Énumération exhaustive et détection spécifique des analogies : étude pour les modèles de langue et la traduction automatique

Julien Gosme

université de Caen Basse-Normandie

UFR de Sciences

École doctorale SIMEM

Thèse

présentée par

Julien Gosme

et soutenue

le 13 février 2012

en vue de l'obtention du

Doctorat de l'université de Caen Basse-Normandie

Spécialité « Informatique et applications »

Arrêté du 07 Août 2006

Énumération exhaustive et détection spécifique des analogies : étude pour les modèles de langue et la traduction automatique

Membres du jury

M. Hervé Blanchon	maître de conférences, HDR, université Pierre Mendès France	rapporteur
M. Kamel Smaïli	professeur, université de Nancy 2	rapporteur
Mme Nadine Lucas	chargée de recherche CNRS, HDR, GREYC	
M. Denis Maurel	professeur, université François Rabelais Tours	
M. François Yvon	professeur, université Paris-sud 11	
M. Yves Lepage	professeur, université de Caen Basse-Normandie et université de Waseda	directeur
M. Jacques Vergne	professeur, université de Caen Basse-Normandie	co-directeur

À la mémoire de mes grands-parents maternels.

REMERCIEMENTS

Je remercie Yves Lepage, mon directeur de thèse, pour ses avis éclairés sans lesquels mon travail serait resté dans un état embryonnaire. Je le remercie également pour ses nombreuses relectures. Je tiens à remercier particulièrement Jacques Vergne et Nadine Lucas à la fois pour leurs conseils pertinents et leurs relectures. Je remercie les membres du jury et tout particulièrement messieurs Hervé Blanchon et Kamel Smaïli pour leurs rapports détaillés. Enfin, je remercie les membres du laboratoire GREYC avec qui j'ai eu le plaisir de travailler au quotidien et qui se reconnaîtront.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Ce mémoire de thèse présente un ensemble de travaux inscrit dans le domaine de la traduction automatique et plus précisément dans une approche particulière : la traduction automatique par l'exemple. Dans cette approche, la source principale d'informations réside dans un corpus d'exemples de traduction. La traduction revient à appliquer un ensemble d'opérations sur des exemples bien choisis du corpus afin d'en déduire de nouvelles traductions adaptées à une phrase à traduire.

Dans le chapitre 1, nous choisissons d'étudier le système de traduction par l'exemple *Aleph* puisqu'il est un très bon représentant de cette approche. Nous mettons en évidence que ce système, comme tout système par l'exemple, est confronté au « problème de la sélection des exemples ». En effet, la bonne sélection des exemples de traduction conditionne la qualité des systèmes par l'exemple puisqu'il s'agit de la première étape de calcul. Dans la suite de ce chapitre, nous proposons une solution à ce problème en organisant manuellement un corpus d'exemples en sous-grammaires par l'exemple minimales. Une expérience permet de soutenir cette solution puisque les sous-grammaires minimales permettent d'améliorer l'efficacité du système *Aleph*. En remarquant que les sous-grammaires par l'exemple peuvent également s'exprimer sous la forme de patrons d'analogies, nous continuons notre travail en cherchant à étudier les patrons d'analogies extraits automatiquement d'un corpus d'exemples monolingue.

Dans le chapitre 2, nous présentons une méthode permettant d'énumérer toutes les analogies entre sous-chaînes d'un texte, c'est-à-dire les quadruplets (A, B, C, D) dont les éléments A, B, C et D sont des sous-chaînes d'un texte et vérifient une définition de l'analogie $A : B :: C : D$. Cette méthode permet de traiter tout type de séquence, en particulier les chaînes de caractères et les séquences de mots. Dans la suite de ce chapitre, nous utilisons cette méthode afin d'extraire les patrons d'analogies entre trigrammes de mots de corpus de phrases en plusieurs langues. Il suffit d'appliquer la méthode d'énumération aux corpus de phrases considérés comme séquences de mots et de limiter l'énumération aux analogies entre séquences de mots de longueur 3. L'extraction automatique de patrons d'analogies entre trigrammes de mots mène à un classement des patrons d'analogies selon leurs fréquences d'apparition.

Dans le chapitre 3 nous étudions la structure des trigrammes inconnus afin de déterminer la capacité de reconstruction des patrons d'analogies. À cette fin, nous nous plaçons dans le cadre de la traduction automatique où un premier corpus d'exemples est connu (corpus d'entraînement) et

un second corpus d'exemples est constitué des phrases à traduire (corpus de test). Le problème principal est de traduire des séquences inconnus (présentes dans le corpus de test mais absentes du corpus d'entraînement). Dans notre cas nous cherchons à reconstruire des trigrammes de mots inconnus à l'aide des patrons d'analogies identifiés et des trigrammes de mots connus. Un trigramme inconnu est reconstructible s'il est en relation d'analogie (respecte un patron d'analogies) avec trois trigrammes connus. Plusieurs expériences soutiennent l'hypothèse selon laquelle les trigrammes inconnus reconstructibles par analogie sont similaires en structure et en fréquence aux trigrammes hapax (observés une seule fois dans le corpus d'entraînement). Ces patrons d'analogies constituent donc une réponse au problème initial de la « sélection des exemples ».

Dans le chapitre 4, nous exploitons les résultats précédents afin de définir une technique de lissage pour les modèles de langue trigramme. Cette technique de lissage dérive des techniques de lissage additives et favorise les trigrammes inconnus reconstructibles au détriment des trigrammes inconnus non-reconstructibles. La technique de lissage par analogie est comparée aux techniques classiques sur les onze langues du corpus Europarl (transcription des débats au parlement européen). Les résultats sont en faveur de la technique de lissage par analogie pour toutes les langues à l'exception du finnois.

Finalement, dans le chapitre 5, nous concluons notre travail et nous donnons quelques perspectives de recherche.

1

ARRIÈRE-PLAN ET PROBLÈMES

La traduction automatique (T.A.) a été l'une des premières applications non numérique des ordinateurs dès le milieu des années 1950 (voir Weaver (1955)). À l'heure actuelle, les principales approches en traduction automatique sont la traduction par règles et la traduction fondée sur les données. Dans une approche de traduction par règles, les connaissances symboliques nécessaires à la traduction sont collectées et codées par des linguistes. Dans une approche de traduction fondée sur les données, les connaissances symboliques ou numériques nécessaires à la traduction sont déduites voire induites d'un corpus d'exemples de traduction, soit composé par des traducteurs, soit assemblé automatiquement.

Nous nous intéressons exclusivement à la traduction fondée sur les données. Nous allons montrer que le système de traduction *Aleph* est un système représentant parfaitement l'approche par l'exemple. À l'aide de ce système, nous allons mettre en évidence *le problème de la sélection des exemples*, problème que rencontrent tous les systèmes de traduction par l'exemple.

1.1 MODÈLES DE DESCRIPTION DE LA LANGUE EN TRADUCTION AUTOMATIQUE FONDÉE SUR LES DONNÉES

L'objectif de la traduction automatique fondée sur les données est de concevoir des systèmes de traduction automatique ne nécessitant pas de description exhaustive des phénomènes linguistiques. Au minimum, les systèmes de traduction fondée sur les données exploitent des corpus alignés afin d'en extraire des traductions généralisables utiles à la traduction d'autres textes.

Il existe deux paradigmes en traduction fondée sur les données : la traduction probabiliste et la traduction par l'exemple. La délimitation entre ces paradigmes n'est pas clairement établie dans la littérature. D'une part, les systèmes de traduction probabiliste sont clairement identifiables puisqu'ils dérivent tous des modèles IBM (Brown et coll., 1990). D'autre part, il est difficile de définir les systèmes de traductions par l'exemple puisque les nombreux systèmes de traduction se réclamant de ce paradigme n'ont parfois que peu de choses en commun.

Somers (1999) propose trois caractéristiques principales des systèmes de traduction fondée sur les données, et selon lui, les systèmes probabilistes sont des systèmes par l'exemple :

1. *l'utilisation d'un corpus aligné* ;
2. *l'utilisation d'exemples de traduction comme source principale de connaissances linguistiques* ;
3. *l'utilisation d'exemples au moment de la traduction.*

D'autre part, Turcato et Popowich (2003) proposent de définir les systèmes de traduction par l'exemple non pas en fonction de l'usage du corpus aligné mais en fonction du statut de celui-ci. Ils proposent, comme caractéristique principale des systèmes de traduction par l'exemple, le fait que le système utilise les exemples de traduction comme connaissances linguistiques implicites, c'est-à-dire que les exemples de traductions offrent des contextes non monotones nécessaires au moment de la traduction.

Hutchins (2005) résume les définitions de Somers (1999) et Turcato et Popowich (2003) et met en évidence les limites de la définition de Somers. Selon Hutchins, la définition d'un système de traduction par l'exemple selon l'utilisation d'un corpus aligné est trop permissive puisque la plupart des systèmes par règles utilisent des corpus alignés afin de circonscrire les phénomènes linguistiques pris en considération. L'utilisation d'exemples de traduction comme source principale de connaissances linguistiques est nécessaire mais insuffisante puisque des règles de transfert peuvent être extraites par généralisation puis utilisées par des systèmes de traduction

par règles. Les systèmes probabilistes et certains systèmes par l'exemple n'ont pas cette caractéristique puisqu'ils n'utilisent pas d'exemple au moment de la traduction.

À la lumière des différentes caractéristiques proposées, nous pouvons considérer que le système de traduction *Aleph* de Lepage et Denoual (2005b) est un excellent représentant des systèmes de traductions par l'exemple. En effet il possède les trois caractéristiques proposées par Somers et respecte la définition de Turcato et Popowich. C'est pourquoi, dans ce chapitre, nous choisissons d'étudier la nature des descriptions linguistiques du système par l'exemple *Aleph*.

1.2 UN MODÈLE DE TRADUCTION PAR L'EXEMPLE : LA TRADUCTION PAR ANALOGIE

L'analogie proportionnelle

Les analogies de sens ont été utilisées intensivement dans les tests SAT d'examen d'entrée aux lycées états-uniens afin de mesurer les connaissances des élèves : étant donné deux mots liés par une relation sémantique claire, les élèves doivent identifier un couple de mots liés par la même relation sémantique parmi une liste de 5 couples de mots. Par exemple, étant donné le couple *maçon* : *pierre*, les étudiants doivent trouver le couple de mots de même relation sémantique dans la liste :

(a) *enseignant* : *craie*

(b) *charpentier* : *bois*

(c) *soldat* : *fusil*

(d) *photographie* : *appareil photographique*

(e) *livre* : *mot*

Dans cet exemple, le couple de mots attendu est *charpentier* : *bois* et les deux couples de mots peuvent se lire : le *maçon* est à la *pierre* ce que le *charpentier* est au *bois*. Turney (2008) a montré comment une technique vectorielle basée sur des corpus peut résoudre ces problèmes sémantiques avec une performance comparable à l'homme.

