de traduction actuels ne comportent que les textes, sans objets non textuels. Ces textes comportent cependant des expressions textuelles telles les variables des fichiers ressources qu'il faudra traiter. Notre modèle autorisant cependant la présence de ces éléments dans la mémoire, nous considérerons le cas général où tous les étages de la structure TELAM sont instanciés, même pour les segments S2 et T2 qui proviennent de la mémoire.

Exemple guide

Nous utiliserons dans ce qui suit le segment (S2, T2), supposé en mémoire :

S2 : <s><mobj1>Click a <hi1>color<hi1/>, and press ENTER</s>
T2 : <s><mobj1>Cliquer sur une <hi1>couleur<hi1/>, et appuyez sur ENTREE</s>

Voici maintenant la représentation sous forme de structure TELAM de ces deux segments.

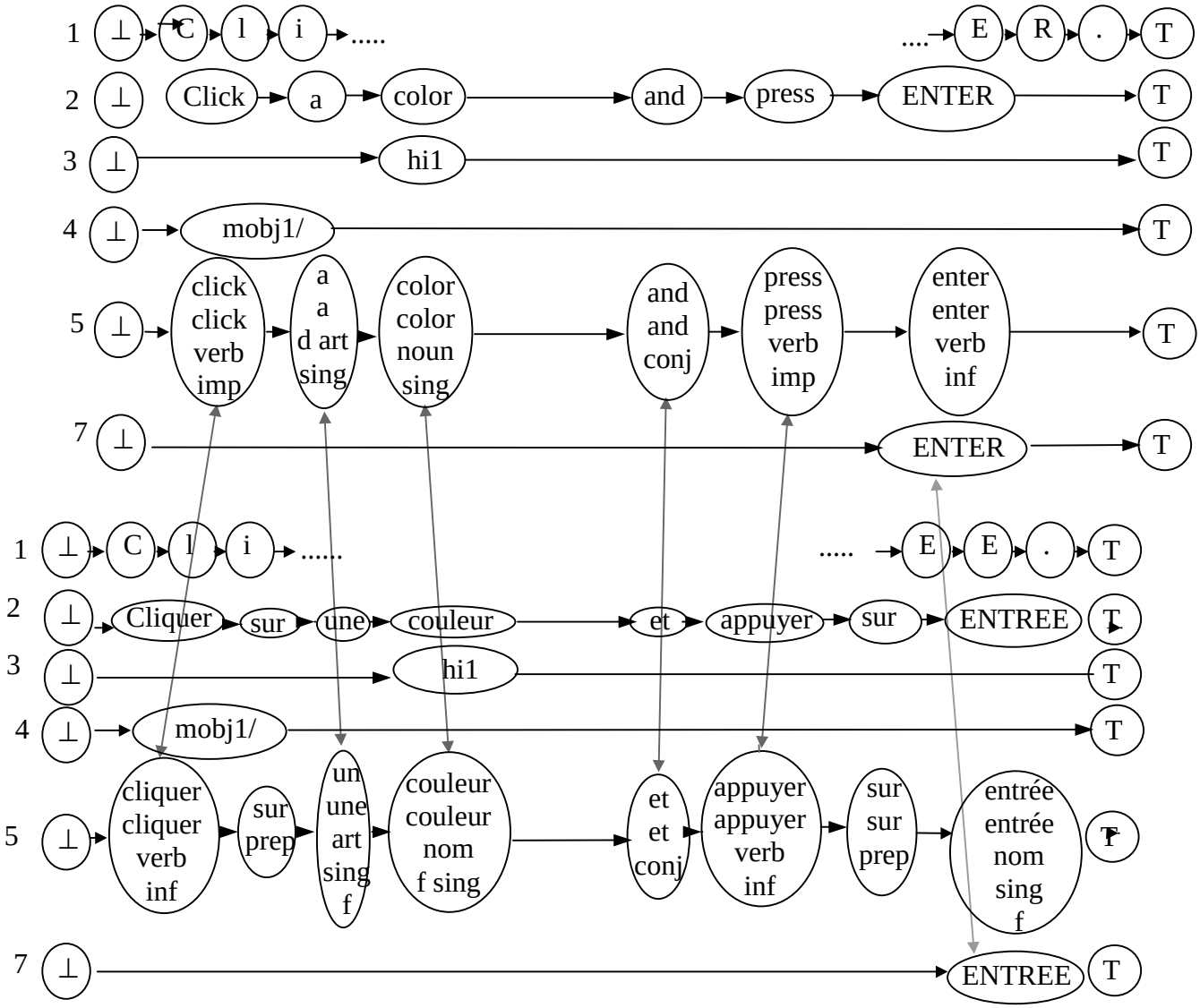


Figure 8 : Correspondance des lemmes entre segments sources et cibles de la mémoire sous forme de structure TELA

Situations à considérer

Le premier paragraphe de ce chapitre a rappelé que les seules situations qui demandent de connaître les correspondances entre S2 et T2, sont celles où il y aura un transfert, à savoir les cas 2, 3, et 4 (transfert de mise en forme, un transfert linguistique concernant un mot au plus, et un transfert via expressions pivots ou expressions à cases). Nous nous plaçons donc dans une de ces trois situations. Dans le cas du transfert linguistique sur un mot, nous considérons que le mot source concerné est noté s_{i0}, et le mot cible t_{i0}.

2.2.2 Utilisation de critères linguistiques

Nous présentons maintenant différentes techniques de mise en correspondance des mots ou des objets de S2 et T2.

Utilisation du lexique bilingue des lemmes
La façon la plus sûre de repérer la correspondance entre les deux représentations est de se baser sur la correspondance des lemmes via un lexique bilingue. Cela suppose donc que l'on a la structure source $TELAM_{S2}$ complète et que la structure cible $TELAM_{T2}$ possède au moins les lemmes correspondant au mot s_{i0} source et au mot correspondant cible t_{i0}.

Soit l'analyse ambiguë en lemmes de S2 et T2. Chaque mot peut correspondre à plusieurs lemmes, et cela donne donc des données sous la forme suivante :

S2 : $(l^S_{11}...l^S_{1q1})$ $(l^S_{21}...l^S_{2q2})$..... $(l^S_{i01}...l^S_{i0q2})$.... $(l^S_{n1}...l^S_{nqn})$
T2 : $(l^T_{11}...l^T_{qp1})$ $(l^T_{21}...l^T_{2p2})$..... $(l^S_{i01}...l^S_{i0q2})$... $(l^T_{m1}...l^T_{nqm})$

L'algorithme de recherche de correspondance consiste à parcourir l'ensemble des lemmes l^T_{ij} possibles de T2 pour un l^S_{i1j1} fixé. On vérifie alors si le lemme l^T_{ij} correspond au lemme l^S_{i1j1}, et on retient ce lien si c'est le cas.

Les estampilles et les échelles linéaires de $TELAM_{S2}$ et $TELAM_{T2}$, permettent alors d'en déduire la correspondance entre les étages 1, 2, et 7.

Les liens bilingues ainsi trouvés sont spécifiés au niveau du treillis par des liaisons de type "bilingue" (voir figure 7).

Remarque :
La correspondance peut n'être que partielle pour les raisons suivantes :

- ambiguïté de correspondance (rare)
- incomplétude du dictionnaire de lemmes (moins rare)
- non correspondance bijective entre les langues sources et cibles (fréquent)

Bases terminologiques bilingues
Si ces bases terminologiques existent, elles constituent souvent la plus sûre des sources de correspondance car elles sont en général spécialisées pour le type de document traité. La technique est donc la même que précédemment, mais le segment est cette fois vu comme une suite de termes extraite de l'étage 7. Il ne correspond en général qu'un terme à un mot donné. La même recherche de correspondance est menée. Cette correspondance lie alors les étages 7 sources et cibles, et la correspondance est éventuellement propagée grâce aux estampilles et échelles linéaires aux autres treillis des TELAM. Ces glossaires ne sont en général pas completes. Ils délaissent en particulier les termes courants pour ne se concentrer que sur les spécialisés. Dans la figure ci-dessous, le lien entre "ENTER" et "ENTREE" est typiquement trouvé par consultation d'une base terminologique. Ces critères étant robustes, ils seront appliqués en premier. Les liens sont aussi matérialisés par des liaisons au niveau des représentations TELAM des segments sources et cibles.

Utilisation d'autres critères linguistiques
Il peut être intéressant et efficace d'utiliser des "patrons linguistiques". Tel patron est une expression hétérogène comportant à la fois des lemmes et des catégories grammaticales formant des modèles de correspondance. Cette technique a été utilisée dans [Takeda 1994]

avec un certain succès, et est appelée "PKS" pour "Portable Knowledge Source". Voici un exemple tiré de [Takeda 1994] :

Dans la phrase "Delete the line", le traducteur peut avoir précisé, lors d'une traduction précédente, le sens de "line" avec le PKS suivant :

```
(PK1 ("line" (cat n)) (sense 1))
```

Ce patron, appliqué à la structure TELAM, réduit alors l'ambiguïté de lemmatisation, et la recherche de correspondance en source et cible en est simplifiée.

Cependant un problème de taille est que ces patrons doivent être construits à la main par les terminologues ou traducteurs en plus de leur travail, et qu'ils ne sont pas acquis automatiquement. Il serait intéressant de prolonger cette idée en allant vers des techniques d'apprentissage de ces patrons.

On notera d'autre part que ces patrons peuvent être exprimés sans difficulté avec les "schémas de liaison" définis dans la partie 2. Les schémas de liaison englobent en fait la notion de PKS.

2.2.3 Utilisation d'indices non linguistiques

Nous l'avons vu, les segment comportent des objets non linguistiques. Parmi ces objets, nous en distinguons deux types :

- les objets qui ont un comportement "pseudo linguistique", représentés à l'étage des mots.
- les autres objets, inclus dans le segment XML pour autre chose que la formation de la phrase, comme par exemple les marques diverses (index, révisions, etc.), représentés à l'étage des monobalises.

Objets au comportement "pseudo-linguistique
En voici tout d'abord quelques exemples pour fixer les idées :

- termes de la base terminologique
- variables informatiques (ressources ou d'aide en ligne)
- expressions numériques : dates, chiffres, sommes d'argent, etc..
- noms propres de lieu géographique ou de personnes

Les objets de ce type se divisent eux-mêmes en deux :

- les objets invariables d'une langue à l'autre (variables informatiques par exemple) :

  <idx/> <rev/> sont par exemple une balise d'index de révision.

- les objets soumis aux variations de langues ou de cultures (les trois autres cités ci-dessus) comme :

samedi 2 mai 1998 → Sàbado a 2 de mayo
Venezzia → Venise
<lang = EN> <lang = FR>

Les termes récupérables dans une base terminologiques, qui pourraient être considérés comme des objets variables, ont déjà été vus. Les objets de type invariable forment des

indices de choix pour la recherche de correspondance, et leur utilisation est immédiate. En ce qui concerne les autres, cela nécessite soit un glossaire, soit une série d'expressions régulières ou petits automates gérant le passage d'une langue à l'autre. L'inclusion de ces mécanismes devrait permettre d'obtenir des correspondances intéressantes. Cette idée a été utilisée dans [Takahashi 1997] avec succès.

Les marques index, qui sont des objets non linguistiques fréquents, sont généralement attachées à un mot (par convention : ils se placent juste à leur suite, par exemple). En ce sens, ce sont de précieux indicateurs puisqu'ils qualifient les mots en langue source et en langue cible.

Utilisation des balises doubles

Dans les types de documents évoqués en première partie, les balises doubles indiquent généralement l'emplacement des mises en forme. Si cette mise en forme est enregistrée en mémoire, elle est en général très significative de leur correspondance.

3. Heuristiques de transfert

Introduction

Ayant trouvé les différences entre les deux segments sources S1 et S2, et la correspondance entre les segments sources et cibles S2 et T2 de la mémoire, il est maintenant possible de générer le segment cible T1 correspondant.

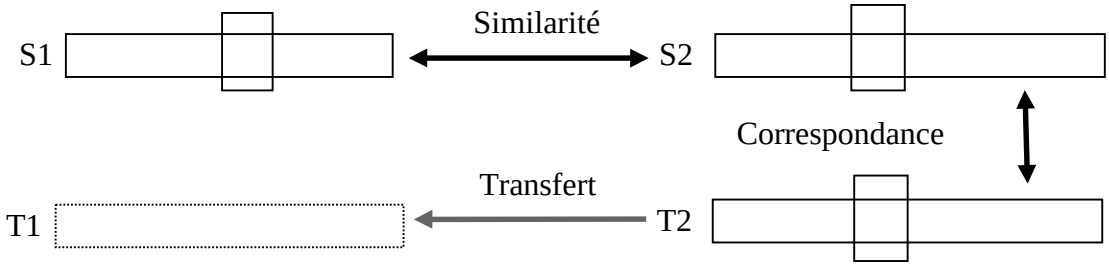


Figure 9 : Transfert vers la traduction recherchée

Nous explorons dans un premier temps les processus généraux de transfert linguistique et de mise en forme au premier paragraphe. Ce cas est en effet le seul cas fondamental où nous autorisons une génération. Puis deux heuristiques sont proposées : le recours à des expressions à cas (paragraphe 2), et à des expressions pivots (paragraphe 3).

3.1 Transfert et génération

3.1.1 Cas où $d_{gr}(S1, S2) = 0$ et $d_{mots}(S12, S2) = 1$

Procédure globale

Dans ce cas, nous avons montré qu'il est possible de connaître exactement le mot qui fait la différence entre S1 et S2. Appelons s_{1i0} le mot changé dans S1, et s_{2i0} le mot remplaçant s_{1i0} dans S2. On considérera soit la forme de ce mot si l'on se place au deuxième étage de la structure TELAM, soit le lemme associé au cinquième étage.

Nous venons d'autre part de voir comment trouver les correspondances entre les parties source S2 et cible T2 de l'unité de traduction la plus proche du segment S1 à traduire. Soit t_{1i0} le mot traduction de s_{1i0}, et t_{2i0} le mot traduction correspondant à s_{2i0}.

L'unité de traduction (S2, T2) sert maintenant de "modèle" pour générer T1. Le remplacement du mot t_{2i0} par t_{1i0} dans la structure $TELAM_{T2}$ donne alors la structure $TELAM_{T1}$ du segment T1 cherché.