En linguistique, historiquement, l'analogie a d'abord été utilisée en morphologie pour la conjugaison et la dérivation. L'analogie sert ainsi à expliquer les dérivations morphologiques telles que (angl.) *ectoskeleton* : *ectocardia* :: *exoskeleton* : *exocardia* où « ecto » et « exo » sont des préfixes attachés aux racines « skeleton » et « cardia ». Un autre exemple

en conjugaison est *vous deviendrez* : *ils deviendront* :: *vous permettrez* : *ils permettront* où la marque de la troisième personne du pluriel au futur simple « -ront » commute avec la marque de la deuxième personne du pluriel « -rez ». Langlais et coll. (2008) montrent comment utiliser l'analogie entre termes médicaux afin de produire de nouveaux mots et de nouvelles traductions.

Au sein d'un lexique, les analogies entre termes sont connues pour structurer le lexique. Par exemple, l'analogie :

$$\textit{connecteur} : \textit{connecter} :: \textit{éditeur} : \textit{éditer}$$

montre les connexions entre forme et sens au sein du couple de substantifs « connecteur » et « éditeur » et du couple de verbes « connecteur » et « éditer ». Claveau et L'Homme (2005a) montrent comment des structures de treillis reflétant la structure du lexique peuvent être dérivées à partir du calcul d'analogie entre surfaces de mots.

Une étude de Lepage (2004b) a montré que l'analogie structure également des ensembles de phrases. Les expériences ont mis en évidence que la quasi-totalité (96 %) des analogies entre phrases faisaient sens comme dans l'exemple (angl.) *Do you like music?* : *Do you go to concerts often?* :: *I like classical music.* : *I go to classical concerts often.*

Traduction par analogie

Parmi les systèmes de traduction par l'exemple, nous nous intéressons à une architecture particulière : le système de traduction par analogie *Aleph*. Cette architecture se distingue d'autres approches de traduction par l'exemple par le fait qu'aucune segmentation, combinaison ou réordonnement n'est effectué explicitement. La seule opération de résolution d'équation analogique permet de construire de nouvelles phrases à partir de trois phrases.

Le principe est le suivant : étant donné une phrases à traduire D,

1. choisir deux phrases A et B du corpus en langue source ;
2. résoudre l'équation analogique $A : B :: x : D$ pour obtenir un ensemble de solutions $\{C\}$;
3. pour toute solution C construite par analogie et apparaissant dans le corpus en langue source ; alors
4. transposer l'analogie en langue cible, ce qui donne : $\hat{A} : \hat{B} :: \hat{C} : y$;
5. si l'équation analogique n'a pas de solution, traduire C de manière récursive ;

6. sortir les solutions de l'analogie en langue cible qui sont des traductions possibles de D.

Le point critique de cet algorithme est le choix des phrases A et B puisque le reste du traitement est conditionné par ce choix. Dans la pratique, une heuristique est nécessaire pour classer les couples de phrases (A, B) afin de privilégier les phrases permettant d'obtenir des solutions à l'équation analogique de l'étape 2. L'heuristique la plus simple est exprimée en termes de distances d'édition, notée d, et d'inclusion. L'hypothèse est que les triplets (A, B, D) ayant de faibles distances entre B et D ($d(B, D)$) et une bonne inclusion de A dans B ($l(B) - d(A, B)$, avec l la longueur de B), ont de fortes chances de conduire à un ensemble de solutions non nul lors de la résolution de l'équation $A : B :: x : D$.

Malheureusement, cette heuristique ne *conduit* pas à suffisamment d'analogies en langue source. Si bien que dans la pratique, la grande majorité du temps de traitement est consacrée à la sélection des couples (A, B) et à la résolution d'équations analogiques infructueuses en langue source.

Plus récemment, d'autres variantes de cette heuristique, privilégiant l'inclusion de D dans B ou de D dans A ont été testées. Mais à notre connaissance, aucune heuristique ne résout véritablement ce problème. Dans le cadre général de la traduction par l'exemple ce problème peut être appelé : *le problème de la sélection des exemples*. Ce phénomène a été identifié à l'aide de l'implantation d'*Aleph* en Python décrite en annexe A.

En mettant de côté le problème de la sélection des exemples, le principe de la traduction par analogie est bien établi. Les traductions effectuées selon ce principe et données en exemple par Lepage et Denoual (2005b) attestent la qualité des traductions. De plus, l'absence de décomposition-recomposition des phrases par l'usage exclusif d'équations analogiques permet d'éviter la segmentation en unités sous-phrastiques et l'alignement des unités sous-phrastiques discontinues.

De cette brève présentation du système *Aleph*, il faut retenir que ce système pousse à l'extrême le principe de la traduction par l'exemple puisque la manipulation des exemples sélectionnés est faite par une seule opération en langue source et une seule opération en langue cible, la même dans les deux langues.

À l'autre extrême se situe la traduction probabiliste puisqu'elle repose sur une sélection des données par égalité stricte de segments obtenus par segmentation en unités sous-phrastique puis en une modélisation probabiliste des unités, de leur ordre, et des relations de traduction entre elles entre les deux langues.

1.3 FINESSE DE DESCRIPTION : LA LANGUE OU LA SOUS-LANGUE ?

1.3.1 *Hypothèses sur les propriétés d'un corpus d'une sous-langue*

La traduction de sous-langues est plus simple que la traduction de texte tout venant, par exemple le système de traduction automatique TAUM-METEO (Isabelle, 1987) a été conçu pour la traduction de bulletins et d'alertes météorologiques de l'anglais vers le français et inversement. Les traductions produites par ce système sont considérées comme presque parfaites (voir section 1.4.1). Bien entendu, ce système traduira moins bien les textes d'autres domaines que la météorologie.

Nous nous intéressons au cas des bulletins météorologiques et plus particulièrement au corpus Météo[1] préparé par Leplus et coll. (2004). Ce corpus est un bi-corpus anglais-français dont la langue source est majoritairement l'anglais. Les traductions françaises ont été obtenues automatiquement puis révisées manuellement.

Nous nous intéressons aux propriétés de ce corpus permettant de dire que les bulletins météorologiques constituent une sous-langue en français et en anglais. Nous commençons cette étude par un décompte simple des mots, bigrammes, trigrammes et 8-grammes de mots différents de ce corpus. Notre hypothèse est qu'une sous-langue doit avoir un lexique fermé, c'est-à-dire qu'au-delà d'une certaine taille du corpus, aucun mot nouveau ne doit être rencontré. Afin de vérifier cette hypothèse simpliste, nous traçons l'évolution de la taille du vocabulaire en fonction du nombre de mots traités dans la figure 1(a) pour l'anglais. Un graphe similaire, non reproduit ici, a été obtenu en français. Cette courbe ne permet pas de vérifier à la lettre l'hypothèse puisque la taille du vocabulaire augmente toujours. Cependant, il est clair que cette évolution est grandement ralentie pour des tailles de corpus importantes. Par exemple, dans une expérience sur le corpus Météo en anglais, pour les sous-corpus de taille supérieure à 20 millions de mots, l'ajout d'un million de mots n'introduit que 80 nouveaux mots (moins de 1 % de la taille du vocabulaire). Nous pouvons conclure que le lexique des bulletins météorologiques est nettement contraint à la fois en français et en anglais.

Nous répétons le même procédé sur un corpus de dates anglaises dont le statut de sous-langue est clairement établi (voir sections 1.4.2 et 1.4.3). Ce corpus est constitué de l'intégralité des instances de dates acceptées par un langage régulier, soit 4 018 dates. Les résultats sont présenté en figure 2(a) et illustre la situation extrême où le vocabulaire est

1 Voir le site Environnement Canada `http://www.meteo.gc.ca/`, la source de ce corpus. le corpus Météo est disponible à l'adresse `http://rali.iro.umontreal.ca/Meteo/index.en.html` (le 18/01/2011)

effectivement fermé : nous ne constatons plus d'évolution dans la taille du vocabulaire à partir d'un corpus de 500 mots.

Une autre propriété pouvant caractériser les sous-langues est la saturation du nombre de n-grammes. Les n-grammes permettent d'aller au-delà des phénomènes purement lexicaux en incluant certains aspects grammaticaux. Selon la taille des n-grammes, ils couvriront partiellement ou totalement les syntagmes. Nous vérifions cette propriété en nous appuyant sur l'évolution du vocabulaire de bigrammes, de trigrammes et 8-grammes. En appliquant le même type de traitement que pour les mots, nous obtenons les courbes en figures 1(b) et 2(b) pour les bigrammes, 1(c) et 2(c) pour les trigrammes, 2(d) pour les 4-grammes et 1(d) pour les 8-grammes. La propriété reste valable dans une moindre mesure pour les bigrammes et trigrammes de mots du corpus Météo ainsi que pour les bigrammes de mots du corpus de dates. Dans les autres cas notre hypothèse simpliste est réfutée.

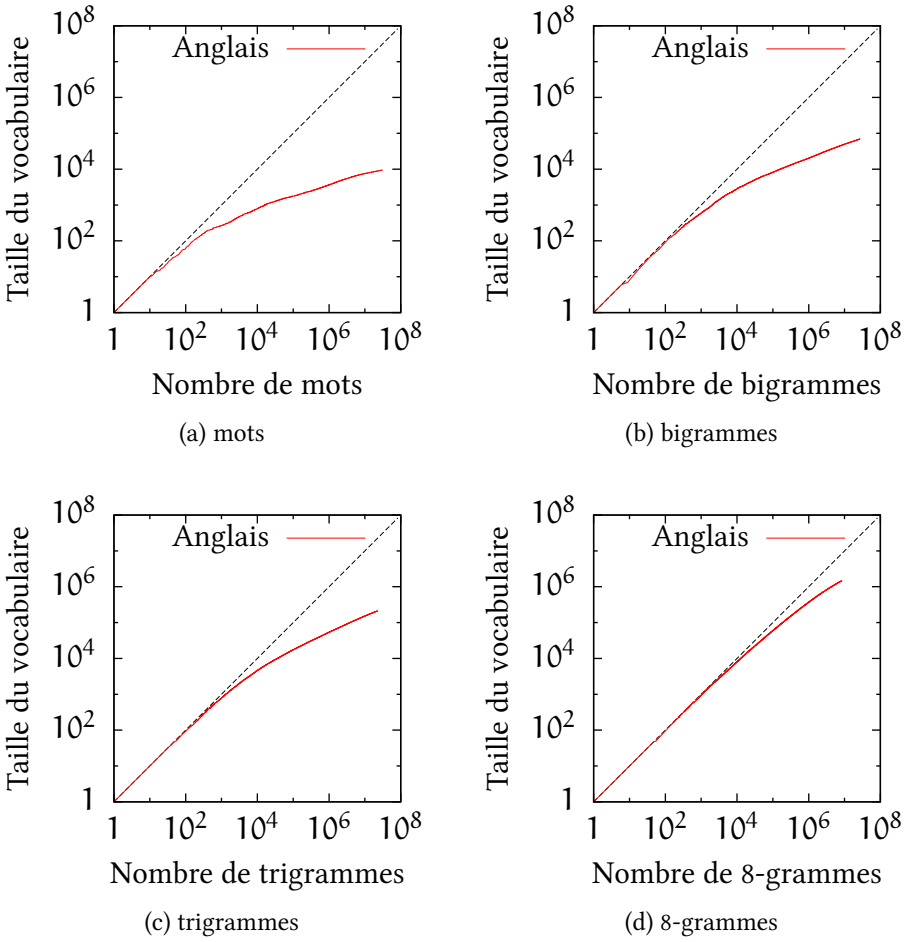
Debili et Souissi (2005) posent la question de la taille optimale du grain de traitement en analyse grammaticale. Selon eux, la plupart des étiqueteurs grammaticaux utilisent des patrons de bigrammes ou de trigrammes attestés afin de produire des modèles de longueurs supérieures selon un principe de chevauchement illustré ci-dessous.

trigramme attesté	$(\text{étiquette}_1, \text{étiquette}_2, \text{étiquette}_3)$
trigramme attesté	$(\text{étiquette}_2, \text{étiquette}_3, \text{étiquette}_4)$
4-gramme candidat	$(\text{étiquette}_1, \text{étiquette}_2, \text{étiquette}_3, \text{étiquette}_4)$

Ils rapportent que les analyseurs vont au-delà de la simple composition des modèles attestés puisque les analyseurs généralisent le principe précédent pour tous les modèles. Nous pouvons synthétiser cette généralisation sous la forme suivante :

n-gramme attesté ou candidat	$(\text{étiqu.}_1, \text{étiqu.}_2, \quad \ldots \quad \text{étiqu.}_n)$
n-gramme attesté ou candidat	$(\text{étiqu.}_2, \text{étiqu.}_3, \quad \ldots \quad \text{étiqu.}_{n+1})$
(n+1)-gramme candidat	$(\text{étiqu.}_1, \text{étiqu.}_2, \text{étiqu.}_3, \quad \ldots \quad \text{étiqu.}_{n+1})$

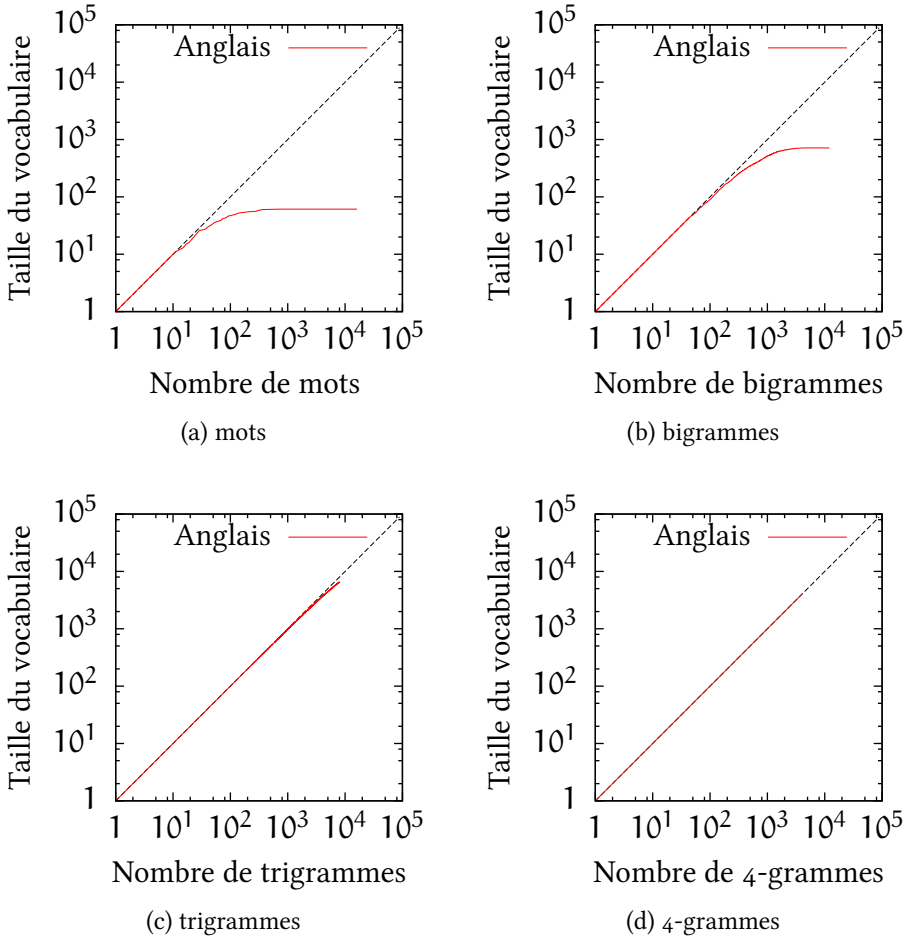
L'originalité de ce travail est de ne pas présenter une évaluation en usage qui consisterait à présenter les performances d'un étiqueteur. Les auteurs choisissent de présenter une évaluation dite *en définition* ou *intrinsèque*. Leurs conclusions sont originales : elles consistent à dire que les étiqueteurs grammaticaux devraient prendre en compte des règles de taille 10 pour l'anglais et l'arabe et 12 pour le français. Les tailles supérieures ne donneraient aucune amélioration. L'hypothèse des auteurs est que la phrase simple en français est longue en moyenne de 7



(a) mots (b) bigrammes

(c) trigrammes (d) 8-grammes

FIGURE 1 : Évolution de la taille du vocabulaire en fonction du nombre de mots, bigrammes, trigrammes et 8-grammes traitées pour l'anglais sur le corpus Météo (plus de 31 millions de mots).



(a) mots

(b) bigrammes

(c) trigrammes

(d) 4-grammes

FIGURE 2 : Évolution de la taille du vocabulaire en fonction du nombre de mots, bigrammes, trigrammes et 4-grammes traitées pour l'anglais sur le corpus de dates (voir section 1.4).

mots. Les tailles de modèles supérieures seraient presque exclusivement hapax. Les auteurs préconisent non pas de substituer les tailles de modèles 2 et 3 par des tailles 12 (pour le français) mais plutôt de combiner ces différentes tailles. Aucune stratégie pour arriver à cette fin n'est cependant proposée dans leur travail. Les tailles 2 et 3 doivent être conservées selon les auteurs puisque ces modèles ont une capacité productive bien supérieure aux tailles supérieures.

1.3.2 *Définition d'une sous-langue*

Dans le cadre applicatif de la traduction automatique, Kittredge (1987) définit une sous-langue comme un sous-système d'une langue ayant les propriétés suivantes :

- une sous-langue fait référence à un *domaine particulier* ou à un ensemble de domaines liés ;
- l'ensemble des phrases et textes d'une sous-langue reflète les usages d'une certaine communauté partageant *un ensemble de connaissances spécifiques au domaine* (des faits, des *a priori*, *etc.*), ces connaissances vont au-delà des connaissances partagées par les locuteurs de la langue ;
- une sous-langue possède toutes les *propriétés essentielles d'un système linguistique* telles que la cohérence, la complétude et le principe du moindre effort ;
- *une sous-langue est maximale* vis-à-vis du domaine, c'est-à-dire qu'il n'existe aucun système de taille supérieure ayant les mêmes propriétés.

Harris (1968) offre une définition plus formelle de la sous-langue : une sous-langue est un *ensemble fermé de phrases*. La fermeture concerne l'application des opérations définies dans la langue.

De ces deux définitions, nous retenons qu'une sous-langue possède toutes les propriétés d'une langue. De prime abord, restreindre la traduction à une sous-langue n'offrirait donc pas de simplification. Cependant, les deux définitions s'accordent sur l'existence d'une restriction de la sous-langue vis-à-vis de la langue. Cette restriction est exprimée en termes de domaine et de connaissances par Kittredge. Dans la définition de Harris, cette restriction est explicite, c'est l'ensemble des opérations nécessaires et suffisantes à la fermeture de l'ensemble des phrases de la sous-langue.

Comme exemple d'application, Harris et coll. (1989) ont étudié le cas de la sous-langue des articles scientifiques du domaine de l'immunologie.

En utilisant l'homogénéité de structure au sein d'une sous-langue, les auteurs ont montré qu'il existait une rupture attestée dans le domaine de l'immunologie : c'est-à-dire que les articles écrits avant la rupture constitue une sous-langue distincte des articles écrits par la suite. En utilisant les termes de Kittredge, nous pouvons dire que la rupture scientifique a eu une conséquence telle sur les *connaissances partagées* par les immunologues que les articles du domaine ne forment plus une seule sous-langue mais bien deux.

1.4 CAS D'ÉCOLE : TRADUCTION DE LA SOUS-LANGUE DES DATES CHINOISES EN ANGLAIS

1.4.1 *Introduction*

Il est bien connu que la traduction de textes restreints à un domaine est plus simple que la traduction de texte tout venant. Par exemple, TAUM-METEO (Isabelle, 1987) a été conçu pour la traduction de bulletins et d'alertes météorologiques de l'anglais vers le français et inversement. Les traductions produites par ce système sont considérées comme presque parfaites. Chandioux rapporte un taux de précision en mots de plus de 93 %.[2] Bien entendu, ce système ne permet pas de traduire des textes en dehors du domaine de la météorologie avec la même qualité.

Dans le cadre de la traduction par l'exemple, nous appelons un corpus d'exemples spécialement conçu pour la traduction d'une sous-langue une *sous-grammaire par l'exemple*.[3] Dans ce chapitre, nous explicitons les propriétés qu'une telle sous-grammaire devrait avoir afin de permettre la traduction d'un cas extrême : les dates. Pour toutes les langues, les dates sont énumérables dans un intervalle donné, de plus la sémantique est univoque. Nous comparons ici, deux systèmes de traduction fondés sur les données : un système statistique[4] et un système par l'exemple.

D'abord nous allons démontrer que le système des dates constitue une sous-langue. Ensuite nous allons discuter de la taille optimale de la sous-grammaire selon deux approches : analytique et empirique. L'étude empirique va nous conduire à introduire la notion de sous-grammaire par l'exemple. Nous allons utiliser cette notion afin de proposer une spécialisation d'un système de traduction par l'exemple et vérifions la qualité de ce système spécialisé en utilisant les sous-grammaires par l'exemple construites pour les dates. Enfin nous proposerons différentes

2 Voir le communiqué de presse du 24 mai 1997 à l'adresse `http://www.chandioux.com/press_meteo20.html`.

3 Il faut entendre grammaire au sens de l'ouvrage.

4 Bien que le terme le plus approprié pour désigner cette approche soit la traduction probabiliste, nous employons le terme statistique afin de ne pas dérouter le lecteur.

manières de construire automatiquement une sous-grammaire à partir d'un corpus d'exemples de traductions.

1.4.2 *Démonstration que les dates sont une sous-langue*

En reprenant point par point les propriétés d'une sous-langue selon la définition de Kittredge, nous pouvons démontrer que les dates constituent une sous-langue.

- Les dates font référence à un *domaine particulier* évident.
- L'*ensemble de connaissances spécifiques* aux dates est décrit par le type de calendrier (grégorien, julien, chinois, *etc.*) qui impose les différentes parties constituant une date (jour de la semaine, mois, année, *etc.*) ainsi que la date de référence (naissance supposée de Jésus-Christ, début de l'ère olympienne, *etc.*).
- Les conventions d'écriture font que les dates possèdent toutes les *propriétés essentielles d'un système linguistique* telles que la cohérence, la complétude et le principe du moindre effort.
- Il n'existe aucun système de taille supérieure au système des dates ayant les mêmes propriétés, on en déduit que le système des dates est *maximal* vis-à-vis du domaine.