Exemple :

Supposons que l'on veuille traduire le segment S1 :

```
S2 : <s><mobj1/>Click a <hi1>shape<hi1/>, and press ENTER</s>
```

Supposons aussi que l'unité de traduction la plus proche est (S2, T2) suivante :

```
S2 : <s><mobj1/>Click a <hi1>color<hi1/>, and press ENTER</s>
T2 : <s><mobj1/>Cliquer sur une <hi1>couleur<hi1/>, et appuyez sur ENTREE</s>
```

S1, S2 et T2 sont analysés en structures TELAM, donnant $TELAM_{S1}$, $TELAM_{S2}$, et $TELAM_{T2}$ (voir p 274). L'extraction des informations des étages 2 et 6 de $TELAM_{S1}$ et $TELAM_{S2}$ donne en particulier les chaînes de mots et de catégories grammaticales :

S1 : (click a shape and press ENTER), (V Art N Conj V Nom)
S2 : (click a color and press ENTER), (V Art N Conj V Nom)

De l'application des algorithmes X' et Y', nous déduisons :

$d_{gr}(S1, S2) = 0$, $d_{mots}(S1, S2) = 1$
s_{1i0} = "shape", s_{2i0} = "color"

La recherche de correspondance nous permet de trouver :

s_{2i0} = "color", par t_{2i0} = "couleur"

Une consultation du dictionnaire donne t_{1i0} = "forme", et sa substitution à t_{2i0} = "couleur" nous permet de récupérer la structure $TELAM_{T1}$ suivante :

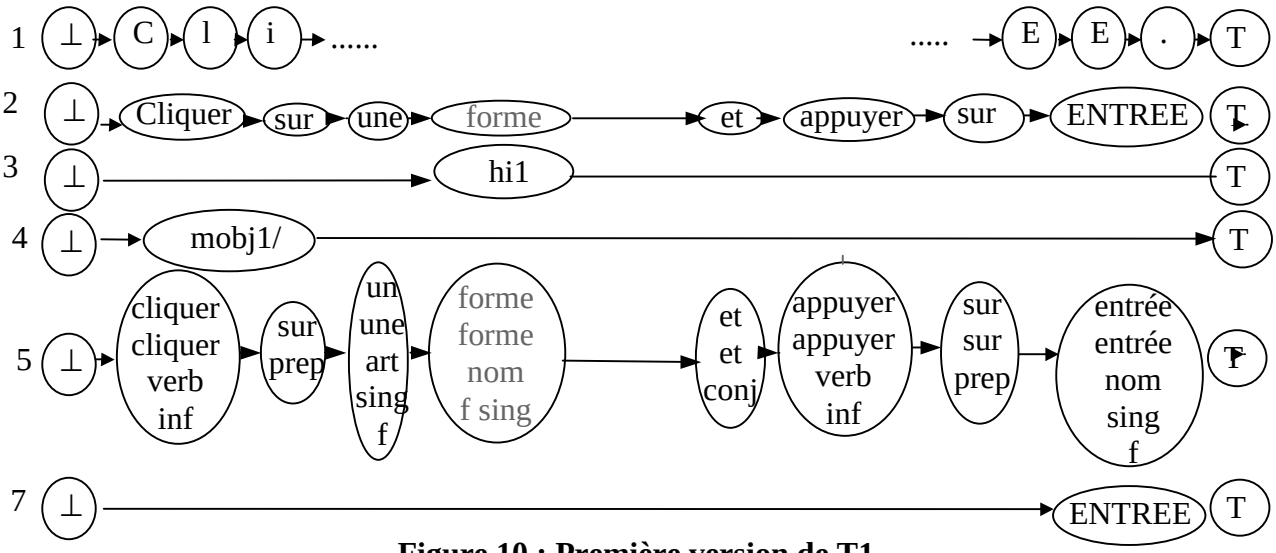


Figure 10 : Première version de T1

La génération du segment XML à partir de cette structure donne le segment T1 :

T1 : <s><mobj1/>Cliquer sur une <hi1>forme<hi1/>, et appuyez sur ENTREE</s>

On remarque que la mise en forme est conservée, et que l'objet non linguistique <mobj1/> est correctement placé.

Génération morphologique

Lors du remplacement du mot "couleur" par "forme", la conservation de l'information portant sur les attributs linguistiques permet de spécifier que forme est au singulier. Les informations complémentaires venant du dictionnaire donnent en particulier le genre de forme.

Dans le cas ci-dessus, "forme" et "couleur" ont tous les deux le même nombre et le même genre, ce qui ne pose pas de problème au niveau de leur morphologie. Si "forme" avait été au pluriel, ou remplacé par "élément" qui est masculin, cela aurait posé des problèmes de morphologie au niveau du mot "élément" lui-même, mais aussi au niveau des mots avec lesquels il s'accordent (l'article ici). Il est donc nécessaire de compléter cette génération par une génération morphologique. Deux situations sont alors possibles :

- soit le système dans lequel on se trouve possède un générateur morphologique pour la langue cible, et on lui donne l'ensemble des lemmes et leurs attributs pour que la génération soit effectivement réalisée.
- soit le système ne possède pas de tel générateur. La phrase résultat est donc fausse au niveau morphologique. Ceci doit être signalé au traducteur en la présentant d'une façon différente. Nous pensons cependant que le fait d'avoir le lemme correct dans sa forme canonique est utile au traducteur. D'autre part, un correcteur orthographique peut être utilisé pour générer la bonne morphologie.

Base terminologique
Si le mot à remplacer avait été "ENTER" par "DOWN ARROW", et dans le cas où ce dernier terme se trouve dans la base terminologique, c'est la traduction donnée par cette base terminologique qui sera utilisée en priorité.

3.1.2 Transfert non linguistique

La génération du segment T1 comprend d'une part le transfert linguistique que nous venons d'aborder, et d'autre part le transfert non linguistique concernant la mise en forme, et des objets non linguistiques. Ces éléments sont portés par les balises doubles et les monobalises aux étages 3 et 4 des structures TELAM.

Dans l'exemple précédent, l'ensemble de la mise en forme et des objets de S1 et celle de S2 était identique, ce qui n'a pas posé de problèmes de transfert : T1 a simplement hérité des éléments non linguistiques présents dans T2.

Supposons maintenant que la mémoire ne comporte pas de mise en forme ou d'objets. Cela est en fait le plus souvent le cas. La structures $TELAM_{S2}$ est montrée dans la figure 10.

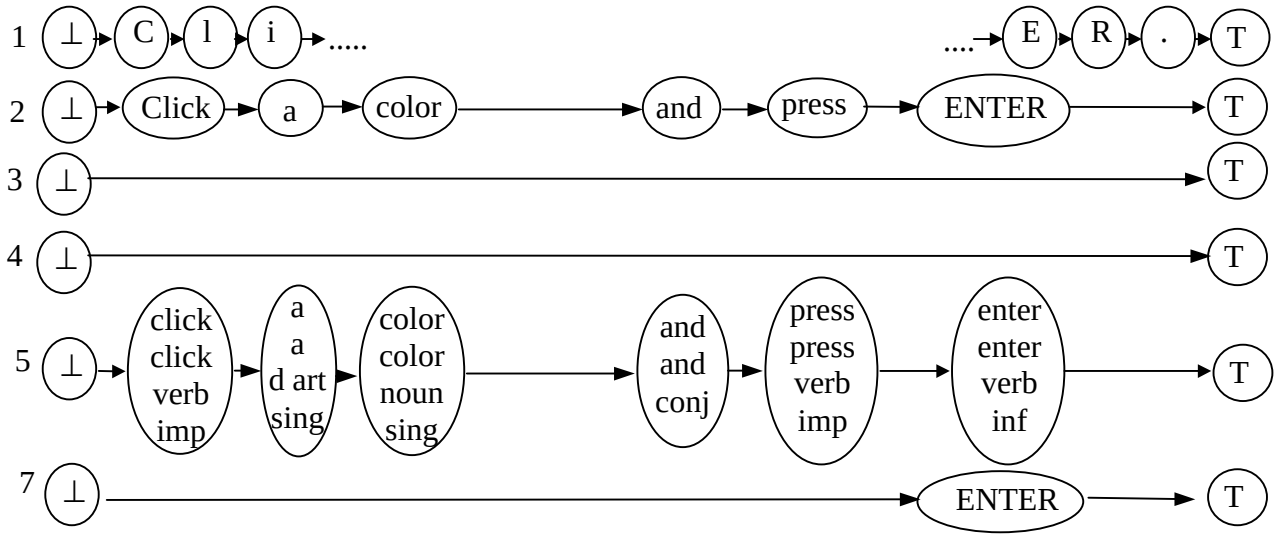


Figure 11 : Unité de mémoire dépourvue d'objets non linguistiques

Si d'autre part, le segment à traduire est toujours S1 (dont la structure $TELAM_{S1}$ est montrée par la figure 11), et comporte donc des attributs non linguistiques, il est nécessaire de gérer le transfert de format de S1 à T1 en décorant T2 du format porté par S1.

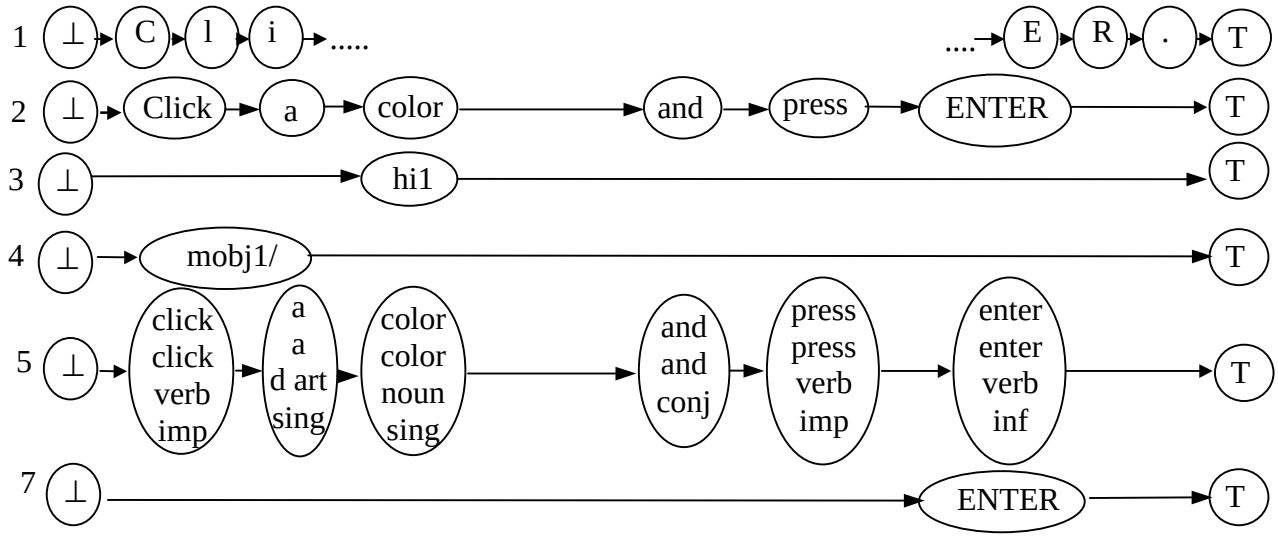


Figure 12 : Analyse de S1

Pour générer les objets non linguistiques de T1, il suffit alors de :

- copier les étages 3 et 4 de S1 sur S2
- les correspondances entre S1 et S2 étant trouvées comme indiqué dans le paragraphe précédent, déduire les liaisons équivalentes à celles de S1 entre les étages 3 et 4, et les autres sur S2. On utilise alors la combinaison des correspondances et des échelles linéaires.
- transférer les étages 3 et 4 plus les liaisons grâce aux correspondances entre S2 et T2.
- générer le segment T1 résultant.

La structure $TELAM_{T1}$ générée et le segment T1 sont montrés par la figure 9 précédente.

3.1.3 Autres transferts

Les mécanismes étudiés ci-dessus permettent de traiter de nouveaux phénomènes, et ne sont pas restreints aux seuls transferts explicités ci-dessus. Nous montrons maintenant un exemple pratique de leur utilisation. Il s'agit de gérer les transferts de "raccourcis claviers". On trouve ces raccourcis par exemple dans les menus d'applications d'édition ("Ctrl N" pour créer un nouveau fichier par exemple). Voici quelques exemples de raccourcis dans le tableau suivant.

Représentation dans la base terminologique

terme S	type racc. S	lettre racc. S	terme T	type racc. T	lettre racc. T
exit	Ctr	2	sortie	Ctr	1
find	Ctr	1	rechercher	Ctr	1
help	Fct	F1	aide	Fct	F1
macro	Fct	F11	macro	Ctr	1

Ce tableau présente une partie de base terminologique. Parmi les champs de la base terminologique, outre les champs pour le terme source et le terme cible, quatre champs supplémentaires sont alloués par terme :

- type raccourci source
- lettre raccourci source
- type raccourci cible
- lettre raccourci cible

Un type de raccourci source ou cible est classique et consiste à utiliser une lettre associée à une touche générique (Control, Alt,..). Ce type de raccourci est indiqué dans le champ de type de raccourci (ex : "Ctr"). La lettre qui doit être utilisée est alors indiquée par son numéro d'occurrence dans le mot (2 pour le X de eXit, par exemple). En indiquant la lettre pour le terme source et le terme cible, le transfert peut alors se faire.

Il se peut que, pour une langue donnée un type de raccourci clavier soit utilisé, et dans une autre langue, ce soit un autre type. Cette gestion est rendue possible grâce à la présence non pas d'un champ de type de raccourci, mais un champ de type de raccourci pour une langue donnée. Cela autorise alors à indiquer que le terme "macro" correspond à l'utilisation de la touche F11 du clavier en anglais, alors que son équivalent français ("macro" aussi), correspond plutôt au raccourci "Ctr M".

Utilisation en phase de transfert

Au niveau des chaînes de caractères utilisées dans certaines ressources de logiciels (d'édition en particulier), les raccourcis claviers ne sont interprétés que s'ils apparaissent sous une certaine mise en forme, comme par exemple : "e<u>X</u>it" (majuscule et soulignement simple).

Lors de la traduction de ce genre de documents, il faut pouvoir générer automatiquement ce genre de formatage. En supposant que les champs au niveau de la base de données terminologique sont remplis comme vu ci-dessus, le procédé pourrait être :

1. placer les termes dans des nœuds de termes de la structure TELAM source et cible (la recherche se fait sur le texte nu des mots du deuxième étage, la mise en forme ne gène donc pas).
2. rechercher les champs relatifs aux raccourcis claviers
3. créer la liaison bilingue engendrée entre le nœud du terme source et le nœud du terme cible.
4. créer au troisième étage cible les nœuds de balises de raccourci clavier s'appliquant sur les nœuds de caractères.

Alors, pourvu qu'une instruction sur la feuille de style XSL correspondante pour l'application de la mise en forme existe et corresponde à la mise en forme voulue, la génération de la mise en forme se fera naturellement au moment de l'application au segment XML créé de la feuille de style générale.