Puisque le système des dates possède les quatre propriétés d'une sous-langue énoncé par Kittredge, on en conclut que le système des dates constitue une sous-langue. On peut remarquer que les propriétés énoncées ne dépendent d'aucune langue particulière, les dates constituent donc une sous-langue dans toutes les langues.

1.4.3 *Taille optimale d'une sous-grammaire*

Détermination expérimentale de la taille optimale d'une sous-grammaire

Dans cette section, nous déterminons la taille optimale d'un corpus d'exemples de traductions utilisé comme corpus d'entraînement pour la traduction fondée sur les données. Dans ce but, nous constituons un corpus d'étude de la sous-langue des dates. Le corpus d'étude est préparé par énumération des dates comprises entre le samedi 1er janvier 2000 inclus et le vendredi 31 décembre 2010 inclus, en anglais et en chinois. Ce corpus totalise 4 018 traductions de dates sous la forme *Sunday 1st April 2001* en Anglais et 二〇〇一年 四月 一日 星期天 en chinois. Les dates de ce corpus définissent un langage fini, donc trivialement régulier. Par conséquent il existe un automate d'états finis reconnaissant les dates de ce langage.

Nous préparons des corpus d'entraînement de tailles différentes par échantillonnage puis sous-échantillonnage. Un premier échantillon de 4 000 dates est obtenu à partir du corpus d'étude de 4 018 dates. Ensuite nous constituons des sous-échantillons en enlevant 250 dates prises aléatoirement à chaque étape. Pour les plus petites tailles, nous enlevons les dates 10 par 10 jusqu'à épuisement du corpus.

Nous comparons quatre systèmes de traduction fondés sur les données :

S1 un système de traduction par l'exemple : le système *Aleph*, décrit par Lepage et Denoual (2005b) ;

S2 un premier système par approche statistique, construit avec la collection d'outils Moses (Koehn et coll., 2007) pour l'entraînement et la traduction proprement dite, en combinaison avec l'outil Giza++ (Och et Ney, 2003) pour l'alignement des mots. La limite de distorsion est fixée à 4 afin de permettre le réordonnement ;[5]

S3 un second système par approche statistique, similaire au système précédent mais avec l'ajout d'une étape de développement par minimisation du taux d'erreur (Och, 2003) après entraînement ;

S4 un système de référence constitué par une mémoire de traduction capable de traduire uniquement le contenu du corpus d'entraînement. Ce système renvoie la traduction de la phrase ayant la distance d'édition (Levenshtein, 1966) la plus faible avec l'entrée. En particulier, lorsque la phrase à traduire existe dans le corpus d'entraînement, sa traduction est retournée sans altération.

La partie chinoise du corpus original est utilisée comme jeu de test, les dates anglaises correspondantes constituent le corpus de référence. Par construction, il existe une seule référence par date de test.

La longueur des dates étant très courte, exactement 4 mots, nous n'utilisons pas la mesure BLEU (Papineni et coll., 2002) pour comparer les résultats. Nous préférons utiliser un critère simple et plus adapté à notre situation : nous comparons la précision des systèmes de traduction vis à vis du corpus de référence. Nous définissons la précision d'un système par le pourcentage de traductions égales à la référence.

Le second système statistique (S_3) utilise le principe de la validation croisée pour les étapes d'entraînement et de développement. Pour ce système, le corpus d'entraînement est partitionné aléatoirement en dixièmes, 9 dixièmes sont utilisés pour l'entraînement, le dernier est réservé au développement. Nous répétons ce schéma dix fois, afin que chaque dixième soit utilisé une fois à l'étape de développement, nous

5 Valeur maximale dans notre cas.

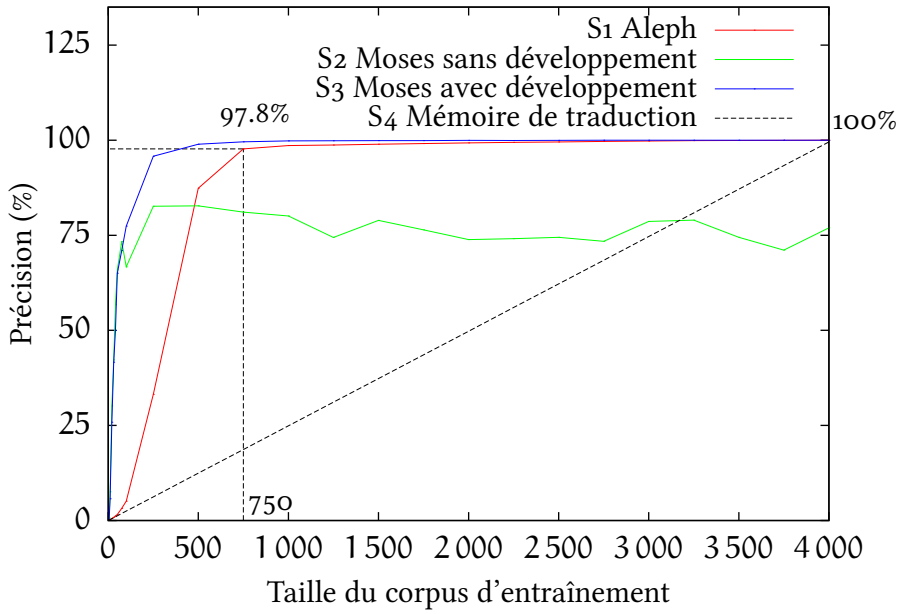


FIGURE 3 : Comparaison de quatre systèmes de traduction pour la traduction de dates du chinois vers l'anglais. Le système Moses avec étape de développement et *Aleph* traduisent plus de 97 % des dates en utilisant des corpus d'entraînement de plus de 750 exemples de traductions.

obtenons ainsi 10 systèmes de traduction.[6] Les résultats rapportés par la suite sont la moyenne arithmétique des résultats obtenus par les 10 systèmes.

Pour les quatre systèmes, il faut prévenir le biais statistique introduit par le tirage aléatoire des sous-corpus d'entraînement, pour chaque taille de sous-corpus, nous tirons 10 sous-corpus de taille identique. Le résultat retenu pour chaque système est la moyenne arithmétique des dix résultats obtenus à partir des 10 sous-corpus de taille identique. Les résultats sont présentés sous formes de courbes dans la figure 3.

Sept cent cinquante exemples de traductions de dates permettent de traduire plus de 97 % du corpus de 4 018 dates avec les systèmes S_1 et S_3, c'est-à-dire le système par l'exemple et le système statistique incluant une étape de développement. L'utilisation d'un corpus d'entraînement de taille plus importante n'entraîne pas d'amélioration significative.

Le système S_3, Moses avec étape de développement, obtient une précision de plus de 96 % en utilisant des corpus d'entraînement de plus de 250 dates. Les précisions obtenues par les systèmes S_1 et S_3 augmentent très légèrement jusqu'à 100 % pour 4 000 exemples et ne sont plus différentiables à partir de 1 000 exemples.

6 Nous respectons ici la terminologie consacrée en traduction statistique, c'est-à-dire qu'a partir d'un seul modèle de traduction entraîné indépendamment sur n corpus distinct, nous obtenons n systèmes de traduction prêt à l'emploi.

Le système S_2, Moses sans étape de développement présente un comportement chaotique. Il obtient de moins bons résultats que le système à mémoire de traduction (S_4) lorsqu'il est entraîné sur des corpus d'entraînement de taille supérieure à 3 250 exemples.

Taille optimale théorique d'une sous-grammaire

Nous montrons maintenant que 750 exemples ne sont pas nécessaires à la traduction de l'intégralité du corpus d'étude : théoriquement 58 dates suffisent pour traduire le corpus de 4 018 dates étudiées.

Comparons d'abord la forme des dates dans les langues étudiées. D'une part, en chinois, les dates sont composées des quatre parties suivantes dans cet ordre :

- l'année (par exemple : 二〇〇一年 pour 2001) ;
- le mois (par exemple : 四月 pour avril) ;
- le jour du mois (par exemple : 一日 pour 1er du mois) ;
- le jour de la semaine (par exemple : 星期一 pour lundi).

En anglais britannique, les dates sont écrites dans l'ordre inverse, par exemple : *Thursday 1st January 2009*.

Puisque les parties constitutives des dates sont clairement identifiables et coïncident d'une langue à l'autre, nous proposons de décrire chaque partie de date comme des phénomènes indépendants du contexte à l'aide de sous-grammaires par l'exemple.

Une grammaire par l'exemple est définie par un ensemble d'exemples de traductions en contextes droit et gauche donnés. L'idée principale des grammaires par l'exemple est de réunir un ensemble d'exemples nécessaire et suffisant pour décrire un phénomène linguistique. À cette fin, seul le phénomène décrit est variable au sein des exemples, le reste étant invariant. Par exemple, nous pouvons constituer une sous-grammaire pour la description des mois en prenant *Monday 1st* comme contexte gauche et *2001* comme contexte droit : c'est la partie invariante de la sous-grammaire. Les 12 mois seront écrits dans ces contextes : c'est la partie variable des exemples permettant de décrire le phénomène recherché.

En suivant le même principe, nous avons écrit les sous-grammaires par l'exemple permettant la traduction complète des dates, elles sont illustrées dans les tableaux 1(a) pour les années, 1(b) pour les mois, 1(c) pour les jours du mois, et 1(d) pour les jours de la semaine. Ensemble, ces quatre sous-grammaires font intervenir 58 exemples distincts : 11 pour les années, 12 pour les mois, 31 pour les jours du mois, et 7 pour les jours de la semaine, auxquels il faut retrancher 3 à cause de l'exemple en

commun. Nous appelons *prototype* cet exemple commun aux quatre sous-grammaires. Puisque ces quatre sous-grammaires ensembles décrivent entièrement tous les phénomènes des dates, 58 dates suffisent pour traduire l'ensemble du corpus de 4 018 dates. Ensemble, elles forment une grammaire nécessaire et suffisante des dates.

Démonstration du caractère suffisant. La preuve s'effectue en quatre étapes et utilise le principe de la traduction par analogie. À chaque étape, nous prouvons que nous pouvons traduire une partie d'une date chinoise de la forme AMJS[7] en utilisant la sous-grammaire adéquate. Soit AMJS, la date à traduire. Avec un exemple issu d'une sous-grammaire et le prototype $A_0M_0J_0S_0$, nous pouvons toujours écrire l'équation analogique suivante :

$$\text{exemple} \ : \ \text{prototype} \ :: \ \text{AMJS} \ : \text{AM}_0\text{JS}$$

En répétant l'opération et quel que soit l'ordre d'application, nous obtenons $A_0M_0J_0S_0$, dont nous connaissons la traduction : $\widehat{S_0}\widehat{J_0}\widehat{M_0}\widehat{A_0}$. Dans la langue cible, les équations analogiques sont résolues dans l'ordre inverse, remplaçant étape par étape les parties de la traduction. Finalement, cette méthode produit $\widehat{S}\widehat{J}\widehat{M}\widehat{A}$, dont les parties $\widehat{S}$, $\widehat{J}$, $\widehat{M}$, et $\widehat{A}$ sont tirées des sous-grammaires. □

Démonstration du caractère nécessaire. Si on enlève un seul exemple d'une sous-grammaire, alors certaines dates ne seront plus exprimables. □

1.4.4 *Système de traduction par l'exemple utilisant les sous-grammaires*

Dans cette section, nous décrivons une modification du système de traduction *Aleph* pour traiter les sous-grammaires. La modification porte sur les récursions qui ont lieu dans *Aleph* : le système ne considère qu'une sous-grammaire par niveau de récursion. En effet, puisque chaque phénomène est décrit par une sous-grammaire dédiée, il est possible de traduire chaque phénomène indépendamment à chaque niveau de récursion.

Une conséquence et un avantage de cette propriété est que le temps de traduction est réduit de manière significative. Les sous-grammaires apportent la modularité au système de traduction : les phénomènes linguistiques indépendants sont traités de manière indépendante par le système de traduction. D'un point de vue « génie linguiciel » (Lafourcade, 1994), une sous-grammaire par l'exemple peut être considérée comme un composant logiciel offrant une grande « généricité » (permet de traiter un

7 Pour Année-Mois-Jour-jour de la Semaine.

TABLEAU 1 : Sous-grammaires par l'exemple pour la traduction des années (tableau (a)), des mois (tableau (b)), des jours du mois (tableau (c)), et des jours de la semaine (tableau (d)). L'exemple commun, appelé prototype est « 二〇〇一年 四月 一日 星期一 ↔ Monday 1st April 2001 », cet exemple est mis en évidence dans les quatre sous-grammaire.

$A \leftrightarrow \hat{A}$
二〇〇〇年 四月 一日 星期一 ↔ Monday 1st April 2000
二〇〇一年 四月 一日 星期一 ↔ Monday 1st April 2001
二〇〇二年 四月 一日 星期一 ↔ Monday 1st April 2002
二〇〇三年 四月 一日 星期一 ↔ Monday 1st April 2003
⋮
二〇一〇年 四月 一日 星期一 ↔ Monday 1st April 2010

(a) année, nombre d'exemples : 11

$M \leftrightarrow \widehat{M}$
二〇〇一年 一月 一日 星期一 ↔ Monday 1st January 2001
二〇〇一年 二月 一日 星期一 ↔ Monday 1st February 2001
二〇〇一年 三月 一日 星期一 ↔ Monday 1st March 2001
二〇〇一年 四月 一日 星期一 ↔ Monday 1st April 2001
⋮
二〇〇一年 十二月 一日 星期一 ↔ Monday 1st December 2001

(b) mois, nombre d'exemples : 12

$J \leftrightarrow \hat{J}$
二〇〇一年 四月 一日 星期一 ↔ Monday 1st April 2001
二〇〇一年 四月 二日 星期一 ↔ Monday 2nd April 2001
二〇〇一年 四月 三日 星期一 ↔ Monday 3rd April 2001
二〇〇一年 四月 四日 星期一 ↔ Monday 4th April 2001
⋮
二〇〇一年 四月 三十一日 星期一 ↔ Monday 31st April 2001

(c) jour du mois, nombre d'exemples : 31

$S \leftrightarrow \hat{S}$
二〇〇一年 四月 一日 星期一 ↔ Monday 1st April 2001
二〇〇一年 四月 一日 星期二 ↔ Tuesday 1st April 2001
二〇〇一年 四月 一日 星期三 ↔ Wednesday 1st April 2001
二〇〇一年 四月 一日 星期四 ↔ Thursday 1st April 2001
⋮
二〇〇一年 四月 一日 星期天 ↔ Sunday 1st April 2001

(d) jour de la semaine, nombre d'exemples : 7

grand nombre de phénomènes différents) et facilite l'« extensibilité » (on pourrait définir des types de composants compatibles mais basés sur un autre principe que l'analogie).

Description du système

La traduction est effectuée en deux étapes :

1. rassembler les exemples nécessaires à la traduction depuis les sous-grammaires ;
2. traduire la date d'entrée en combinant les exemples précédemment rassemblés.

L'organisation en quatre sous-grammaires réduit le nombre d'exemples manipulés en même temps. Pour chaque exemple d'une sous-grammaire, seul le phénomène décrit varie, les contextes restants inchangés. Pendant la première étape, l'exemple le plus proche de l'entrée est sélectionné parmi tous les exemples. Nous employons la distance de Levenshtein pour comparer la date à traduire aux exemples connus. Le fait que les contextes soient identiques au sein d'une sous-grammaire implique que l'exemple sélectionné permettra d'obtenir la meilleure traduction possible.

Une adaptation du triangle de Vauquois

Le système de traduction par sous-grammaires possède des similarités avec le triangle de Vauquois (voir Vauquois et Boitet (1985) et Boitet (2007) pour une présentation de ce triangle).

Chaque traduction est représentable par un trapèze, lui-même faisant partie d'un triangle (voir figure 4). La première étape de notre système est similaire à l'analyse dans le triangle de Vauquois. La partie gauche est ascendante et part de la phrase d'entrée. Le long du coté gauche, la phrase est de plus en plus proche de la phrase prototype (notée P). Le transfert de Vauquois est équivalent à l'étape de récursion (la flèche reliant S_{n+1} à $\widehat{S_{n+1}}$ au sommet du trapèze). La même méthode est appliquée à la phrase résultante, d'où le nom d'étape de récursion. La troisième étape est similaire à la génération dans le triangle de Vauquois. Lors de cette étape, une autre équation analogique est résolue en langue cible. Enfin la traduction construite ferme le trapèze par la grande base et permet la récursion.

Illustration étape par étape de la traduction dans notre système

La figure 6 détaille la traduction de la date chinoise S_0=« 二〇〇七年 十一月 十六日 星期五 » par notre système.

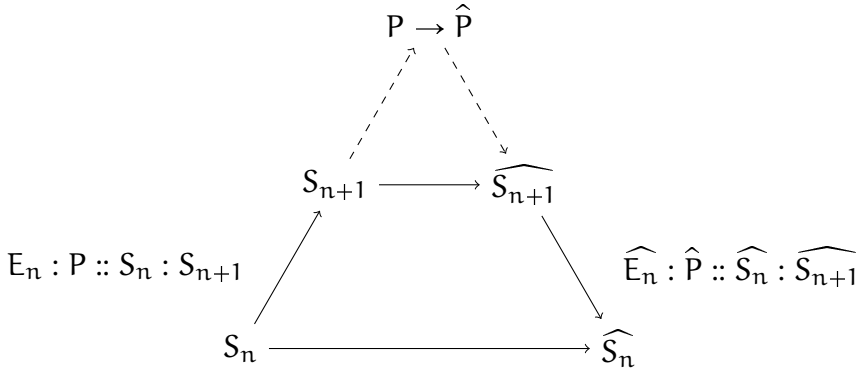


FIGURE 4 : Similitude de notre système de traduction avec le triangle de Vauquois.

Les calculs débutent en chinois en transformant, étape par étape, la date d'entrée en la date prototype P. Une sous-grammaire différente est employée à chaque niveau de récursion n. D'abord, l'exemple le plus proche E_0 est extrait de la sous-grammaire 1(d). Ensuite l'équation analogique $E_0 : P :: S_0 : x$ est résolue et fournit les solutions S_1. Si l'équation n'a pas de solutions, alors la sous-grammaire est simplement considérée comme inappropriée et ignorée. La sortie de cette étape est alors égale à l'entrée : $S_1 = S_0$.

La récursion prend fin lorsque la phrase courante est identique au prototype, soit $S_n = P$. Dans notre exemple, la date chinoise S_4 =« 六七八九年 四月 一日 星期一 » est égale au prototype P. Pour résumer, à ce point, les traductions de P, E_3, et S_4 sont connues et notées $\hat{P}$, $\widehat{E_3}$, et $\widehat{S_4}$. Les calculs suivants sont effectués en langue anglaise avec l'équation analogique $\widehat{E_3} : \hat{P} :: x : \widehat{S_4}$. Les solutions de cette équation sont les traductions de S_3 : $x = \hat{S}_3$. Le calcul continue par récursion jusqu'à l'obtention de $\hat{S}_0$.

La traduction de l'entrée S_0 est $\hat{S}_0$. Dans l'exemple, $\hat{S}_0$ = « Friday 16th November 2007 ».

1.4.5 *Expériences de traductions*

Dans cette section, nous évaluons notre système de traduction par sous-grammaires afin de vérifier qu'il est bien capable de traduire les 4 018 dates de l'ensemble d'origine. En même temps, nous le comparons avec un système de traduction statistique.

Nous utilisons de nouveau l'ensemble de dates décrit dans la section 1.4.3, soit les 4 018 dates comprises entre le samedi 1er janvier 2000 et le vendredi 31 décembre 2010. La partie chinoise sert de jeu de test et la partie anglaise sert de corpus de référence.

Nous comparons notre système avec le système de traduction statistique Moses. Cependant les paramètres par défaut de Moses ne permettent pas le réordonnement des mots.[8] C'est pourquoi nous comparons 5 configurations de Moses en faisant varier la taille de la fenêtre de réordonnement de 0 à 4 mots. Cette expérience est effectuée deux fois : une fois avec phase de développement et une fois sans.

Les systèmes sont comparés à l'aide de TER (Snover et coll., 2006) et de la précision telle que définie dans la section 1.4.3. Tous les systèmes utilisent les sous-grammaires présentées dans les tables 1(a), 1(b), 1(c) et 1(d).

Tableau 2 : Comparaison de notre système et de Moses.

	Sans développement		Avec développement	
Système (taille de la fenêtre de réordonnement)	TER	Précision (%)	TER	Précision (%)
Moses(0)	0,75	0	0,68	0,20
Moses(1)	0,75	0	0,68	0,20
Moses(2)	0,25	0	0,34	0,05
Moses(3)	0,25	0	0,33	0,02
Moses(4-...)	0,02	92	0,09	69,00
Système à sous-grammaires	**0,00**	**100**	—	—
Mémoire de traduction	0,67	1	—	—

Les résultats sont présentés dans le tableau 2. Comme attendu, et par construction, notre système est capable de traduire toutes les dates du corpus d'étude sans exception : il obtient une précision de 100 %. Dans le cas de Moses, le meilleur résultat est obtenu lorsque la taille de la fenêtre de réordonnement est égale ou supérieure à 4. Puisque les dates anglaises sont écrites dans le sens opposé aux dates chinoises, et puisque les dates sont composées de 4 termes, il est normal que la taille de la fenêtre de réordonnement soit 4. Moses(2-3) obtient un score TER de 0,25 parce que la taille de la fenêtre permet de « rétablir » l'ordre de quelques mots. D'un autre coté, lorsque la taille de la fenêtre est de 0 ou 1, aucun réordonnement n'est possible et le score TER est de 0,75, ce qui est le maximum pour quatre mots.