3.2 Composition de segments à l'aide d'expressions pivot

Nous avons proposé au chapitre 5 la notion de phrase à pivot (p 144). Une telle phrase contient une expression pivot invariable qui coupe la phrase en deux parties A et B :

A (expression pivot) B

On pourrait imaginer une locution pivot coupant la phrase S1 plus généralement en k parties, mais cela est moins utile car ce genre de situation est plus rare[2]. Le raisonnement qui va être expliqué maintenant s'appliquerait cependant tout aussi bien. Considérons donc que l'expression pivot source comporte en tout k_S mots notés π^s_1 à π^s_k.

[2] La phrase "Premièrement A, puis B, et enfin C" est un exemple de phrase à pivot coupée en trois parties.

Ayant analysé la phrase S1 et trouvé une expression pivot (par recherche de l'occurrence d'expressions stockées dans une liste finie d'expression pivots), l'étage de l'expression pivot a pu être construit, et donne une structure $TELAM_{S1}$ source, et le début de la structure TELAM cible constituée seulement de l'expression pivot cible. Cette expression pivot cible a été en effet trouvée dans la partie droite de la liste d'expressions pivots. Supposons qu'elle comporte k_T mots. Voici comment se présentent les listes d'expressions pivots stockées de l'anglais vers le français :

source	cible
in order to	pour
so that	de telle façon à ce que
nevertheless	néanmoins
therefore	donc
because	parce que

Table : Exemples d'expressions pivots

Nous avons donné au chapitre précédent les distances liées aux phrases pivots que nous rappelons ici.

		but 3
		(composition)
1	**caractères base**	nil
2	**caractères texte**	>0
3	**mots**	≤ k+1
4	**lemmes**	≤ k+1
5	**termes**	≤ k+1
6	**catégories gr.**	0
7	**autres catégories**	≤ k+1
8	**sémantique**	≤ 3
9	**balises doubles**	≥ 0
10	**monobalises**	≥ 0

Table : distances associées à une phrase à pivot

L'autorisation d'une distance au plus de k pour les mots (par exemple) est liée au fait que la phrase à pivot peut être coupée en k parties. Chaque partie est alors traitée comme un segment isolé. Chacun de ces sous segments isolés $S1^q$ peut alors se trouvé exactement traduit, ou au pire à une distance de 1 au niveau des mots (tout en gardant une distance syntaxique nulle) d'un sous-segment $S2^q$. Si ces k parties se trouvent à une distance de 1 de leurs segments $S2^q$, alors l'ensemble du segment S1 (qui est la concaténation des segments $S1^q$ et des k-1 parties de l'expression pivot) se trouve à une distance k au niveau des mots de S2 (qui lui est la concaténation des segments $S1^q$ et des k-1 parties de l'expression pivot). Voyons cette situation sur un schéma, pour le cas où k =2.

Le segment S1 est donc ici composé de deux segments $S1^1$ et $S1^2$ liés par l'expression pivot "and" dont l'équivalent en français est "et". Pour traduire S1, la méthode consiste alors à rechercher dans la mémoire des bi-segments un bi-segment pour chacun des sous-segments $S1^1$ et $S1^2$. On applique pour cela les méthodes expliquées plus haut.

Si pour chacun des sous-segments sources il existe effectivement des segments $S2^1$ et $S2^2$ correspondants dans la mémoire tels que :

$$d_{gr}(S1^1\ S2^1) = 0 = d_{gr}(S1^2, S2^2) \text{ et } d_{mot}(S1^1\ S2^1) \leq 1 \geq d_{mot}(S1^2, S2^2)$$

S1 est traduisible, et l'on applique à chaque sous-segment les transferts expliqués plus haut. En "concaténant" les deux parties $T1^1$ et $T1^2$ trouvées comme traductions respectives de $S1^1$ et $S1^2$, avec l'expression pivot cible ("et" ici), on trouve alors la traduction désirée.

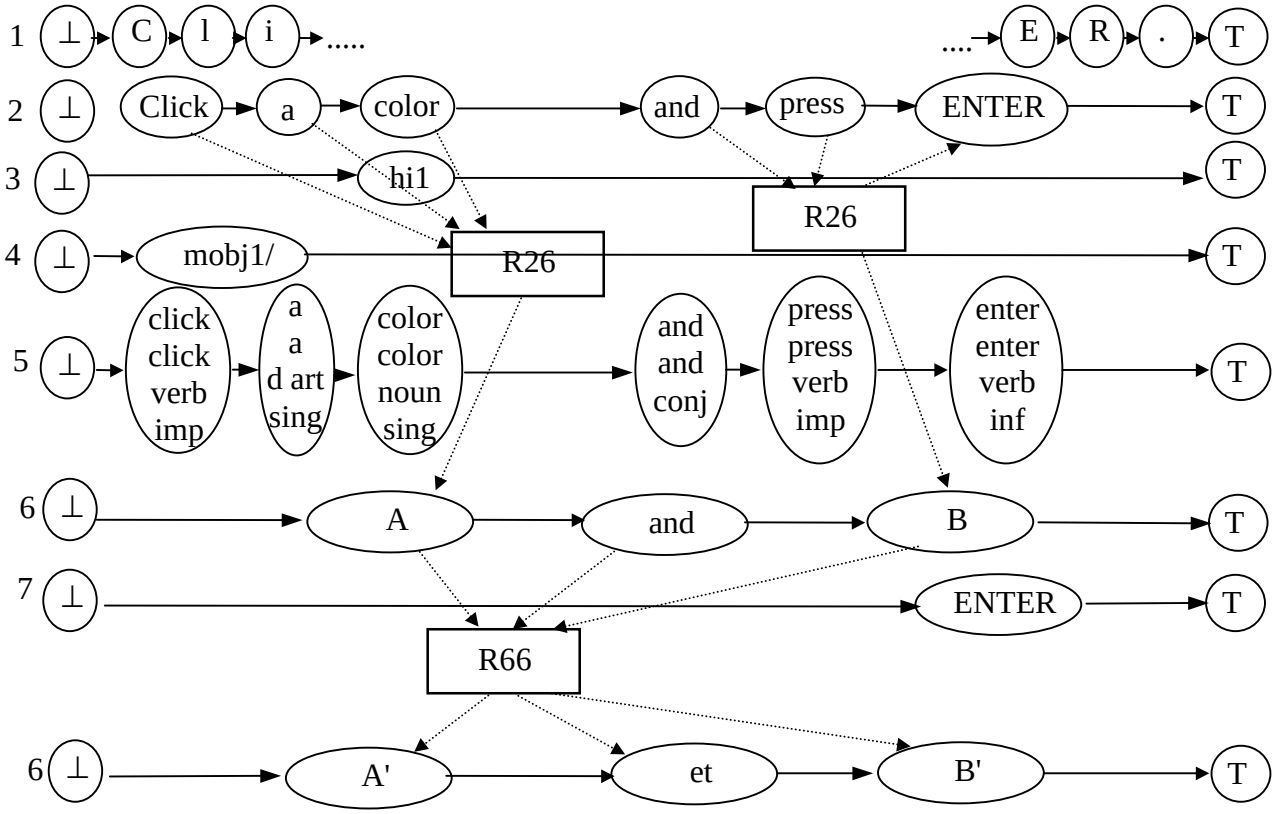


Figure 13 : Nœuds pivots sur le treillis $TELAM_{S1}$, et étage 6 du treillis $TELAM_{T1}$

Cela donne le segment T1 :

Cliquer sur une couleur, et appuyer sur ENTREE

Remarque :

Nous avons déjà vu au chapitre 5 que certaines expressions pivots inversent l'ordre des parties A et B d'une langue à l'autre, comme c'est souvent le cas de l'anglais vers le japonais ou vers le chinois. Il suffit dans ce cas de suivre les correspondances indiquées dans la liste des expressions pivots utilisées.

3.3 Génération fondée sur les expressions à cases

Les nombres apparaissant dans les textes surtout techniques sont en général traduits de façon systématique d'une langue à l'autre. "Systématique" ici ne signifie pas qu'ils sont recopiés tels quels, mais qu'ils suivent des règles simples. Connaissant ces règles, et en supposant que seuls les nombres changent d'un segment S2 de la mémoire à un segment S1 à traduire, il est alors aisé de générer la traduction de T1. Nous prendrons l'exemple des variables des fichiers ressources généralement notées "% n" où n est un nombre.

On peut considérer que "n" est une pseudo catégorie grammaticale "NBR". On génère alors une suite hétérogène composée de mots et de cette catégorie grammaticale qui forme une "expression à cases" comme :

S1 : Press on %n → S2 : Appuyez sur %n

Il est alors aisé de trouver la traduction pour d'autres valeurs de n.

Cela est encore vrai pour les chapitres ou les dates, moyennant des règles de passage d'une langue à l'autre, spécifiques à un couple de langues donné.

S1 : mercredi %n mai → S2 : mièrcoles a %n de mayo

4. Vers un système de Traduction Fondée sur la Mémoire

Introduction

Ayant posé les bases d'une méthodologie de traduction fondée sur la mémoire, nous ouvrons brièvement ici quelques perspectives vers les Outils de Traduction Fondée sur la Mémoire de seconde génération.

Pour cela, nous rappelons d'abord la méthodologie générale du système. Les étapes principales y sont décrites. Puis nous montrons quelles sont les possibilités de traitements offerts par ces nouveaux systèmes : celles qu'offraient les outils de première génération, et celles inhérentes à la nouvelle méthodologie.

Enfin nous donnons quelques lignes directrices pour l'implémentation d'un outil complet, concernant l'architecture globale, la gestion documentaire, et quelques points importants d'ergonomie.

4.1 Le système dans son ensemble

Le système dans son ensemble doit permettre, à partir d'une base de données (la "mémoire" de bi-segments) au format XML, de proposer un ou plusieurs segments traductions du segment entré.

4.1.1 Méthodologie générale

Préparation de la mémoire : fichier index

Avant l'utilisation de la mémoire, celle-ci doit avoir été préparée. Les étapes principales de la préparation d'une telle mémoire sont les suivantes :

- collecte des textes et cibles correspondants
- conversion de ces textes au format XML
- alignement de ces textes
- balisage ou lemmatisation de ces textes
- création de l'index

On notera qu'outre le traitement de nouveaux textes, il existe déjà beaucoup de mémoires de traduction constituées avec les outils de première génération. La reprise de ces mémoires est tout à fait réalisable dans ce cadre. De même l'utilisation d'aligneurs proposés par les OTFM1g rentre dans le cadre de cette procédure.

Il est nécessaire pour mener à bien ces opérations de développer des filtres depuis les formats d'enregistrement des documents que l'on veut créer, vers XML, et vice versa. La tendance actuelle étant à un rapprochement vers SGML, et notre raisonnement étant basé sur des documents au format XML, cela devrait être assez facile.

Les problèmes classiques de gestion de bases de données se posent pour la mémoire des bi-segments. Cette mémoire peut en effet devenir très grande. Outre les techniques inhérentes aux grandes bases de données (index performants, compression des données) il est important de donner des moyens de gestion de projets, en classifiant par exemple les entrées de la mémoire par date de création et modifications, par domaine, client, type de documents, etc.

Construction de la structure $TELAM_{S1}$ du segment source S1

Un nouveau segment étant entré, la première opération consiste à créer la structure source $TELAM_{S1}$ associée au segment S1. Les techniques de création et de manipulation de TELAM ont été expliquées au chapitre 7.

Recherche d'unités de mémoire candidates, et sélection

La structure $TELAM_{S1}$ étant disponible, les différents chaînes correspondant aux chemins les plus récents des étages de $TELAM_{S1}$ servent de base à la recherche d'unités de mémoire adéquates, par comparaison de ces chaînes avec les parties sources de ces unités de mémoire. La recherche se fait essentiellement en deux temps :

- sélection basée sur l'utilisation de l'index de la mémoire
- sélection plus précise par comparaison multidimensionnelle basée sur la distance d'édition

Le résultat de cette étape est l'obtention soit de l'ensemble vide, soit de l'unité de traduction (S2, T2) la plus proche possible de S1.

Construction des structures TELAM de S2 et T2
Pour pouvoir effectuer les traitements suivants, les structures $TELAM_{S2}$ et $TELAM_{T2}$ de S2 et T2 sont construites.

Construction du squelette de la structure $TELAM_{T1}$ cible
La structure cible $TELAM_{T1}$ est construite par analogie avec (S2, T2). Cela repose sur les étapes suivantes :

- recherche de similarité entre S1 et S2 en termes d'opérations d'édition
- recherche de correspondance entre S2 et T2
- transfert et génération de $TELAM_{T1}$ analogie

Le résultat de cette étape est donc la structure $TELAM_{T1}$ comportant les informations linguistiques et de mise en forme nécessaires à la création du segment T1.

Génération du segment cible T1
Il s'agit du passage inverse de $TELAM_{T1}$ vers le segment T1. Le segment XML est construit par report des informations portées par $TELAM_{T1}$. Le placement exact des différentes données est effectué en particulier grâce aux échelles linéaires qui indiquent la place relative des différents éléments.

4.1.2 Robustesse de la méthode

Il est important d'insister sur le fait que la recherche de similarité en particulier, est basée sur une approche en cascade. Cela permet de remédier à l'absence ou l'incomplétude d'un des modules d'analyse. Les ressources linguistiques n'étant pas toujours disponibles pour tous les couples de langues traitées, cette approche permet de traiter au mieux la traduction, avec les ressources disponibles.

4.2 Traitements rendus possibles

4.2.1 Situations de traduction

Les OTFM1g actuels permettent de traiter globalement les situations suivantes :

- traduction exacte (S1 = S2), encore appelée "correspondance à 100%" : elle nécessitait que l'ensemble des éléments, du plus petit code d'index aux caractères de texte soient les mêmes entre S1 et S2..
- traduction linguistique exacte, mais format différent sur l'ensemble du segment. Dans ce cas les OTFM1g permettent généralement de récupérer la traduction avec le bon format sur l'ensemble du segment.

- traduction approchée. Dans ce cas, que ce soit par différence de mise en forme ou linguistique, la traduction est donnée comme "approchée".