Lorsque la taille de la fenêtre de réordonnement est positionnée à 4, le système Moses traduit toutes les dates à l'exception de 345. Nous avons remarqué que ces dates sont toutes en avril. Puisque les dates de notre

8 Le terme consacré pour le réordonnement des mots par Moses est « distorsion » puisqu'un modèle de distorsion probabiliste est employé.

sous-grammaires dérivent du prototype « lundi 1er avril 2001 », avril apparaît dans la majorité des exemples et introduit ainsi un biais par rapport à un ensemble de dates prises aléatoirement comme présenté en section 1.4.3. Les systèmes de traduction probabiliste ne peuvent pas exploiter les sous-grammaires par l'exemple que nous avons introduit, en effet le but des sous-grammaires par l'exemple est d'éliminer les redondances du corpus d'entraînement, ce qui affecte gravement la représentativité statistique.

Il faut noter que le système Moses(4-...) obtient de meilleurs scores sans étape de développement, la raison principale est que l'étape de développement requiert un dixième du corpus d'entraînement comme jeu de développement, ce qui réduit d'autant la quantité d'exemples disponibles pour l'entraînement.

1.4.6 *Perspectives : acquisition automatique de sous-grammaires par l'exemple*

Les expériences décrites en section 1.4.3 montrent que l'acquisition automatique de sous-grammaires par l'exemple à partir de corpus d'exemples de traduction est un problème de généralisation. L'étape de développement dans l'approche probabiliste de la traduction automatique est en fait une généralisation. Dans le cadre des méthodes basées sur l'exemple, de meilleurs résultats ont été obtenus à l'aide d'exemples généralisés tels que les sous-grammaires par l'exemple introduites en section 1.4.3.

Les dates sont un exemple de langage régulier. Une technique de minimisation d'automate permet de représenter les 4 018 dates chinoises en un automate fini tel qu'en figure 5. Cet automate distingue clairement les 4 parties variables d'une date. Dans un véritable corpus, nous pouvons nous attendre à ce que seulement une partie des dates apparaissent. Le problème de généralisation consiste alors à obtenir un automate du même genre qu'en figure 5 ou des sous-grammaires semblables à celles de la section 1.4.3 à partir d'un ensemble incomplet de dates.

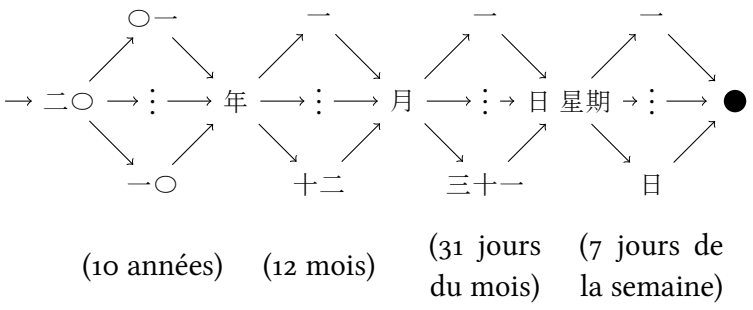


FIGURE 5 : Automate minimisé obtenu à partir du corpus de 4 018 dates chinoises. Les 4 parties variables des dates sont clairement identifiables.

L'algorithme d'apprentissage non-supervisé ADIOS (Automatic DIstillation Of Structure) (Solan et coll., 2003) peut être une méthode pour acquérir automatiquement des sous-grammaires à partir d'un corpus d'exemples grâce à son opération de généralisation.

1.4.7 *Conclusion de la traduction de la sous-langue des dates*

Dans cette section, nous avons vérifié la validité d'un système de traduction par l'exemple sur une petite sous-langue contrainte. La sous-langue utilisée en exemple est la sous-langue des dates en chinois et en anglais entre le samedi 1er janvier 2000 et le vendredi 31 décembre 2010. Dans notre configuration, les dates chinoises ont exactement une traduction en anglais et réciproquement. Nous avons constaté par l'expérience que le système de traduction probabiliste Moses avec développement et que le système de traduction par l'exemple *Aleph* ont besoin de 750 exemples de traduction choisis aléatoirement parmi les 4 018 afin de traduire presque toutes les dates de l'intervalle. La même expérience indique la supériorité de Moses sur *Aleph* lorsque les corpus d'entraînement ont moins de 750 exemples de traduction.

Dans un second temps, nous avons prouvé que 58 exemples de traduction bien choisis suffisent pour traduire les 4 018 dates. Ces 58 dates sont toutes des variations d'un seul exemple de traduction que nous appelons prototype. Les sous-grammaires par l'exemple sont constituées de dates identiques au prototype à l'exception d'une partie variable.

De plus, nous avons décrit une version modifiée du système *Aleph* qui met à profit la notion de sous-grammaire et permet de traduire la sous-langue dans son intégralité. Ce résultat doit être comparé aux 8 % du corpus que le système Moses reste incapable de traduire dans sa meilleure configuration. Par construction, nos sous-grammaires par l'exemple contiennent les exemples nécessaires à la traduction des 4 018 dates, en retour cette organisation des données est statistiquement déséquilibrée, ce qui désavantage la traduction par approche probabiliste.

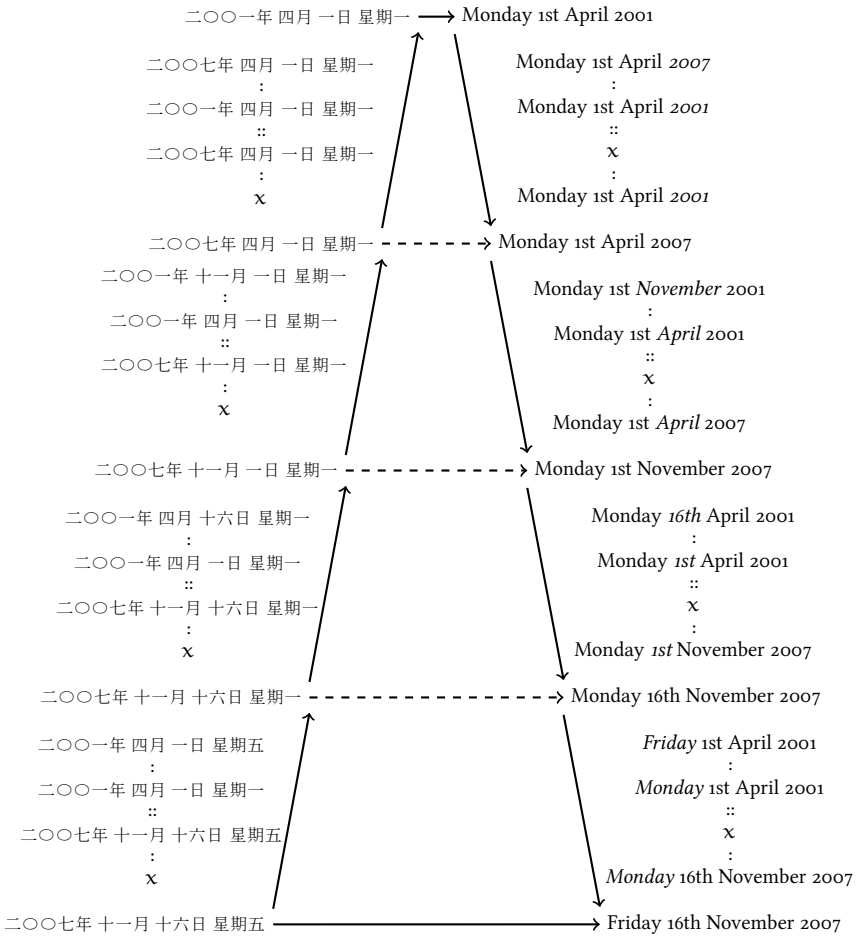


FIGURE 6 : Exécution pas à pas du système de traduction par sous-grammaires dérivé d'*Aleph*. L'entrée est « 二〇〇七年 十一月 十六日 星期五 ». Les flèches aux traits discontinus indiquent les étapes de traduction du système. Les flèches bidirectionnelles correspondent aux liens de traductions. Le prototype P est constant, S_n sont les dates en entrées. E_n est la date la plus proche de l'entrée S_n et S_{n+1} est la solution à l'équation analogique $E_n : P :: S_n : x$. En langue cible, ici l'anglais, $\widehat{E_n}$ est la traduction de E_n, $\widehat{P}$ est la traduction de P, $\widehat{S_{n+1}}$ est la date d'entrée et $\widehat{S_n}$ est la solution à l'équation analogique $\widehat{E_n} : \widehat{P} :: x : \widehat{S_{n+1}}$.

1.5 CONCLUSION

Dans ce chapitre nous avons mis en évidence le problème de la sélection des exemples en traduction automatique par l'exemple. Nous avons montré que ce problème est lié à un domaine de traduction trop large. En effet, un système de traduction élaboré dans le but de traduire une sous-langue suffisamment restreinte permet de traduire de nouveaux textes de cette sous-langue avec une grande qualité. Dans un premier temps, nous avons cité l'exemple du système TAUM-METEO, développé spécialement pour la traduction des bulletins et alertes météorologiques entre le français et l'anglais. Dans un second temps, nous avons détaillé un cas d'école, la sous-langue des dates anglaises et chinoises, afin de prouver qu'avec une organisation des exemples en sous-grammaires, le système de traduction par l'exemple *Aleph* n'est pas confronté au problème de la sélection des exemples de traduction.

Afin de proposer une solution automatisable au problème de la sélection des exemples de traduction pour le système *Aleph*, nous allons maintenant analyser la structure des analogies présentes dans un texte. Pour mener à bien cette analyse, nous aurons besoin d'énumérer toutes les analogies présentes dans un texte. Dans le chapitre suivant, nous allons donc proposer une méthode permettant d'énumérer toutes les analogies d'un texte. Cette méthode sera conçue pour être à la fois adaptable aux deux définitions actuelles de l'analogie et capable de traiter divers grains : chaînes de caractères, n-grammes de mots, phrases, *etc.*

2

ÉNUMÉRATION DE TOUTES LES ANALOGIES D'UN TEXTE

Le chapitre précédent a mis en évidence que l'utilisation de sous-grammaires dans un système de traduction par l'exemple, le système *Aleph* permet d'éliminer le problème de la sélection des exemples. Afin de constituer automatiquement des sous-grammaires, il faut d'une part être capable d'énumérer toutes les analogies d'un texte et d'autre part être capable de faire émerger une généralisation des exemples grâce à l'analogie. Dans ce chapitre, nous allons présenter une méthode générique pour énumérer toutes les analogies d'un texte, sans répétition, et conformément aux définitions de Lepage (2004a) et Yvon (2003). Cette méthode permettra par construction de restreindre l'énumération aux analogies de n-grammes de mots en imposant une borne supérieure à la longueur des termes considérés. Grâce à cette méthode, il sera possible de mettre en évidence les patrons d'analogies présents dans un texte. Ces patrons pourront nous servir à la mise au point d'une technique de lissage qui sera présentée au chapitre suivant.

2.1 INTRODUCTION

Dans le but de proposer une solution au problème de la sélection des exemples mis en évidence dans le chapitre précédent, nous souhaitons observer la structure des analogies dans les textes en langue source et cible. Pour effectuer cette observation, nous avons besoin d'une méthode générique d'énumération des analogies. Idéalement la méthode devrait être capable d'énumérer les analogies entre phrases, mots ou même chunks, plus généralement entre toutes sous-séquences d'un texte. Une solution à ce problème est présentée dans ce chapitre.