L'étude précédente a montré qu'il est maintenant possible de traiter des situations de traduction nouvelles. Nous rappelons ici ce nouvel ensemble de possibilités

- traduction exacte : même situation que précédemment
- traduction exacte linguistiquement, mais mise en forme différente de celle de l'unité de traduction. Dans ce cas, notre méthode permet maintenant de transférer le format de S1 vers T1. Nous gagnons donc une part de silence par rapport aux OTFM1g qui classaient ce genre de segment en "traduction approchée".
- traduction très proche. Nous avons montré que nous sommes maintenant capables de générer les traductions de segments qui diffèrent d'un segment de la mémoire d'un mot, pourvu que la catégorie grammaticale soit conservée. Cela diminue nettement le silence car ce genre de situation est fréquent, surtout dans les documents industriels. Un cas particulier de traduction proche est donné par la gestion d'objets pseudo-linguistiques par les "expressions à cases" proposées au paragraphe précédent.
- composition de segments : nous avons montré comment, dans certains cas, il est possible de composer des segments de la mémoire avec une expression pivot. Là encore, le silence est réduit. Une extension de cette notion devrait permettre de proposer d'autres types de compositions.
- traductions approchées : cette catégorie est aussi proposée, mais elle a une signification différente. Nous avons maintenant en effet la possibilité de séparer clairement la recherche de similarité par type d'information. Ainsi, en annulant le cœfficient de la distance concernant les balises doubles et monobalises, il est possible d'obtenir des traductions approchés au seul sens linguistique.

4.2.2 Génération linguistique

Les techniques présentées dans les chapitres précédents permettent un traitement linguistique précis que nous rappelons maintenant.

Traitement des codages transcriptions
La gestion des codages de langues au niveau du segment XML, puis celle à la fois des codages et des transcriptions au niveau du premier étage des structures TELAM permet de traiter clairement ces problèmes de représentations des différentes langues. Cela autorise en particulier l'utilisation d'une transcription ou d'un codage différents de celui du document XML pour une adaptation efficace aux conventions des outils linguistiques utilisés par la suite.

Substitution de phrases entières
Comme dans les OTFM1g, la substitution de phrases exactement trouvées en mémoire est possible. Cela garantit de faire aussi bien que dans les OTFM1g.

Remplacement d'un mot par un autre
Nous avons vu que les techniques précédentes permettent maintenant de remplacer le mot d'une unité de traduction par un autre de même nature grammaticale. Cela nous permet de résoudre le problème suivant, cité en première partie (p 70) :

S2 : "Click on the right button" → T2 : "Cliquer sur le bouton droit" S1 : "Click on the left button" → T1 : "Cliquer sur le bouton gauche"

Changement de flexion
Pourvu qu'un générateur morphologique soit disponible, nous avons montré comment conserver les attributs morphologiques des mots dans les structures TELAM pour pouvoir les transférer. Cela permet de résoudre un autre problème vu en première partie (p 71) :

S2 : "Click on the right button" → T2 : "Cliquer sur le bouton droit" S1 : "Click on the right buttons" → T1 : "Cliquer sur les boutons droits"

Composition d'unités de mémoire
Il est maintenant possible de composer les mémoires. Ainsi les problèmes suivants soulevés aussi en première partie (p72) sont maintenant traités :

4.2.3 Transfert du format et des objets non linguistiques

Nos techniques permettent en outre de gérer correctement à la fois les mises en formes, et les objets non linguistiques, comme dans :

S2 : "Click on the right button" → T2 : "Cliquer sur le bouton droit" S1 : "Click {hyperlink}on the right button" → T1 : "Cliquer {hyperlink}sur le bouton droit"

S2 : "Click on the right button" → T2 : "Cliquer sur le bouton droit" S1 : "Click on the right button" → T1 : "Cliquer sur le bouton droit"

4.3 Vers un système complet

Les techniques précédentes ne peuvent être utiles qu'englobées dans un système. Nous présentons ci-dessous trois points qui nous paraissent importants pour la construction du système de traduction.

4.3.1 Architecture

Le système de traduction fondée sur l'exemple que nous proposons repose sur le stockage et la manipulation des données fondées sur une structure TELAM. Différents modules doivent pouvoir accéder à ces structures, pour lire ou pour modifier ces données. Or ces modules demandent souvent une transformation (transcodage, transcription, extraction ou filtrage de données). D'autre part, ils possèdent souvent des types de communication différents. Ils peuvent par exemple nécessiter une communication par réseau dans le cas où le module (de lemmatisation par exemple) est distant du système.

L'architecture de "Tableau Blanc" proposée par [Boitet 1988] permet ces fonctionnalités en proposant un "tableau blanc" central, qui porte alors dans notre système les structures TELAM. La communication des modules avec le tableau se fait via des "managers" qui gèrent les problèmes de transformations de structures, et l'ensemble des communications est géré par un coordinateur.

4.3.2 Gestion documentaire

Dans le travail précédent, nous n'avons manipulé que la notion de segment. Nous avons en effet clairement supposé que le texte à traduire pouvait être exprimé en unités de traductions sous forme XML. Nous avons considéré par convention que chaque unité était délimitée par les balises <s> et </s>. Cela pourrait correspondre à d'autres balises plus générales comme et , ou plus spécifiques comme <seg> ou </seg>, mais l'essentiel est qu'il y ait une unité, et que l'on puisse s'y référer.

Pour gérer ces unités au sein du document entier, ou d'un groupe de documents, il est nécessaire d'une part de posséder des méthodologie et des outils de segmentation des documents en unités de traduction, et d'autre part d'organiser et gérer l'ensemble des unités par rapport aux documents et aux groupes de documents[3].

D'autre part il est essentiel de posséder un module d'alignement de documents sources et cibles de façon à pouvoir construire automatiquement des mémoires de bi-segments à partir de documents existants. La génération de ces mémoires sera d'ailleurs facilitée par la segmentation systématique des documents sources et cibles selon des règles précises.

4.3.3 Ergonomie

La proposition des différents types de phrases trouvées dans le système doit permettre au traducteur de reconnaître clairement dans quelle situation se trouve la phrase (un des cinq cas précédents). Aussi est-il important de perpétuer la coutume lancée par les OTFM1g, et d'opter dans l'interface pour par exemple une couleur par type de phrases selon un dégradé par exemple, au moins pour quatre types de phrases :

- un style pour les phrases exactes (cas 1)
- un style pour les phrases reconstruites automatiquement (cas 2, 3, et 4)
- un style pour les phrases approchées (cas 5)
- un style pour les phrases pour lesquelles aucune unité de mémoire correspondante n'a été trouvée.

[3] Il est éventuellement nécessaire de gérer des notions de groupes de segments en paragraphes ou chapitres, etc.

Conclusion

Nous venons de présenter une méthodologie, fondée sur les structures TELAM et un certain nombre d'algorithmes, qui permettent non seulement de proposer les solutions que les OTFM1g offrent, mais aussi de générer de nouvelles solutions. Cela nous permet de réduire significativement le silence observé dans les mémoires de traductions traditionnelles, et des tests sont en cours pour confirmer notre approche.

Nos méthodes se basent sur une approche robuste, et à dégradation progressive suivant la pénurie d'outils. Le pire des résultat correspond à ce qu'offrent les OTFM1g actuels. D'autre part une attention particulière a été observée pour assurer la continuité avec les OTFM1g, et pour que les mémoire de bi-segments actuelles soient exploitables par ce nouveau système.

On notera enfin deux points d'importance. Premièrement, cette méthode et les données utilisées (les bi-segments) n'impliquent pas une directionnalité : notre raisonnement fonctionne dans les deux sens, pourvu que les mêmes modules linguistiques nécessaires soient disponibles dans un sens comme dans l'autre (dictionnaires, lemmatiseurs). D'autre part les structures TELAM pouvant être décrites totalement sous la forme de fichiers XML, nous pouvons maintenant envisager de garder et transmettre des mémoire ne contenant pas seulement des bi-segments, mais l'analyse complète de la structure.

Supposons maintenant que le rédacteur ou un premier traducteur du document travaille sur un poste capable d'analyser les phrases au fur et à mesure qu'il les écrit. Alors, moyennant un module de désambiguïsation intéractive, il devient possible de préciser par exemple que c'est tel terme de la base terminologique qui doit être employé, ou tel lemme parmi ceux proposés par le lemmatiseur. Cela permet d'envisager de garder en mémoire l'analyse du traducteur (au niveau désiré) de façon la propager et l'utiliser dans les traductions de ce document.

Conclusion et Perspectives

Conclusion

Rappel du problème

Notre étude concerne les Outils de Traduction Fondée sur la Mémoire qui font partie des outils pour la Traduction Humaine Assistée par la Machine tout comme les dictionnaires en ligne, les correcteurs orthographiques ou les bases terminologiques.

Les Outils de Traduction Fondée sur la Mémoire de première génération, bien qu'ils s'acquittent d'une partie de leur mandat, c'est à dire offrir au traduction des phrases stockées en mémoire correspondant exactement ou de façon approchée à celles qu'il faut traduire, font preuve à la fois de défauts, de manques souhaitables et en tout cas d'un éloignement étonnant des possibilités scientifiques actuelles.

Parmi les défauts, le plus important est de fournir des phrases approchées dans un ordre erroné (manque de précision), et plus rarement d'oublier des propositions (manque de rappel). Au nombre des manques souhaitables, on pense essentiellement au :

- non transfert de la mise en page
- non transfert des objets non linguistiques
- non génération
- non composition d'unité de traduction

Apports de la thèse

Eveil

Il nous semble que le premier apport consiste en un éveil de la communauté scientifique sur un domaine peu traité, mais posant des problèmes intéressants.

Nous avons ainsi montré que les erreurs et les manques présents dans les OTFM1g provenaient en grande partie du fait que leur représentation interne des données n'était pas adaptée aux traitements nécessaires, et en particulier linguistiques.

Proposition d'une structure de représentation à deux volets

Pour remédier à ce manque, nous en sommes venus à proposer une représentation en deux volets. Tout d'abord il nous a semblé important de garder un format primaire ou d'échange qui permettent le transport efficace et à moindre coût de toute donnée textuelle, dans quelque langue que ce soit. Nous avons choisi le format XML. Les documents à traiter sont donc filtrés vers XML, et un filtre retour leur redonne leur format d'origine après le traitement. Outre le fait que de plus en plus de documents sont exprimés dans un langage proche de SGML (et donc de XML, on peut penser au passage opéré en 1997 par Microsoft du codage de leur fichiers d'aide en HTML), nombre de filtres depuis et vers les formats classiques se développent, dans le projet OTELO [OTELO 1998] ou ILE [ILE 1997] par exemple.

Parce que nous avons vu que ce genre de structure hétérogène ne permet pas de mener de façon correcte les traitements nécessaires, nous avons proposé la structure TELA (Treillis Etagés et Liés pour le traitement Automatique) qui sépare les données présentes dans un segment XML en autant d'étages que de types de données. Parmi ces types nous voyons essentiellement :

- les caractères
- les mots
- les balises doubles
- les monobalises
- les lemmes et leurs attributs
- les expressions pivots

Cette liste n'est pas exhaustive, et l'on peut par exemple aussi penser à un étage pour les catégories grammaticales, les termes provenant d'une base terminologique, ou encore les concepts issus une ontologie.

Chaque étage est un treillis. Chaque nœud du treillis correspond à un objet à représenter : un caractère, un mot, un lemme par exemple. Chaque nœud porte une décoration qui code l'information à conserver (les caractères d'un mot, les attributs d'un lemme,..). Les treillis autorisent en particulier le codage d'ambiguïtés ou d'alternatives d'analyses qui constituent autant de chemins joignant la borne inférieure à la borne supérieure du treillis. La notion de séquentialité est conservée autant que faire se peut et permet de voir les nœuds d'un chemin comme autant de successeurs. La correspondance entre les nœuds de chemins différents d'un même treillis ou entre deux treillis est donnée par les échelles linéaires.

Quand il s'agit de relation complexes, comme par exemple la correspondance entre deux mots discontinus et un lemme, les liaisons permettent de les spécifier. L'ensemble des liaisons définie alors une relation de correspondance sur l'ensemble des treillis.

Nous donnons une formalisation mathématique complète, une première description externe sous forme d'une DTD XML pour l'échange de structures TELA et une deuxième sous forme d'un Langage à Objets pour la manipulation informatique des structures TELA.

Les structures TELA permettent enfin de pouvoir traiter les données par étage, sans s'encombrer de données superflues et gênantes comme dans les structures primaires. Cela peut être fait tout en conservant les liens entre les différentes données, grâce aux liaisons et aux échelles linéaires. On peut par exemple, comparer les segments au niveau de leurs caractères sans être gêné par le codage de mise en forme. La structure TELA fournit ainsi un support efficace pour les traitements en particulier linguistiques. Le remplissage de cette structure se fait soit par analyse de la structure du segment XML à représenter, soit par appel de modules externes, comme un lemmatiseur par exemple.

Clarification et formalisation de la notion de correspondance entre segments : la similitude

Nous avons ensuite précisé ce sur quoi se base la recherche de correspondance entre deux segments, en proposant la notion de similarité. Nous fondons cette notion sur celle de distance d'édition, telle que Wagner et Fischer l'ont définie. Elle permet de déterminer la distance entre deux segments vus comme une suite d'entités homogènes, en termes de nombre minimal d'opérations d'édition de base sur ces entités nécessaires pour passer d'un segment à l'autre. En considérant que les entités correspondent à des caractères, des mots, des balises, des lemmes, des catégories grammaticales, ou tout autre type de donnée représentée au niveau d'une structure TELA, on arrive alors à qualifier de façon précise et explicite la notion de distance entre deux segments.

Les algorithmes X et Y de Wagner et Fischer ont été légèrement modifiés (donnant X' et Y') de telle façon à pouvoir connaître exactement la nature des opérations d'édition. Maintenant, les algorithmes X' et Y' permettent à la fois de connaître la distance d'édition entre deux segments, pour une dimension donnée (un type de donnée), et donnent *une* suite d'opérations minimale faisant passer d'un segment à l'autre, nous permettant de localiser l'endroit où les deux segments sont différents. Dans le cas d'une seule opération, cette suite est unique, et si les segments ne possèdent pas de répétition, l'algorithme Y' donne exactement l'endroit où se situe la différence.

Nous en profitons alors pour formaliser mathématiquement à la fois la notion de texte en nous basant sur les travaux de Schreider, et la notion de similarité en nous basant sur ceux de Wagner et Fischer.

Nouveau paradigme de traduction fondée sur la mémoire

Ayant donné un structure de représentation des segments et la façon de les comparer, il est désormais possible d'effectuer de façon correcte les opérations que l'on souhaitait faire. En particulier d'une part la séparation par type de données, et d'autre part l'utilisation d'analyseurs linguistiques donnant par exemple l'étage des lemmes, permet d'effectuer une recherche de segments similaires avec une précision et un taux de réussite accrus.

D'autre part, il est maintenant possible d'accéder à des fonctionnalités qui n'existaient pas dans les OTAFM1g. Parmi ces ouvertures, nous proposons trois heuristiques de traduction :

- *le transfert de format et d'objets non linguistiques* : en s'appuyant sur les correspondances fournies par les liaisons et les échelles linéaires nous proposons un mécanisme pour transférer systématiquement la mise en forme et les objets non linguistiques (liens hypertexte, balises d'index, etc.) du segment source vers le segment cible, dans le cas où une unité de mémoire correspondante a été trouvée.
- *la génération faible* : c'est un pas vers la traduction fondée sur la mémoire. Ainsi, dans le cas où, au segment à traduire correspond un segment identique au niveau des catégories grammaticales (distance nulle), et au plus différent d'une entité (distance de 1) au niveau des lemmes ou des mots, nous autorisons le remplacement du mot différent dans l'unité de traduction par le mot correspondant du segment à traduire. Le mot cible correspondant est alors trouvé par recherche dans le dictionnaire, ou la base terminologique. Alors, soit la solution cible est proposée telle quelle, avec donc le lemme du mot remplacé, soit un générateur morphologique est appliqué pour rétablir les accords et les flexions correspondantes.
- *la composition d'unités de traduction* : La notion d'expression pivot est introduite pour ouvrir la possibilité de composer les unités de traduction. Une expression pivot consiste en un mot ou un groupe de mot invariants, séparant la phrase et deux

en langue source, et en langue cible. On peut par exemple penser à "in order to" et "afin de" pour le couple anglais-français, "sin embargo" et "pourtant" pour le couple espagnol-français, ou encore "kara" et "parce que" pour le couple japonais-français. Dans ce dernier cas, la première partie de la phrase japonaise correspond à la seconde partie de la phrase française, et notre formalisme permet aussi de le coder. L'idée consiste alors à rechercher en mémoire un segment correspondant à la première partie de la phrase séparée par l'expression pivot, faire de même avec la seconde partie, et si les joindre les traductions de ces segments par la traduction de l'expression pivot. Il devient en particulier faisable de traduire une seule moitié de phrase, mais de façon exacte, notion qui n'existe pas dans les OTAFM1g.

Il est ainsi devenu possible de générer de nouvelles traduction à partir des unités de traduction stockées en mémoire. Nous trouvons donc réponse aux désir exprimés en début de cette étude et aussi dans certains travaux comme par exemple [LANGE 1997]. Les procédures présentées sont robustes et générales, contrairement aux essais de la littérature antérieure comme, par exemple [MEUNIER 1993].

Complexité statique et dynamique des traitements

Un des avantages de cette approche est la complexité des traitements. C'est ainsi qu'il a été prouvé au chapitre 7 que les algorithmes de construction de la structure TELA évoluent de façon linéaire avec le nombre d'éléments du segment XML. Au niveau du poids relatif de la structure TELA, cela consiste globalement à multiplier par le nombre d'étages le poids de la représentation sous forme primaire. Pour donner un ordre d'idée, un calcul montre qu'il faut de 14 à 19 octets pour coder en structure TELA un segment XML de dix mots et dix balises, quelque soit la langue (codage Unicode).

De même la complexité dynamique des algorithmes X' et Y' sont respectivement en $O(n^2)$ et $O(n)$, où n est la longueur moyenne des segments comparés. Cela semble extrêmement raisonnable, surtout si l'on restreint le nombre de segments candidats par une utilisation d'un index sur la mémoire de traduction, comme cela se fait déjà dans les OTFM1g.

Prototypage

Un prototype en langage LISP a été développé sous Macintosh. Cela a permis de vérifier d'une part la faisabilité et l'efficacité des comparaisons envisagées sur un corpus de cinquante segments XML pour lesquels un dictionnaire comportant l'analyse en lemmes et catégories grammaticale a été réalisé ad hoc. Le prototype permet de créer dynamiquement tous les étages TELA du segment XML choisi (y compris la recherche d'information linguistique dans le dictionnaire), et de représenter cette structure. En outre, un segment ayant été modifié, ou un nouveau segment ayant été entré, le prototype permet de rechercher en mémoire les trois segments les plus proches selon la dimension choisie (caractère, mot, lemme), d'afficher la distance de Wagner et Fischer entre les deux segments, ainsi que leur représentation graphique.

Le prototype a permis en particulier de vérifier la rapidité des comparaisons, dans la mesure des données restreintes auquelles fait appel le prototype. C'est ainsi que l'analyse en structure TELA à cinq étages, l'affichage de cette structure, et la comparaison de deux segments XML de 20 mots et balises se fait en tout en 4 centièmes de secondes sur un Power Macintosh G3 !

Perspectives

Perspectives scientifiques

Vers une mémoire multilingue

Il est étonnant de constater que, même si par exemple le formalisme TMX de l'organisation LISA permet une telle représentation, dans la pratique, presque aucune mémoire de traduction multilingue n'est utilisée : seules les mémoires bilingues dominent. Nous sommes d'avis que les traitements pourraient être plus efficaces si une seule mémoire contenait plusieurs langues. On entrevoit en particulier que l'utilisation des liaisons de structures TELA devrait permettre par exemple un codage plus intéressant de la correspondance entre termes par exemple par la concrétisation de dictionnaires à d'acceptions [Sérasset 1994].

Des structures TELA pour une mémoire analytique linguistique

Il semble intéressant d'étudier comment les structures TELA permettraient de conserver une pré-analyse linguistique conduite au niveau de la source (désambiguïsation ou langage contrôlé) en association avec la mise en page. La description XML des structures TELA devrait permettre un échange aisé de ces analyses. On pourrait alors envisager une chaîne de traduction fondée sur cet échange : premièrement les rédacteurs désambiguisent la langue source, le document est échangé sous la forme XML de TELA, et c'est cette forme qui est traduite en cible par une application combinée de traduction basée sur la mémoire, et de traduction fondée sur les règles. Ces dernières règles devraient alors être en accord avec celles du module d'analyse de départ.

Vers des systèmes hybrides

La structuration des données et l'emploi d'algorithmes fondés sur une distance permet d'entrevoir une relation entre la *traduction fondée sur la mémoire* telle que nous l'avons décrite, et les paradigmes de *traduction fondée sur l'exemple* ou *traduction fondée sur l'analogie* des écoles japonaises [Nagao 1984], américaines [Brown 1996], ou anglaises [Collins & Cunningham 1996]. Ces dernières techniques se situeraient à une position

intermédiaire sur un axe conduisant de la *traduction fondée sur la mémoire* à la *traduction fondée sur les règles*.

Dans le cadre d'un post-doctorat de deux ans débutant à l'automne 1998, et qui serait effectué dans le laboratoire de Nippon Telephones and Telegraphs de Nara (équipe du Dr. Ooyama, ancienne équipe du Dr. Ikehara), l'auteur se voit proposé l'opportunité d'étudier ces liens, en particulier dans le but de créer des systèmes hybrides qui utilisent les trois paradigmes. Cette étude, encadrée par le Dr. Osamu Furuse, devrait entre autres conduire à l'examen des liens avec la "Constituant Boundary Parsing" [Furuse 1996].

Perspectives d'application

Prototypage

La réalisation en LISP d'un prototype implémentant les concepts avancés dans la thèse a été menée dans le cadre du stage de DEA de Christophe Chenon, encadré par l'auteur. Les premiers résultats ont été expliqués plus haut. Un prototypage plus avancé, mais en JAVA maintenant devrait nous permettre de démontrer la faisabilité industrielle des idées avancées dans cette étude.

Vers un atelier de traduction

Ce deuxième prototype, s'il confirme nos attentes, ouvrira alors la porte à un système plus complet. Nous espérons ainsi pouvoir contribuer à l'amélioration des outils de traduction fondée sur la mémoire de seconde génération.

Evaluation des Systèmes de Traduction Fondée sur la Mémoire

Dans le cadre de la mise sur le marché d'un nouveau produit, le laboratoire de la société Xerox de Grenoble a proposé à l'auteur un stage d'évaluation des performances des outils de première génération, et une mise en perspective de celles de l'outil Xerox d'une part et des principes émis dans la thèse d'autre part. Ce stage s'effectuera de mai à août 1998. Il devrait en ressortir d'une part une évaluation plus linguistique des différents OTFM, et d'autre part une confirmation de l'adaptation de la distance de Wagner et Fischer à l'évaluation des systèmes de traduction fondée sur la mémoire, qui est un thème d'étude initié par [Thomson 1994].

Quelques commentaires conclusifs

Les Outils de Traduction Fondée sur la Mémoire de première génération sont en fait des *outils de traduction humaine assistée par la mémoire électronique*. Nous avons montré comment améliorer la recherche de segments en mémoire pour une meilleure précision et un meilleur rappel. Mais nous avons fait mieux : nous proposons des heuristiques permettant maintenant de générer des traductions. Le système devient ainsi un vrai *système de traduction automatique fondée sur la mémoire*, et l'on détourne ainsi ces systèmes vers la *traduction du réviseur* (voir p 27). Il devient capable non seulement de suggérer des traductions existant en mémoire, mais aussi d'adapter légèrement, ou de composer ces mémoires. Le pas vers la traduction automatique est franchi.

Le travail dans son ensemble fournit une étude pratique et théorique complète des Outils de Traduction Fondée sur la Mémoire. Il a le mérite d'exister là où le vide les plus consternant laissait le curieux sans réponse. Il se peut ainsi que ce travail devienne un premier ouvrage de référence pour un domaine en pleine évolution, et c'est en tous cas ce que nous lui souhaitons.

Enfin l'auteur espère avoir contribué par ce travail au rapprochement des communautés scientifiques et industrielles. Il espère en particulier que la première y verra une incitation à l'abord de problèmes concrets et l'évolution du scientifique vers les besoins de l'industrie. Cette attitude permet à la fois une justification pragmatique des études scientifiques et une jouissance des plus grandes devant l'apport technologique et scientifique. En outre il pense avoir montré que la considération de problèmes industriels n'empêche pas une possible formalisation des phénomènes, et un travail sur des problèmes de fond intéressants.

Quand à la communauté industrielle, elle peut prendre comme humble exemple ce travail pour se convaincre que le dialogue avec les scientifiques, et l'investissement (puisqu'il s'agit souvent d'un maître mot) dans de telles études, peut apporter des réponses à des questions qui ne se posent parfois même pas, mais qui n'en sont pas moins cruciales.

Reférences

AIT-KACI, H. (1985). A Lattice-Theoretic Approach to Compilation Based on a Calculus of Partially Ordered Type Structures, University of Pensylvania.

ANDO S., L., Y. (1997). Linguistic Structure Analysis by Analogy: Its Efficiency. NLPRS'97, incorporating SNLP'97, Phuket, Thailand.

BERGE, C. (1959). Espaces Topologiques - Fonctions Multivoques. Paris, Dunod.

BIRKHOFF, G. (1948). Lattice Theory. Providence, Rhode Island, American Mathematical Society.

BLANCHON, H. (1994). Perspectives of DBMT for monolingual authors on the basis of LIDIA-1, an implemented mock-up. COLING-94, Kyoto, Japan.

BOITET, C. and Y. Zaharin (1988). Representation trees and string-tree correspondences. Coling-88, 22–27 August 1988.

BOITET, C. (1988). Representation and computation of units of translation for Machine Interpretation of spoken texts, GETA - ATR.

BOITET, C. (1993). La TAO comme technology scientifique : le cas de la TA fondée sur le dialogue. III-ièmes journées Scientifiques "Traductique-TA-TAO", réseau LTT de l'UREF et de l'Université de Montréal.

BOITET, C. (1994). Machine-Aided Human Translation. International Conference on Linguistic Applications (ICLA-94), Penang, Malaysia, University Sainz Malaysia.

BOITET, C. (1994). Présentation de la table ronde sur la TAO - Journées sur le TALN du PRC-CHM. Traitement Automatique du Langage Naturel (TALN-94), Marseille.

BOITET, C. (1994). Lattices rather than Q-Graphs or Charts ?, GETA.

BOITET, C. (1995). Factors for success (and failure) in Machine Translation - some lessons of the first 50 years of R&D-. MT-Summit V, Luxemburg.

BOSAK (1997). XML, Java, and the future of the WEB.
http://sunsite.unc.edu/pub/suninfo/standards/xml/why/xmlapps.ps.zip.

BOUILLON, P., CLAS, André (1993). La Traductique, Les presses de l'Université de Montréal.

BRAY, T., PAOLI, Jean, Sperberg-McQueen, C. M. (1997). XML Specifications. http://www.w3.org/TR/WD-xml-lang.

BRAY, T. And al. (1997). XML - Principles, Tools, and Techniques, Ed. Dan Connelly, O'REILLY.

BROWN, P., COCKE, J., DELLA PIETRA S., DELLA PIETRA V., JELINEK, F., MERCER, R., ROOSSIN (1988). A statistical approach to language translation. COLING'88, Budapest.

BROWN, P. F., D P.V., MERCER R.L., DELLA PIETRA V.J., LAI J.C. (1992). Class-Based n-grams Models of Natural Language. Computational Linguistics **18**: 467-479.

BROWN, P. F., DELLA PICTRA S. A., DELLA PICTRA V. J., MERCER R. L. (1993). The Mathematics of Statistical Machine Translation: Parametric Estimation. Computational Linguistics. **19:** 263-311.

BROWN, R., FREDERKING, Robert (1995). Applying Statistical English Language Modeling to Symbolic Machine Translation. TMI 95.

BROWN, R. D. (1996). Example-Based Machine Translation in the Pangloss System. Coling, Copenhagen, Danmark.

BURNARD, L. (1991). TEI EDW26 - An Introduction to the Text Encoding Initiative. PreCOLING-92, Nantes, France.

BURNARD, L., Sperberg-McQueen, C. M. (1996). TEI Lite, la TEI simplifiée : une introduction au codage des fichiers électroniques, en vue de leur échange. http://www.univ-rennes.fr/pub/GUTenberg/, http://www-tei.uic.edu/orgs/intos/teiu5.tei/**:** Traduction de François Rôle, Ministètère de l'Education Nationale.

CANDIDO, M.-H. (1996). A principle-based hierarchical representation of LTAGs. COLING'96, Copenhagen.

CHEN, K.-H., CHEN Hsin-Hsi (1995). Machine Translation: An Integrated Approach. TMI'95.

CLARK, J. (1997). Jame's DSSSL Engine - Jade. http://www.jclark.com/dsssl/.

COLLINS, B., Cunningham, Pàdraig (1996). Translating Software Documentation By Example : An EBMT Approach to Machine Translation. COLING'96.

CRANIAS, L., PAPAGEORGIOU, Harris, PIPERITIS, Stelios (1997). “Example retrieval from a translation memory.” Natural Language Engineering **3**(4): 255-277.

DOI, S., MURAKI Kazunori (1992). Translation Ambiguity Resolution Based on Text Corpora Source and Target Languages. COLING'92.

DOI, S., MURAKI Kazunori (1993). Evaluation of DMAX Criteria for selecting Equivalent Translation based on Dual Corpora Statistics. TMI'93.