Le problème de la validité sémantique des analogies identifiées par cette méthode n'est pas traité dans ce chapitre. En effet les analogies identifiées par cette méthode respectent par construction une formalisation de l'analogie, leurs validité dépend donc de la formalisation elle-même. En ce qui concerne les analogies entre phrases, Lepage (2004b) rapporte une proportion d'analogies sémantiquement valides de 96 %.

Un certain nombre de travaux en traitement automatique des langues exploitent l'analogie. Nous n'en citons que quelques-uns ici. Par exemple, sur des tâches de segmentation morphologique, Lavallée et Langlais (2010) ont récemment obtenu d'excellents résultats dans la découpe des mots par analogie. Claveau et L'Homme (2005b), entre autres auteurs, avaient auparavant étudié, en faisant usage de l'analogie, dans quelle mesure la similarité liait la forme et le sens des mots : *connector* : *to connect* :: *editor* : *to edit*. En plus des analogies entre mots eux-mêmes, Hathout (2009) a récemment exploité les analogies entre définitions extraites du TLFi pour construire automatiquement des familles de mots liés par la forme et le sens. Dans le même ordre d'idée, Langlais et coll. (2008) avaient proposé d'utiliser l'analogie pour forger de nouvelles équivalences terminologiques dans le domaine médical à cheval sur deux langues. Sur le seul plan sémantique, Turney (2008) a quant à lui présenté une approche générale au problème de l'association entre mots utilisant l'analogie entre vecteurs contextuels : *mason* : *stone* :: *carpenter* : *wood*, approche qu'il prétend généralisable aux relations de synonymie et d'antonymie. Lepage et Denoual (2005a) quant à eux ont conçu un système de traduction automatique entièrement fondé sur l'analogie. Dans le cadre de la traduction automatique aussi, Denoual (2007) et Langlais et Patry (2007) ont montré la possibilité de traduire certains mots inconnus par analogie.

Une fois les formalisations utilisées et les conventions de notations posées, nous décrirons la méthode d'énumération exhaustive des analogies. Cette méthode d'énumération parcourt un espace de recherche, en conséquence nous discuterons de la meilleure stratégie de parcours ainsi que de la meilleure stratégie d'initialisation.

2.2 TYPES D'ANALOGIES ET FORMALISATIONS

2.2.1 *Types d'analogies*

Étant donné un corpus (voir figure 7), l'objectif est d'énumérer toutes les analogies non-triviales A : B :: C : D, « non-triviales » signifie que les termes A, B, C et D sont tous différents deux à deux. Un exemple est *I'd like some instant coffee.* : *I'd like some instant coffee, please.* :: *May I have the menu?* : *May I have the menu, please?*, analogie extraite du BTEC illustré à la figure 7.

I'd like some instant coffee.↵
I'd like some instant coffee, please.↵
May I have the menu?↵
May I have the menu, please?↵
May I have tea instead of coffee?↵
Tea, with milk.↵

FIGURE 7 : Extrait de la partie anglaise du BTEC distribuée lors de la campagne d'évaluation des systèmes de traduction IWSLT 2008 (Paul, 2008). Les fins de lignes sont matérialisées par le symbole ↵.

Les analogies triviales sont les analogies de la forme A : A :: B : B et A : A :: A : A telle que *I'd like some instant coffee.* : *I'd like some instant coffee.* :: *May I have the menu?* : *May I have the menu?*. Évidemment, elles ne donnent aucune information, et nous ne sommes donc pas intéressé à leur énumération, qui est d'ailleurs triviale.

Un cas particulier d'analogie triviale est le cas dégénéré ϵ : ϵ :: ϵ : ϵ où ϵ représente la chaîne vide.

En résumé, il n'y a aucun intérêt à énumérer les analogies de la forme A : A :: B : B.

Définition 1. *Une analogie A : B :: C : D est qualifiée de* ***triviale*** *lorsqu'elle est composée de deux paires de termes égaux, c'est-à-dire qu'il existe une analogie équivalente (voir la propriété 5) d'une des formes suivantes : A′ : A′ :: B′ : B′ ou A : A :: A : A.*

2.2.2 *Formalisations de l'analogie entre chaînes de caractères*

Deux formalisations de l'analogie ont été proposées récemment.

Définition 2. *Selon Lepage (2004a), l'analogie est définie par :*[1]

$$\begin{cases} d(A,B) &= d(C,D) \\ d(A,C) &= d(B,D) \\ \forall a, |A|_a - |B|_a &= |C|_a - |D|_a \end{cases}$$

Définition 3. *Selon Yvon (2003) (voir également (Stroppa et Yvon, 2005)), l'analogie est définie par :*[2]

$$A : B :: C : D \Rightarrow A \bullet D \cap B \bullet C \neq \varnothing$$

Plutôt que de choisir l'une ou l'autre de ces formalisations, nous préférons utiliser une caractérisation relâchée compatible avec les deux formalisations :

Définition 4.

$$A : B :: C : D \Rightarrow \forall a, |A|_a + |D|_a = |B|_a + |C|_a$$

Il est évident que la définition 4 est une version relâchée de la formalisation de Lepage puisqu'il suffit d'enlever la contrainte sur les distances d'édition entre chaînes.

Quant à la formalisation de Stroppa et Yvon, la démonstration est simple :

Démonstration. Par définition de l'opération de mélange, on a :

$$\forall a, |A \bullet B|_a = |A|_a + |B|_a$$

Par conséquent :

$$A \bullet D \cap B \bullet C \neq \varnothing \Rightarrow \forall a, |A|_a + |D|_a = |B|_a + |C|_a$$

Donc toute analogie respectant la définition 3 de Stroppa et Yvon respecte la définition 4. □

Il est extrêmement simple de filtrer les analogies obtenues par cette méthode afin de ne retenir que les analogies respectant l'une ou l'autre des formalisations précédentes. Dans la suite de ce chapitre, l'analogie se réfère à la définition 4.

Une conséquence immédiate de la définition de l'analogie est la propriété :

1 d signifie la distance d'édition canonique. Cette distance prend en compte les deux opérations élémentaires suivantes : la suppression et l'insertion. $|A|_a$ représente le nombre de caractères a présents dans la chaîne A.

2 Le symbole $\bullet$ désigne l'opération *mélange* sur les chaînes de caractères.

Propriété 5. *Toute analogie $A:B::C:D$ peut s'écrire sous les huit* ***formes équivalentes*** *(Lepage, 2004a) suivantes :*

$$
\begin{aligned}
&A : B :: C : D \\
&A : C :: B : D \quad \textit{(permutation des moyens)} \\
&B : A :: D : C \quad \textit{(inversion des rapports)} \\
&B : D :: A : C \\
&C : A :: D : B \\
&C : D :: A : B \quad \textit{(symétrie de la conformité)} \\
&D : B :: C : A \quad \textit{(permutation des extrêmes)} \\
&D : C :: B : A \quad \textit{(symétrie de lecture)}
\end{aligned}
$$

Des formes équivalentes de l'analogie, nous extrayons les quatre permutations idempotentes ($p \circ p = id$) suivantes :

$$
\begin{aligned}
&A : B :: C : D \overset{p_{AB}}{\longleftrightarrow} A : B :: C : D \quad \textit{(identité)} \\
&A : B :: C : D \overset{p_{AC}}{\longleftrightarrow} A : C :: B : D \quad \textit{(permutation des moyens)} \\
&A : B :: C : D \overset{p_{CD}}{\longleftrightarrow} C : D :: A : B \quad \textit{(symétrie de la conformité)} \\
&A : B :: C : D \overset{p_{DB}}{\longleftrightarrow} D : B :: C : A \quad \textit{(permutation des extrêmes)}
\end{aligned}
$$

Il est facile de démontrer que ces quatre permutations suffisent pour produire les huit formes équivalentes de l'analogie par composition. En effet, on a :

$$
\begin{aligned}
A : B :: C : D &= p_{AB}(I) \\
A : C :: B : D &= p_{AC}(I) \\
B : A :: D : C &= p_{AC} \circ p_{CD} \circ p_{AC}(I) \\
B : D :: A : C &= p_{CD} \circ p_{AC}(I) \\
C : A :: D : B &= p_{CD} \circ p_{DB}(I) \\
C : D :: A : B &= p_{CD}(I) \\
D : B :: C : A &= p_{DB}(I) \\
D : C :: B : A &= p_{AC} \circ p_{DB}(I) \\
&\text{avec } I = A : B :: C : D
\end{aligned}
$$

2.2.3 *Conventions d'écriture*

Notons un texte quelconque T, le symbole à la position i dans T étant noté $T[i]$. La sous-chaîne de T allant de la position i incluse à la position j exclue est notée $T[i, j[$.

De plus, toute chaîne du texte T peut être localisée exactement par un couple (longueur, positions). Par exemple, dans le texte *May I have tea*

instead of coffee?, la chaîne *tea* peut être représentée par $(3, \{11, 18\})$ puisque la longueur de la chaîne *tea* est 3 et n'apparaît qu'aux positions 11 et 18 dans le texte.

La localisation de A dans T notée $l(A)$ est définie par :

$$l(A) = (m, \{i_1, \ldots i_p\}) \,/\, T[i_j, i_j + m[\; = A$$

L'inclusion de la chaîne A dans T notée $A \sqsubset T$ est définie par :

$$A \sqsubset T \iff \exists u, v \,/\, u \cdot A \cdot v = T$$

Évidemment la localisation d'une chaîne A dans T implique l'inclusion de A dans T.

2.3 MÉTHODE ITÉRATIVE PERMETTANT L'ÉNUMÉRATION EXHAUSTIVE ET SANS REDITES DES ANALOGIES D'UN TEXTE

2.3.1 *Analogies subséquentes*

Nous proposons ici une méthode pour l'énumération complète de toutes analogies contenues dans un texte par constructions itératives des analogies en commençant par des analogies graines.

Pour une analogie $A : B :: C : D$; nous appelons analogie subséquente une analogie de la forme $AA' : BB' :: CC' : DD'$ où A', B', C' et D' suivent respectivement A, B, C et D dans le texte et sont en relation d'analogie $A' : B' :: C' : D'$.

Toutes analogies subséquentes d'une analogie graine $A : B :: C : D$ peuvent être dérivées par application itérative d'une opération successeur basée sur les sous-chaînes d'un texte.