FURUSE, O., IIDA, H. (1992). Cooperation betwen transfer and Analysis in Example-based Framework. COLING'92.

FURUSE, O., IIDA, H. (1994). Constituent Boundary Parsing for Example-based Machine Translation. Coling'94, '94.

FURUSE, I. (1996). Incremental Translation Utilizing Constituant Boundary Patterns. COLING'96.

GALE, W. A., CHURCH Kenneth W.Subject (1993). A Program for Aligning Sentences in Bilingual Corpora. Computational Linguistics **19**(1): 75-102.

GOLDFARB, C. (1990). The SGML Handbook. Oxford, Clarenton Press.

GRISHMAN, R., KODAKA, M. (1992). Combining Rationalist and Empiricist Approaches to Machine Translation. TMI-92.

HAI, M. L., KAWTRAKUL A. & POOVORAWAN, Y. (1997). Phrasal Transfert Model for Vietnamese-English Machine Translation. NLPRS'97, Incorporating SNLP'97, Phuket, Thailand.

HEYN, M. (1992). Present and Future Needs in the CAT-World. Bruxelles, Trados Benelux S.A.

HUTCHINS, W. J., Harold SOMERS (1992). An Introduction to Machine Translation, Academic Pr.

IDE, N. c. p. (1996). EAGLES Text Corpora Working Group Workshop. Madrid, Espagne, Eagles, DG XIII, Comunauté Européenne.

IDE, N., VERONIS, Jean (1993). Background and context for the development of a Corpus Encoding Standard. Aix-en-Provence, Laboratoire Parole et Langage, Université de Provence EAGLES - Corpus sub-group on Text Representation.

IDE, N., VERONIS, J. (Eds.) (1995). The Text Encoding Initiative: background and context. Kluwer Academic Publishers, Dordrecht, 342p [reprinted from Triple special issue of Computers and the Humanities, 29, no 1/2/3, with an original bibliography].

IIDA, H., SUMITA Eiichiro and FURUSE Osamu (1995). Spoken-Language Translation Method Using Examples. COLING 96.

ISHIDA, R. (1998). Barriers to Global Design & Development. Meylan, Grenoble, France, Xerox.

JACOBSON, I. (1992). Object-Oriented Software Engineering - A case Driven Approach. Wokingham, England, Addison-Wesley.

JAJA, J. (1992). An Introduction to Parallel Algorithms. Reading, Masschussets, Addison-Wesley Publishing Companay.

JOLOBOFF, V. (1989). Document representation. Structure Documents. J. Andre, Furuta, R., Quint, V. Cambridge, INRIA. **1:** 220.

JONES, D. (1992). Non-Hybrid Example-based Machine Translation Architectures. TMI-92.

JOSHI, A. K. (1987). An introduction to Tree Adjoining Grammars. Mathematics of Language. Manaster-Ramer. Amsterdam, John Benjamins.

KAJI, H., KIDA, Y., MORIMOTO, Y. (1992). Learning Translation Templates from Bilingual Text. Coling'92.

KAJI, H., AIZONO, Toshiko (1996). Extracting Word Correspondances from Bilingual Corpora Based on Word Co-occurrence Information. Coling 96.

KANO, N. (1995). Developping International Software for Windows 95 and Windows NT.

KATAOKA, Y., KATAOKA, T., UEZONO, K., OHARA, H. (1997). The Essentials for Developping Multilingual Computer Environment. IJCAI'97, MULSAIC'97 Workshop, Nagoya, Japon, AAAI.

KATOH, N., Uratani, N. (1992). Extraction and Machine Translation of News Sentences with Fixed Patterns. 44th Annual Convention IPS Japan.

KAY, Martin. (1973). The MIND system. Courant Computer Science Symposium 8: Natural Language Processing. R. Rustin. New York, Algorithmics Press**:** 155-188.

KAY Martin (1980). Algorithm Schemata and Data Structures in Syntactic Processing. Palo Alto, XEROX Palo Alto Research Center.

KITANO (1993). A Comprehensive and Practical Model for Vietnamese-English Machine Translation. NLPRS'97, Incorporating SNLP'97, Phuket, Thailand.

KUGLER, M., Ahmad, K., Thurmair, Gr. (1992). The Translator's Workbench, Tools and Terminology for Translation and Text Processing in Europe - TWB Final Report, Communauté Européenne, LRE, DG XIII.

LAFOURCADE, M. (1993). LEAF, ou comment garder l'origine de l'ambiguité. Troisième Journées Scientifiques Traductique-TA-TAO, Montréal, Canada.

LAFOURCADE, M. (1994). Génie Logiciel pour le Génie Linguiciel. Laboratoire GETA-CLIPS. Grenoble, France, Université Joseph Fourier**:** 289.

LANGE, J.-M. (1994). Système d'aide à la traduction: un point de vue industriel. Traitement Automatique du Language Naturel en France aujourd'hui (TALN-94), Marseille, GDR-PRC Communication Homme-Machine, Pôle Langage Naturel.

LANGE J.-M., G. E., & DAILLE, B. (1997). Bricks and Skeletons: Some Ideas for the Near Future of MAHT. Machine Translation **12**.

LANGLAIS, P., Simard, M., Véronis, J., Armstrong, S., Bonhomme, P., Debili, F., Isabelle, P., Souissi, E., Théron, P. (1998). ARCADE: A cooperative Research Project on Parallel Alignment Evaluation. ELREC, Grenade, Espagne.

LAW, H.-C., CHAN Chorkin (1996). N-th Order Ergotic Multigram HMM for Modeling of Languages without Marked Word Boundaries. COLING'96, Copenhagen.

LEE, G., Jung, Hanmin, LEE, Jong-Hyeok (1995). Bi-directional memory-based dialog translation: The KEMDT approach. PACLING II 1995.

LEPAGE, Y., & ANDO, S. (1996). Un éditeur pour la construction de banques d'arbres. TALN'96, Marseille, France.

LEPAGE, Y. (1997). Corpus Contraction by Sentence Extraction using Analogy. NLPRS'97, incorporating SNLP'97, Phuket, Thailand.

LEPAGE, Y. (1997). Recherche tolérante de motifs de chaînes.

LEPAGE, Y., IIDA, H. (1998). Résolution d'analogie indépendamment de la langue, avec détection d'échec. Quatrième conférence annuelle de l'association pour le traitement automatique des langues (Gengo Shori Gakkai), Fukuoka, Japon.

LEPAGE, Y. (1999). Représentativité de textes par analogie. TALN'98, Paris, France.

LEVIN L. (1995). Using Context in Machine Translation of Spoken language. TMI-95.

LI, J.-J., & CHOI, K.-S. (1997). Corpus-Based Chinese-Korean Abstracting Translation Systems. IJCAI'97, Nagoya, Japon.

MARCOU, Yves (1994) INTRO. http://tornade.ere.umontreal.ca/~marcoux/INTRO/1.htm

MARGERMAN, D. M., MARCUS, M. P. (1990). Parsing a Natural language Using Mutual Information Statistics. Proceedings of AAAI 90.

MARSHALL, B. (1996). An Introduction to SGML. USA, Softquad Inc.

MASINI, G., NAPOLI, A., COLNET, D., LEONARD D., TOMBRE, K. (1991). Les langages à objets. Paris, InterEditions.

MATSUO, Y., SHIRAI Satoshi, YOKOO Akio, IKEHARA Satoru (1994). Direct Parse Tree Translation in Cooperation with Transfer Method. NeMLaP.

McLEAN (1992). Example-Based machine Translation using Connectionnist Matching. TMI-92.

McROY, S. W. (1992). Using Multiple Knowledge Sources for Word Sense Disambiguation. Coling'92.

MELBY, K. (1997). Data Exchange Standards from the OSCAR and MARTIF Projects. MT Summit VI, San Diego.

MEUNIER , F. (1993). Découpage de phrases et alignement de sous-phrases dans un corpus bilingue. UFR Informatique. Paris, Université Paris 7**:** 56.

MILLER, G., Beckwith R., Fellbaum Ch., Gross, Derek, Miller, Catherine (1993). An introduction to WordNet : An Online Lexical Database. http://www.cogsci.princeton.edu/pub/wordnet/5papers.ps.

MIMA, H., FURUSE Osamu, IIDA HitoshiSubject (1997). Improving Performance of Transfer-Driven Machine Translation with Extra-Linguistic Information from Context, Situation and Environment. IJCAI'97, Nagoya, Japan.

MITAMURA, T., NYBERG, Eric H., CARBONELL, Jaime G. (1993). Automated Corpus Analysis and Acquisition of Large, Multi-Lingual Knowledge Bases for MT. TMI 93.

MOYNE, J. A. (1985). Understanding Language, Man or Machine. New York, Plenum Press.

MURAKI Kazunori, Y. K., KAMEI Shin-ichiro, DOI Shinichi (1996). Transparency Encourages Users Getting into Systems- Interactive Disambiguation in TWP -.

MURAYAMA, T., WATANEBE, H., OGINO, S. (1990). An Interactive Japanese Parser for Machine Translation. Coling'90, Helsinky.

MURAYAMA, H. (1993). Pattern-Based Translation: Context Free Transducer and Its applications to Practical NLP. Natural Language Pacific Rim Symposium (NLPRS'93).

NAGAO, M. (1984). A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. Artificial and Human Intelligence. E. A. a. B. R., NATO Publications.

NAGAO, M. (1990). Dependancy Analyser: A Knowledge-Based Approach to Structural Disambiguation. COLING'90.

NAGAO, M. (1990). Toward Memory-Based Translation. COLING'90.

NAGAO, M. (1992). Some Rationales and Methodologies for Example-based Approach. International Workshop on Fundamental Research for the Future Generation of Natural Language Processing (FGNLP), Manchester.

NIREMBURG Sergei, D. C., GRANNES D.J. (1993). Two Approaches to Matching in Example-Based Machine Translation. TMI'93.

OGURA, K., HASHIMOTO Kazuo, MORIMOTO Tsuyoshi (1989). Object-Oriented Interface for a Linguistic Database. Working Conference on Data and Knowledge Base Intergration, University of Keele.

PEREIRA, F. C. N., Stuart M. SHIEBERSubject (1987). Prolog and Natural Language Analysis. Menlo Park, CA 94025, CSLI/SRI International.

PIERREL, J.-M. (1987). Dialogue oral homme-machine: connaissances linguistiques strategies et architectures des systemes. Paris, Hermes.

PLANAS (1997). Towards an Evolution of Memory Based Translation Systems. PACLING, Ohme, Tokyo, Japan.

QUINT, V. (1989). Systems for the manipulation of structured documents. Structure Documents. J. Andre, Furuta, R., Quint, V. Cambridge, INRIA. **1:** 220.

RUTHERFORD, D. E. (1965). Introduction to Lattice Theory. Edinburg, Oliver & Byd Ltd.

SADLER, V. (1989). The Bilingual Knowledge Bank.

SADLER, V. (1989). Translating with a simulated Bilingual Knowledge Bank, BSO.

SADLER, V. (1989). Working with Analogical Semantics: Disambiguation Techniques in DLT. Dordrecht/Providence, FORIS.

SADLER, V. (1990). Pilot Implementation of a Bilingual Knowledge Bank. COLING'90: 449-451.

SATO (1989). Memory-based Translation. IPSJ-WG.

SATO, S., NAGAO Makoto (1990). Toward Memory-based Translation. COLING'90.

SATO, S. (1991). Example-Based Translation Approach. International Workshop on Fundamental Research for the Future Generation of Natural Language Processing (FGNLP), ATR, Kyoto.

SATO (1991). Example-Based Machine Translation. Kyoto University.

SAVOUREL, Y. (1997). OpenTag. http://www.opentag.org/.

SCHREIDER, A. J. (1975). Equality, Resemblance, and Order. Moscou, Mir Publishers.

SCHWALL, U., & THURMAIR, G. (1997). From METAL to T1: Systems and Components for Machine Translation Applications. MT Summit VI, San Diego, AMTA.

SELIGMAN, M., BOITET Christian (1993). A "Whiteboard" Architecture for Automatic Speech Translation. International Symposium on Spoken Dialogue, Waseda University, Tokyo.

SEMMAR, N., FLUHR, Ch., PLANAS, E. (1997). Methodologies for Multimedia Software Localization. IJCAI'97, MULSAIC'97 Workshop, Nagoya, Japon.

SERASSET, G. (1994). SUBLIM : un système universel de bases lexicales et NADIA : sa spécialisation aux bases lexicales interlingues par acceptions. Thèse de doctorat, Université Joseph Fourier, Grenoble.

SHABES, Y., ABEILLE, Anne, JOSHI, Aravind K. (1988). Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. Coling'88, Budapest.

SHABES, Y., WATERS R.C.Subject (1995). Tree Insersion Grammar: A cubic-Time Parsable Formalism that Lexicalizes Context-Free Grammars without changing the Trees Produced. Computational Linguistics. **21:** 479-513.

SHIEBER, M., SHABES Y.Subject (1990). Synchronous Tree-Adjoining Grammars. 13th International Conference on Computational Linguistics.

SHIRAI, S., BOND Francis, TAKAHASHI Yamato (1997). A Hybrid Rule and Example based Method for Machine Translation. NLPRS'97.

SOBASHIMA, Y., FURUSE Osamu, AKAMINE Susumu, KAWAI Jun, IIDA Hitoshi (1994). A Bidirectional, Transfer-Driven Machine Translation System for Spoken Dialogues. Coling'94, Kyoto.

SPERBERG-McQUEEN, C., M., BURNARD, Lou (Eds) (1994) Guidelines for Electronic Text Encoding and Interchange. (TEI P3), Association for Computers and the Humanities (ACH), Association for Computational Linguistics (ACL), Association for Literacy and Linguistic Computing (ALLC), Vol. 1 & 2, Chicago.

STANFILL, C., WALTZ, D. (1986). Toward Memory Based Reasoning. CACM **29**(12): 1213-1228.

STEPHEN, G. A. (1994). String Searching Algorithms. Singapore, World Scientific.

SUKEHIRO (1995). Aligning sentences in bilingual corpora using document format - MT system for revised documents. Osaka, OKI Electric Ind. Co. Ltd., Kansai Laboratory, Japon.

SUMITA, E., TSUTSUMI Y. (1988). A Translation Aid System sing Flexible Text Retrieval based on Syntax-Matching. Tokyo, IBM Tokyo Research Laboratory.

SUMITA, I. H. (1991). Experiments and Prospects of Example-Based Machine Translation. Procedinds of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley.

SUMITA, E., IIDA Hitoshi, (1992). Example-based Transfer of Japanese Adnominal Particles into English. IEICE Transaction on Information and Systems **E-75-D**(4): 585-594.

SUMITA, E., OI Kozo, FURUSE Osamu, IIDA Hitoshi, HIGUCHI Tetsuya, TAKAHASHI Naoto, KITANO Hiroaki (1993). Example-Based Machine Translation on Massively Parallel Processors. IJCAI'93.

SUMITA, E., IIDA, Hitoshi (1995). Heterogeneous Computing for Example-Based Translation of Spoken Language. TMI 95.

TAKAHASHI, Y., SHIRAI, S., BOND, F. (1997). A method of Automatically Aligning Japanese & English Newspaper Article. NLPRS'97, incorporating SNLP'97, Phuket, Thailande.

TAKEDA, K., URAMOTO, NASUKAWA, T., TSUTSUMI, T. (1992). Shalt 2- A symetric Machine Translation System with Conceptual Transfer. Coling'92, Nantes.

TAKEDA, K. (1996). Pattern-Based Context-Free Grammars for Machine Translation. Proceedings of the 34th Annual Meeting of the ACL, Santa Cruz, California.

TAKEDA, K. (1996). Pattern-Based Machine Translation. Coling, Copenhagen, Danmark, ICCL.

TAMURA, S., KAMEI S., DOI S., YAMABANA K. (1996). Collecting of Verbal Idiomatic Expressions and Development of a Large Dictionary for Japanese to English Machine Translation. 2nd Annual Convention of Association for Natural Language Processing.

THURMAIR, G. (1997). Exchange Interfaces for Translation Tools. MT Summit VI, San Diego, AMTA, pp 74-93

THOMPSON, H. S., BREW Chris (1994). Automatic Evaluation of Computer Generated Text: Final Report on the TextEval Project. Edinburg, Human Communication Research Centre, University of Edinburgh.

UCHIDA, H., K. SUGIYAMA (1980). A machine translation system from Japanese into English based on conceptual structure. Coling, Tokyo.

URAMOTO, D. (1991). Lexical and Structural Disambiguation Using an Example-Base. Proceedings of the 2nd Japan-Australia Joint Symposium on NLP.

URAMOTO, N. (1995). Automatic Learning of Knowledge for Example-Based Disambiguation of Attachement. TMI 95.

VERONIS, J. (1996). SGML : MULTEXT. Aix - en - Provence, Laboratoire Parole et langage, Université de Provence, EC, LRE.

VERONIS, J., Langlais, Philippe (1997). Arcade - Evaluation de systèmes d'alignements multilingues. Aix-en-Provence, Université de Provence.

WAGNER, A. R., FISHER, J. M. (1974). The String to String Correction Problem. Journal of the Association for Computing Machinery **21**(1): 168-173.

WINOGRAD, T. (1982). Language as a Cognitive Process. Reading, Massachussets.

WITCAM, T. (1988). DLT-An industrial R&D project for multilingual MT. 12th International Conference on Computational Linguistics (COLING'88), Budapest.

WOOD, W. A. (1970). Transition Networks Grammars for Natural Language Analysis." Computational Linguistics, D.G. BOBROW, Editor. **13**(10): 591-606.

YAMABANA, K., DOI, Shinichi, KAMEI, Shin-ichiro, SATOH, Kenji, TAMURA Shinko, ANDO, Shinichi (1995). Interactive Machine-Aided Translation Reconsidered -Interactive Disambiguation in TWP -. NLPRS 95.

YAMABANA, K., KAMEI Shin-ichiro, MURAKI Kazunori, DOI Shinichi, TAMURA Shinko, SATOH KenjiSubject (1997). A Hybrid Approach to Interactive Machine Translation - Integrating Rule-based, Corpus-based, and Example-based Method-. IJCAI'97, Nagoya, Japan.

ZAHARIN, Y. (1987). String-Tree Correspondance Grammar: a declarative grammar formalism for defining the correspondance between dtrings of terms and tree structures. 3rd

Conference of the European Chapter of the Association of Computational Linguistics, Copenhagen.

TMX specifications. Organisation LISA (1997). http://www.lisa.orgtmx/.

Projet MULTEXT. (1996)
http://www.lpl.univ-aix.fr/projects/multext/.

Projet EAGLES (1998)
http://www2.echo.lu/langeng/en/le3/eagles/eagles.html.

Projet OTELO (project LE-2703 de la communauté Européenne, LRE). (1998). http://www.otelo.lu/.

Projet WORDNET (1998). http://www.cogsci.princeton.edu/~wn/.

Annexes

Contenu

Annexe 1

Mémoires du test

1. Mémoires du test

1.1 Mémoire Workbench du test

<RTF Preamble>
<FontTable>
{¥fonttbl
{¥f1¥fmodern¥fcharset0¥fprq1 Courier New;}
{¥f2¥fswiss¥fcharset0¥fprq2 Arial;}}
<StyleSheet>
{¥stylesheet
{¥St ¥s0 {¥StN Normal}}
{¥St ¥cs1 {¥StB ¥v¥f1¥fs24¥sub¥cf12 }{¥StN tw4winMark}}
{¥St ¥cs2 {¥StB ¥cf4¥fs40¥f1 }{¥StN tw4winError}}
{¥St ¥cs3 {¥StB ¥f1¥cf11 }{¥StN tw4winPopup}}
{¥St ¥cs4 {¥StB ¥f1¥cf10 }{¥StN tw4winJump}}
{¥St ¥cs5 {¥StB ¥f1¥cf15 }{¥StN tw4winExternal}}
{¥St ¥cs6 {¥StB ¥f1¥cf6 }{¥StN tw4winInternal}}
{¥St ¥cs7 {¥StB ¥cf2 }{¥StN tw4winTerm}}}
</RTF Preamble>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Course Overview
<Seg L=FRE>Présentation générale
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Description
<Seg L=FRE>Description
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>This is a one-day, instructor-led course.
<Seg L=FRE>Ce cours est dispensé sur un jour et animé par un instructeur.
</TrU>
<TrU>

<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>This course teaches students how to support the various features of AAA.
<Seg L=FRE>Les stagiaires y apprendront comment prendre en charge les différentes fonctionnalités de AAA.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>This course does not cover publishing Web page content.
<Seg L=FRE>Ce cours n'aborde pas la publication du contenu des pages Web.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Audience
<Seg L=FRE>Profils des stagiaires
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>This course is intended for Site Managers (or Webmasters) who want to learn about the CCC products.
<Seg L=FRE>Ce cours s'adresse aux gestionnaires de site (ou administrateurs Web) qui souhaitent connaître les produits CCC.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>In addition, it is intended for System Engineers who want to prepare for the forthcoming DDD exam.

<Seg L=FRE>Il s'adresse aussi aux ingénieurs système qui veulent préparer le prochain examen du programme DDD.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>The key skills required by organizations considering a Web presence include the following:
<Seg L=FRE>Les sociétés exigent des gestionnaires de site Web potentiels les compétences suivantes :
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Technical ability.
<Seg L=FRE>Compétence technique.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Technical ability. Webmasters must understand UNIX or Microsoft Windows NT(r) operating system nuances as well as server and router complexities.
<Seg L=FRE>Compétence technique. Les gestionnaires de site doivent connaître les nuances des systèmes d'exploitation UNIX ou Microsoft(r) Windows NT(r) et maîtriser parfaitement les serveurs et les routeurs.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997
<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Understanding of the user interface.
<Seg L=FRE>Connaissance de l'interface utilisateur.
</TrU>
<TrU>
<CrD>17101997
<CrU>PLANAS
<ChD>17101997

<ChU>PLANAS
<UsD>17101997
<UsC>0
<Seg L=USE>Information structure expertise.
<Seg L=FRE>Connaissance approfondie de la structure de l’information.
</TrU>

1.2 Mémoire Translation Manager du test

<NTMMemoryDb>
<Description>
Test pour la thèse sur IIS
</Description>
0000000001
<Control>
000001_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>Course Overview
</Source>
<Target>Présentation générale
</Target>

0000000002
<Control>
000001_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>Description
</Source>
<Target>Description
</Target>

0000000003
<Control>
000001_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>This is a one-day, instructor-led course.
</Source>
<Target>Ce cours est dispensé sur un jour et animé par un instructeur.
</Target>

0000000004
<Control>
000002_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>This course teaches students how to support the various features of AAA.
</Source>
<Target>Les stagiaires y apprendront comment prendre en charge les différentes fonctionnalités de AAA.
</Target>

0000000005
<Control>
000003_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>

<Source>This course does not cover publishing Web page content.
</Source>
<Target>Ce cours n’aborde pas la publication du contenu des pages Web.
</Target>

0000000006
<Control>
000004_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>Audience
</Source>
<Target>Profils des stagiaires
</Target>

0000000007
<Control>
000004_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>This course is intended for Site Managers (or Webmasters) who want to learn about the CCC products.
</Source>
<Target>Ce cours s’adresse aux gestionnaires de site (ou administrateurs Web) qui souhaitent connaître les produits CCC.
</Target>

0000000008
<Control>
000005_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>In addition, it is intended for System Engineers who want to prepare for the forthcoming DDD exam.
</Source>
<Target>Il s'adresse aussi aux ingénieurs système qui veulent préparer le prochain examen du programme DDD.
</Target>

0000000009
<Control>
000006_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>The key skills required by organizations considering a Web presence include the following:
</Source>
<Target>Les sociétés exigent des gestionnaires de site Web potentiels les compétences suivantes :
</Target>

0000000010

<Control>
000006_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>Technical ability.
</Source>
<Target>Compétence technique.
</Target>

0000000011
<Control>
000007_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>Webmasters must understand UNIX or Microsoft Windows NT(r) operating system nuances as well as server and router complexities.
</Source>
<Target>Les gestionnaires de site doivent connaître les nuances des systèmes d'exploitation UNIX ou Microsoft(r) Windows NT(r) et maîtriser parfaitement les serveurs et les routeurs.
</Target>

0000000012
<Control>
000008_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>2. Understanding of the user interface.
</Source>
<Target>2. Connaissance de l'interface utilisateur.
</Target>

0000000013
<Control>
000009_0_0000000877204641_English(U.S.)_FRENCH(NATIONAL)__EQFASCII
_SOURCE~1.TXT
</Control>
<Source>3. Information structure expertise.
viii Error! </Source>
<Target>3. Connaissance approfondie de la structure de l'information.
</Target>

</NTMMemoryDb>

1.3 Mémoire Transit du test

1.3.1 Extrait du fichier de référence source (partie source de la mémoire)

```
_# filter v3.0b  RTF Text File
# RTF Release v1.0
# Created on: 07.01.98  21:24:49
#_1_

|FMT - Info Title#_2_
Training Template - 8 1/2 by 11,  A4_3_
|FMT - Info Author#_4_
Jim Semick_5_
|FMT - Info DocComm#_6_
Used to create Chapter, Appendix, and Part-Opening pages
forLetter and A4 training materials._7_
|FMT - Info Operator#_8_
Emmanuel Planas_9_
|FMT - Company#_10_
UNU/IAS_11_
|FMT - Header#_12_

|FMT - Autonum Def#_13_
._14_
|FMT - Autonum Def#_15_
._16_
|FMT - Autonum Def#_17_
._18_
|FMT - Autonum Def#_19_
)_20_
|FMT - Autonum Def#_21_
(_22_
|FMT - Autonum Def#_23_
)_24_
|FMT - Autonum Def#_25_
(_26_
|FMT - Autonum Def#_27_
)_28_
|FMT - Autonum Def#_29_
(_30_
|FMT - Autonum Def#_31_
)_32_
|FMT - Autonum Def#_33_
(_34_
|FMT - Autonum Def#_35_
)_36_
|FMT - Autonum Def#_37_
(_38_
|FMT - Autonum Def#_39_
)_40_
|FMT - Text#_41_
```

```
<{><¥fs24 >Course Overview_42_
|FMT - Text#_43_
Description_44_
|FMT - Text#_45_
This is a one-day, instructor-led course. _46_This course
teaches students how to support the various features of AAA.
_47_This course does not cover publishing Web page
content._48_
|FMT - Text#_49_
Audience_50_
|FMT - Text#_51_
This course is intended for Site Managers (or Webmasters) who
want to learn about the CCC products. _52_In addition, it is
intended for System Engineers who want to prepare for the
forthcoming DDD exam._53_
|FMT - Text#_54_
The key skills required by organizations considering a Web
presence include the following:_55_
|FMT - Text#_56_
<{><¥fs24 > 1.<¥tab >Technical ability. _57_Webmasters must
understand UNIX or Microsoft Windows
|FMT - Text#
<{><¥fs14 >_<}><{><¥fs24 > operating system nuances as well
as server and router complexities.
|FMT - Text#_58__59_
 2.<¥tab >Understanding of the user interface._60_
|FMT - Text#_61_
 3.<¥tab >Information structure<}><{><¥i¥fs24 > <}><{><¥fs24
>expertise<}><{><¥i¥fs24 >._62_<}>_63_
```

1.3.2 Extrait du fichier de référence cible (partie cible de la mémoire)

```
_# filter v3.0b  RTF Text File
# RTF Release v1.0
# Created on: 07.01.98  21:30:57
#_1_

|FMT - Info Title#_2_
Training Template - 8 1/2 by 11,  A4_3_
|FMT - Info Author#_4_
Jim Semick_5_
|FMT - Info DocComm#_6_
Used to create Chapter, Appendix, and Part-Opening pages
forLetter and A4 training materials._7_
|FMT - Info Operator#_8_
Emmanuel Planas_9_
|FMT - Company#_10_
UNU/IAS_11_
|FMT - Header#_12_

|FMT - Autonum Def#_13_
._14_
|FMT - Autonum Def#_15_
._16_
|FMT - Autonum Def#_17_
._18_
|FMT - Autonum Def#_19_
)_20_
|FMT - Autonum Def#_21_
(_22_
|FMT - Autonum Def#_23_
)_24_
|FMT - Autonum Def#_25_
(_26_
|FMT - Autonum Def#_27_
)_28_
|FMT - Autonum Def#_29_
(_30_
|FMT - Autonum Def#_31_
)_32_
|FMT - Autonum Def#_33_
(_34_
|FMT - Autonum Def#_35_
)_36_
|FMT - Autonum Def#_37_
(_38_
|FMT - Autonum Def#_39_
)_40_
|FMT - Text#_41_
<{><¥fs24¥lang1036 >Présentation générale_42_
|FMT - Text#_43_
```

```
Description_44_
|FMT - Text#_45_
Ce cours est dispensé sur un jour et animé par un instructeur.
_46_Les stagiaires y apprendront comment prendre en charge
les différentes fonctionnalités de AAA. _47_Ce cours n'aborde
pas la publication du contenu des pages Web._48_
|FMT - Text#_49_
Profils des stagiaires_50_
|FMT - Text#_51_
Ce cours s'adresse aux gestionnaires de site (ou
administrateurs Web) qui souhaitent connaître les produits
CCC. _52_Il s'adresse aussi aux ingénieurs système qui veulent
préparer le prochain examen du programme DDD._53_
|FMT - Text#_54_
Les sociétés exigent des gestionnaires de site Web potentiels
les compétences suivantes_55_
|FMT - Text#_56_
<{><¥fs24¥lang1036 > 1.<¥tab >Compétence technique. _57_Les
gestionnaires de site doivent connaître les nuances des
systèmes d'exploitation UNIX ou
Microsoft<}><{><¥fs14¥lang1036 >_<}><{><¥fs24¥lang1036 >
Windows NT<}><{><¥fs14¥lang1036 >_<}><{><¥fs24¥lang1036 > et
maîtriser parfaitement les serveurs et les routeurs._58_
|FMT - Text#_59_
 2.<¥tab >Connaissance de l'interface utilisateur._60_
|FMT - Text#_61_
 3.<¥tab >Connaissance approfondie de la structure de
l'information<}><{><¥i¥fs24¥lang1036 >._62_<}>_63_
```

Annexe 2

Caractéristiques détaillées de quatre outils de Traduction Fondée sur la Mémoire

2. Caractéristiques détaillées des OTFM1g testés

Remarque importante : Les caractéristiques décrites ci-dessous n'ont pas été toutes testées. Ce sont celles annoncées par les sociétés éditrices de ces logiciels. Il se peut que certaines fonctionnalités annoncées ne fonctionnent pas totalement. Typiquement le traitement des langues asiatiques est souvent un problème.

2.1 Matériel et Technologie

		TRADOS	EUROLANG	STAR	IBM
Serveur					
	Type		IBM 486 Windows NT, Windows 95		
	Vitesse (Mhz)		> 66		
	RAM (Mo)		32		
	Disque dur (Mo)		15		
	SGBD		SQL		
	Système d'Exploitation		Windows NT, Windows 95, Unix		
	Réseau		TCP/IP		
Client (Configuration minimale)					
	Type	IBM PC 486	IBM PC 486	IBM 486	IBM 386
	Vitesse (Mhz)	33	66	66	66
	RAM (Mo)	8	16	16	16
	Disque dur (Mo)	8	15		
	Système d'Exploitation	Windows 3.1, ou 95	Windows 95, Unix	Windows 95	Windows 3.11, 95, OS/2
	Réseau	Novell Netware ou Microsoft LAN Manager	LAN	LAN	•Warp LAN manager (for OS/2 and DOS/Windows) •Novell NetWare 4.x •Windows for Workgroups (Version 3.11) •Windows 95 •Windows NT (versions 3.5, 3.51 and 4.0) as a LAN server
	Interface avec Traitement de texte	Word 6, WordPerfect, Amipro	Word 7 ou FrameMaker	Word 6, AmiPro, WordPerfect	Non
Technologie					
	Base de	Relationnelle,	Relationnelle	Relationnelle,	Relationnelle,

	données	dédiée	(SQL)	dédiée	dédiée
	Configuration	Autonome	Client/Serveur	Autonome	Autonome
	Mise en réseau	Possible	Obligatoire	Possible	Possible

2.2 Aligneur

		TRADOS	EUROLANG	STAR	IBM
Existence		WinAlign*(3)	Aligner	Intégré	Intégré
Formats Traités					
	AmiPro			oui	oui (2)
	AMRI				oui
	ANSI				oui
	ASM				oui
	Code source C			oui	
	Context			oui	
	Bookmaster				oui
	HP-Tag				
	HTML	oui (help files)		oui	oui(V2)
	Interleaf	oui(1)	oui	oui (Windows et Sun Solaris)	oui
	ITP				oui
	FrameBuilder				oui
	FrameMaker	oui(2)	oui	oui	oui
	MRI				oui
	Page Maker	oui		oui	oui (6.5)
	Quark XPress			oui	oui (3.32)
	RTF	oui	oui	oui	oui
	Siemens S5			oui	
	Siemens CNC			oui	
	SGML	oui	oui	oui	table à faire
	Ventura	oui		oui	oui
	Word	oui (6 et 7)	oui (6 et7)	oui (2 et 6)	oui (2 et 6)
	WordPerfect			oui	oui (5.0, 6.x)
	Windows Help files	oui		oui	
	Windows resources			oui	oui
	OS/2 ressources				oui
	Xyvision			oui	
	CODES ASCII	437, 850, 852, 333, Unicode		437, 850, MAC	oui

(1) Nécessite un transducteur supplémentaire appelé STagger pour Interleaf
(2) Idem : STagger pour FrameMaker
(3) Permet d'aligner des langues européennes et asiatiques

2.3 Langues traitées

(Source : 1)(Cible : 2)(Source et Cible : 3) (*Système d'exploitation spécifique nécessaire pour certaines langues)

		TRADOS	EUROLANG	STAR	IBM
	Africaans			3	3
	Allemand	3	3	3	3
	Allemand Suisse				3
	Anglais UK	3	3	3	3
	Anglais US	3			3
	Arabe				2* (OS/2)
	Biélorusse			3	
	Bulgare			3	2
	Catalan	3			3
	Chinois mandarin	2		3*	3*
	Chinois simplifié	2		3*	3*
	Coréen	2		3*	3*
	Croate			3	2*
	Danois	3	2	3	3
	Espagnol	3	3	3	3
	Espagnol mexicain	3			
	Finnois	3	2	3	3
	Français CAN	3			3
	Français FR	3	3	3	3
	Gaélique (Irlande)	3			
	Grec	3		3	3*
	Hébreu				2* (OS/2)
	Hongrois	3		3	2*
	Indonésien			3	2
	Islandais				3
	Italien	3	3	3	3
	Letton			3	
	Japonais	2		3*	3*
	Néerlandais moderne	3	3	3	3
	Néerlandais "permissive"				2
	Néerlandais restreint				3
	Néerlandais préféré				3
	Norvégien (bokmal)	3	2	3	3
	Norvégien (nynorsk)				3
	Polonais	3		3	2*
	Portuguais (Brésil)	3		3	3
	Portuguais (Europe)	3	2	3	3
	Roumain			3	
	Russe	3		3	3*
	Serbe			3	
	Slovaque			3	

	Slovène			3	
	Suédois	3	2	3	3
	Tchèque	3		3	2*
	Thaï			3*	
	Turc			3	2*
	Ukrainien			3	
	Vietnamien			3	
	Welsh (Pays de Gales, GB)	3			

2.4 Traitement des formats

		TRADOS	EUROLANG	STAR	IBM
Traitement de texte					
	Type	Hébergé	Hébergé	Editeur spécial	Editeur spécial
	Traitements de textes	Word 6, WordPerfect, Amipro	Word 6 ou FrameMaker	Emulation Word, WordPerfect, WordStar	
Format Interne		propriétaire	ELDIF (SGML)	propriétaire	SGML
Formats traités					
	AmiPro			oui	oui (2.0)
	ANSI	oui	oui	oui	oui
	ASCII	oui	oui	oui	oui
	Assembly				oui
	BookMaster				oui
	Code source C			oui	
	Context			oui	
	Excel	oui	oui		
	FrameBuilder				oui
	FrameMaker	oui(1)	oui	oui	oui (4.0)
	HP-Tag				
	HTML	oui (fin 1997)		oui	oui
	Interleaf	oui (ASCII)(1)		oui (Windows et Sun Solaris)	oui
	IPF (Aide en ligne OS/2)				oui
	OS/2 Ressources				oui
	PageMaker	oui (fin 1997)		oui	oui (6.5)
	Quark XPress			oui	oui (3.32)
	RTF	oui	oui	oui	oui
	SGML	oui	oui	oui	
	Siemens S5			oui	
	Siemens CNC			oui	
	Ventura				oui
	Windows, ressources			oui	oui
	Windows, fichiers aide	oui		oui	
	Word	oui (6, 7)	(Intégration)	oui (2 et 6)	oui (2, 6)
	WordPerfect	oui	oui	oui	oui (5, 6)
	WordSTAR				
	Write	oui	oui		
	XyVision			oui	

(1) Nécessite un filtre supplémentaire appelé “STagger”

2.5 Fonctionnalités diverses

	TRADOS	EUROLANG	STAR	IBM
Décompte de mots	oui	oui	oui	oui
Aide en ligne	oui	oui	oui	oui
Pré-traduction	oui	oui	oui	oui
Pré-Analyse	oui			oui
Traduction interactive	oui	oui	oui	oui
Reconnaissance termes fléchis		oui	oui	oui
Reconnaissance d'abbréviations	oui	oui	oui	oui
Extraction de terminologie			oui	oui
Extraction de phraséologie redondante	oui		oui	oui
Création de la liste des mots nouveaux d'un document			oui	oui
Vérification de terminologie			oui	
Insertion automatique de terminologie			oui	
Correction orthographique			oui	oui
Gestion des mémoires	basique	évoluée	basique	basique
Gestion de la terminologie	évoluée	évoluée	évoluée	basique
Mise à jour des documents	oui		oui	
Correcteur orthographique intégré			oui	oui
Connexion à un module de traduction fondée sur les règles	oui (Transend)	oui (LanTmaster)	oui (Logos)	non
Macros commandes			oui	non

2.6 Caractéristiques des mémoires de traduction

		TRADOS	EUROLANG	STAR	IBM
Base maximale		non communiquée	Illimitée	illimitée (une mémoire est un texte source plus un texte cible)	non communiquée
Critères associés à une entrée					
	Date	oui	oui		oui
	Utilisateur	oui	oui		
	Fréquence d'utilisation	oui			
	Autres attributs	oui	oui		
	Champs texte	oui	oui		
	Langue		oui		oui
	Contexte		oui	oui	
	Fichier				oui
Propositions traductions					
	Exactes	oui	oui	oui	oui
	Fuzzy	oui	oui	oui	oui
	Pourcentage affiché	oui		non	non
Fusion des mémoires			oui	oui	oui
Edition			oui	segments en contexte	un segment à la fois
Gestion			oui	faible	faible

2.7 Caractéristiques des bases terminologiques

		TRADOS	EUROLANG	STAR	IBM
Formats d'Import/Export					
	Spécifique	oui	oui	oui	oui
	Ascii	oui		oui	
	ANSI	oui		oui	
	CATS	oui		oui	
	SGML		oui		
	SDF	oui	oui		oui
	Définissable	oui		oui	
Définition de Champs		laborieuse mais complète		4 champs	non
Basculement automatique de langues		oui		oui	
Nombre de champs par entrée		1000	22	28	
Nombre de caractères d'un champ		4096	255	16384 pour textes, 1024 pour attributs	
Type de l'entrée		par concept	par concept	par concept	par homonymes
Recherche de termes	normale	oui	oui	oui	
	par jokers	oui	oui	oui	oui
Critères associés à une entrée					
	Date	oui	oui	oui	oui
	Utilisateur	oui	oui	oui	oui
	Champs texte	oui	oui	oui	
	Champ Graphique	oui		oui	
	Langue	oui	oui	oui	oui
	Contexte	oui	oui		
	Tri	oui	oui	oui	auto
Conflits	traités		oui	oui	
Impression		oui	oui	oui	oui
Liens	hypertexte	oui	non	oui	non
Masques d'affichage		oui	oui	oui	

2.8 Utilisation de données terminologiques

		TRADOS	EUROLANG	STAR	IBM
Dénomination		Multiterm	Intégré	TermStar	intégré
Traitements de texte hôtes		Word, WordPerfect Amipro	Word, FrameMaker	Word, WordPerfect, Amipro	aucun
Actions depuis Terminal					
	Recherche	oui	oui	oui	oui
	Filtre d'affichage	oui	oui	oui	non
	Remplacement auto	oui	oui	oui	oui
	Impression	oui	oui	oui	non
Recherche de doublons			oui	oui	
Interrogation en réseau					
	Possibilité	oui	oui	oui	oui
	Droits adaptables	oui	oui	oui	non
Connexion Internet		oui	non	non	non
Interrogation de plusieurs bases physiques		oui		oui	oui
	Graphiques	oui	non	oui	non
Version Lite					
	Existence	oui	non	oui	n/a
	Nombre d'entrées	8192	n/a		

2.9 Connexion à un module de traduction automatique

		TRADOS	EUROLANG	STAR	IBM
		Transend	LanTmaster	Logos	LMT
Langues traitées					
	Anglais, Allemand	oui		oui	
	Anglais, Français	oui		oui	
	Anglais, Espagnol	oui		oui	
	Anglais, Italien	oui		oui	
	Anglais, Japonais			oui	
	Anglais, Portuguais	oui			
	Allemand, Anglais	oui		oui	
	Allemand, Français			oui	
	Allemand, Italien			oui	
	Français, Anglais	oui			
	Espagnol, Anglais	oui			
	Partage des dictionnaires	oui			

2.10 Comparatif des prix (en Francs)

Ces prix sont donnés aussi pour LANT Eurolang Optimizer qui n'a pas été testé dans cette étude.

	TRADOS (20/01/1998)		**EUROLANG (1997)**		**STAR (17/01/1998)**		**IBM (15/06/1998)**	
Serveur	absent		Serveur Optimizer	29400	absent		absent	
Poste TAO	Workbench		Poste Optimizer	9950	Transit		Translation Manager	
(1 Poste)		16500		39350		14228		11300
(2 Postes)		30200		49300		24187		20300
(5 Postes)		71400		94750		56912		55000
(10 Postes)		112400				113824		104900
Poste TAO version limitée	aucun		aucun		Transit Light	7200	TM Light	4000
Poste Terminologie	Multiterm	6200	inclus	16000	TermStar	3850	inclus	
Terminologie version limitée	Multiterm Lite	2800			TermStar View Station	971		
Terminologie version Internet						23267		
Aligneur	WinAlign	19000 (50% univ.)	Aligner		Inclus		inclus	
Configuration		35500		45450		14228		11300

complète (1 poste)								
Configuration complète (5 postes)		90400		140150		56912		55000
Filtre Interleaf	STagger Interleaf	16800				6528	inclus	
Filtre Interleaf SUN						13224		
Filtre PageMaker						3180		
FrameMaker	STagger FrameMaker	16800						
Bases Terminologie spécialisées	Informat. Aviation Pharma.	275 510 1050			PackTerm (général tech.)	970		
Traduction Automatique	Transend (couple 2 sens)	5075	LanTmaster	?	Logos + Transit interface for Logos	? + 16740	LMT	?
Correction orthographe					Lexique, par langue	328		
Formation	1 jour, 6 pers.	5200	2 jours, 4 pers.	9750	1 jour, 2 pers.	5022	1 jour in situ, 1 jour, IBM	7200 5970