2.3.2 *Successeurs d'une analogie*

Successeurs d'une sous-chaîne d'un texte

Nous rappelons que la notation $l(A) = (m, \{i_1, i_2, \ldots, i_p\})$ dans le texte T représente toutes les instances $T[i_j, i_j + m[$ de la chaîne A dans T. L'ensemble des successeurs de cette chaîne est l'ensemble $\{T[i_j, i_j + m + 1]\} = \{A \cdot a_q\}$ où le symbole a_q parcourt l'ensemble des caractères suivant la chaîne A dans T. L'ensemble des successeurs de A est donc défini comme suit :

$$\mathit{SuccCh}(A) = \{l(A \cdot a_q) \,/\, a_q = T[i_j + m + 1] \wedge A = T[i_j, i_j + m[\}$$

Par exemple, les successeurs de $\mathtt{inst}$ dont la localisation est $l(\mathtt{inst}) = (4, \{14, 44, 133\})$ dans le texte de le figure 7 sont :

$$\begin{aligned} \mathit{SuccCh}(\mathtt{inst}) = \{&\mathtt{insta}, \mathtt{inste}\} \\ &\text{avec } l(\mathtt{insta}) = (5, \{14, 44\}) \\ &\text{et } l(\mathtt{inste}) = (5, \{133\})\} \end{aligned}$$

Successeurs d'une analogie

Dans un premier temps, nous définissons le successeur gauche d'une analogie graine $A : B :: C : D$ comme l'analogie $A' : B' :: C : D$ telle que A' et B' sont des successeurs des chaînes A et B partageant le dernier symbole. Gauche est ici pour le terme gauche du signe $::$. Formellement :

$$\begin{aligned} &\mathit{SuccGauche}(A : B :: C : D) = \\ &\{A \cdot a : B \cdot a :: C : D \ / \ A \cdot a \sqsubset T \wedge B \cdot a \sqsubset T\} \end{aligned}$$

Par exemple, les successeurs gauches de l'analogie $\mathtt{inst} : \mathtt{inst} :: \mathtt{lik} : \mathtt{lik}$ du texte de la figure 7 sont :

$$\begin{aligned} &\mathit{SuccGauche}(\mathtt{inst} : \mathtt{inst} :: \mathtt{lik} : \mathtt{lik}) = \\ &\{\mathtt{insta} : \mathtt{insta} :: \mathtt{lik} : \mathtt{lik},\ \mathtt{inste} : \mathtt{inste} :: \mathtt{lik} : \mathtt{lik}\} \end{aligned}$$

Il faut noter que les successeurs gauches d'une analogie ne représentent pas l'intégralité des successeurs existants pour cette analogie. L'intégralité des successeurs d'une analogie est énumérée à l'aide des 4 permutations[3] introduites en section 2.2.2 :

$$\begin{aligned} &\mathit{Succ}(A : B :: C : D) = \\ &\quad \{\ p_{xy} \circ \mathit{SuccGauche}(A : B :: C : D) \circ p_{xy}, \\ &\qquad\qquad \forall xy \in \{AB, AC, CD, DB\}\ \} \end{aligned}$$

Par exemple, l'analogie $\mathtt{inst} : \mathtt{inst} :: \mathtt{lik} : \mathtt{lik}$ a pour ensemble de successeurs :

$$\begin{aligned} &\mathit{Succ}(\mathtt{inst} : \mathtt{inst} :: \mathtt{lik} : \mathtt{lik}) = \\ &\{\ \mathtt{insta} : \mathtt{insta} :: \mathtt{lik} : \mathtt{lik}, \mathtt{inste} : \mathtt{inste} :: \mathtt{lik} : \mathtt{lik}, \\ &\quad \mathtt{inst} : \mathtt{inst} :: \mathtt{like} : \mathtt{like}, \mathtt{inst} : \mathtt{inste} :: \mathtt{lik} : \mathtt{like}, \\ &\quad \mathtt{inste} : \mathtt{inst} :: \mathtt{like} : \mathtt{lik}\ \} \end{aligned}$$

3 Puisque les permutations sont idempotentes, il est inutile d'écrire p_{xy}^{-1} à droite ou à gauche de la formule.

2.3.3 *Causes de redondance*

L'énumération des analogies à l'aide de l'opération Succ définie précédemment peut amener à produire la même analogie plusieurs fois. Il existe deux sources de redondances dans la définition précédente.

D'une part, l'application de *Succ* à une analogie peut produire deux formes équivalentes de la même analogie. Nous allons proposer une solution à ce problème en section 2.3.4.

D'autre part, l'application de *Succ* à deux analogies différentes peut produire deux formes équivalentes de la même analogie. Ce problème sera résolu en section 2.3.5.

2.3.4 *Nouvelle définition de* Succ

La première cause de redondance vient de la définition des successeurs donnée en section 2.3.2. Par exemple, dans l'exemple précédent, $\mathtt{inst} : \mathtt{inste} :: \mathtt{lik} : \mathtt{like}$ et $\mathtt{inste} : \mathtt{inst} :: \mathtt{like} : \mathtt{lik}$ sont deux formes de la même analogie. Une procédure automatique doit éliminer l'une d'elle afin d'économiser du temps de calcul.

Afin d'identifier les formes équivalentes d'une analogie, une forme parmi les huit formes équivalentes, appelée *forme canonique* doit être retenue.

En utilisant un ordre total sur les chaînes de caractères noté $\leqslant$, il est facile de prouver que la condition $A \leqslant B$, $B \leqslant C$ et $A \leqslant D$ n'est vérifiée que par une seule analogie sur les huit formes équivalentes. On peut alors la choisir comme représentante de la classe des huit formes équivalentes. En remarquant que les ensembles de positions sont représentables par des listes ordonnées, nous pouvons définir un ordre total entre ensembles de positions équivalent à l'ordre lexicographique des listes sous-jacentes. Ainsi, l'ordre total sur les sous-chaînes du texte T est défini par la relation suivante :

avec $l(A) = (\mathrm{long}_A, \mathrm{pos}_A)$ et $l(B) = (\mathrm{long}_B, \mathrm{pos}_B)$:

$$A \leqslant B \quad \Leftrightarrow \quad \mathrm{long}_A < \mathrm{long}_B \text{ ou } \mathrm{long}_A = \mathrm{long}_B \text{ et } \mathrm{pos}_A \leqslant \mathrm{pos}_B$$

Nous appelons Can la fonction identifiant la forme canonique d'une analogie. Par exemple $\mathtt{Can}(\mathtt{inste} : \mathtt{inst} :: \mathtt{like} : \mathtt{lik}) = \mathtt{lik} : \mathtt{like} :: \mathtt{inst} : \mathtt{inste}$ dans le texte de la figure 7.

Nous pouvons à présent donner une nouvelle définition de Succ à l'aide de Can :

$$\begin{aligned} &\mathit{CanSucc}(A : B :: C : D) = \\ &\quad \{\ \mathit{Can}(x),\ \ \forall x \in \mathit{Succ}(A : B :: C : D)\ \} \end{aligned}$$

À l'aide de cette nouvelle définition, l'ensemble des successeurs de notre exemple précédent $\mathtt{inst} : \mathtt{inst} :: \mathtt{lik} : \mathtt{lik}$ est réduit. Il passe de 5 analogies à 4 analogies canoniques :

$$\{\ \mathtt{lik} : \mathtt{lik} :: \mathtt{insta} : \mathtt{insta}, \mathtt{lik} : \mathtt{lik} :: \mathtt{inste} : \mathtt{inste},\\ \mathtt{like} : \mathtt{like} :: \mathtt{inst} : \mathtt{inst}, \mathtt{lik} : \mathtt{like} :: \mathtt{inst} : \mathtt{inste}\ \}$$

2.3.5 *Utilisation de la programmation dynamique*

Nous avons mentionné l'existence d'une seconde cause de redondance en section 2.3.3. Il s'agit du cas où deux analogies différentes ont une ou plusieurs analogies successeurs équivalentes. Ce cas reste possible même avec les analogies canoniques obtenues par CanSucc. Par exemple, les deux analogies différentes $\epsilon : \epsilon :: \mathtt{ab} : \mathtt{ab}$ et $\mathtt{a} : \mathtt{a} :: \mathtt{a} : \mathtt{a}$ permettent d'obtenir $\mathtt{a} : \mathtt{a} :: \mathtt{ab} : \mathtt{ab}$ par application de CanSucc. Ce cas précis est représenté par les flèches aux corps gras dans la figure 8 qui représente l'espace de recherche du texte $\mathtt{ab}\hookleftarrow\mathtt{c}\hookleftarrow$.

La technique la plus simple pour éliminer cette cause de redondance est d'appliquer une technique de mémoïsation :[4] la programmation dynamique est en effet la réponse habituelle à ce problème classique. Nous avons donc mis en œuvre cette technique dans notre implantation.

2.4 ESPACE DE RECHERCHE ET STRATÉGIES DE PARCOURS

L'énumération exhaustive et sans redite des analogies d'un texte est la fermeture réflexive et transitive de CanSucc, notée CanSucc*. Autrement dit l'énumération se résume à une application itérative de la fonction CanSucc à partir d'analogies graines. L'ensemble des analogies obtenu constitue l'espace de recherche.

2.4.1 *Représentation de l'espace de recherche d'analogies*

Chaque itération permet d'obtenir un ensemble d'analogies à partir d'une analogie : $A_i : B_i :: C_i : D_i \longrightarrow A_{i+1} : B_{i+1} :: C_{i+1} : D_{i+1}$. La somme des longueurs des termes de l'analogie augmente selon la relation $|A_{i+1}| + |B_{i+1}| + |C_{i+1}| + |D_{i+1}| = |A_i| + |B_i| + |C_i| + |D_i| + 2$. Cette propriété permet de représenter l'espace de recherche d'analogies par un graphe orienté acyclique où la profondeur d'une analogie est la somme

4 La mémoïsation est une technique permettant de réduire le temps de calcul au détriment de l'espace mémoire. Une fonction mémoïsée n'est exécutée qu'une fois pour une liste d'arguments donnée et le résultat est stocké dans une structure de données pour les appels de fonction suivants.

des longueurs de ses termes. La figure 8 représente un tel espace de recherche.

De manière générale, la profondeur d'une analogie $A : B :: C : D$ dans l'espace de recherche est :

$$p(A : B :: C : D) = \frac{|A| + |B| + |C| + |D|}{2}$$

Cette valeur est également la longueur du chemin entre $A : B :: C : D$ et l'analogie racine $\epsilon : \epsilon :: \epsilon : \epsilon$.

Par exemple, dans le texte $T = ab \hookleftarrow c \hookleftarrow$ de la figure 8, la profondeur de l'analogie $a : a :: ab : ab$ est :

$$p(a : a :: ab : ab) = 3$$

en effet, les chemins allant de l'analogie racine à $a : a :: ab : ab$ sont formés de trois applications successives de $\mathrm{CanSucc}$. Les deux chemins possibles à partir de la racine sont mis en évidence par des flèches épaisses dans la figure 8.

2.4.2 *Stratégies de parcours*

Dans le but de parcourir l'espace de recherche, deux stratégies sont envisageables : en profondeur d'abord et en largeur d'abord.

Classiquement, l'algorithme de parcours en profondeur d'abord est implanté à l'aide d'une pile, une pile d'analogies ici. De plus, certaines structures de données permettent de mémoïser les analogies déjà parcourues. Nous ne détaillerons pas plus le parcours en profondeur d'abord, car cette stratégie ne permet aucune optimisation particulière.

En revanche, en remarquant que les analogies de profondeur p_{i+1} sont par définition les successeurs directs des analogies p_i, nous pouvons restreindre la mémoïsation des analogies à deux niveaux de profondeur en employant une stratégie de parcours en largeur. Techniquement, il suffit d'employer deux queues dans l'implantation du parcours de l'espace de recherche en largeur d'abord. La première queue contient les analogies de profondeur p_i en cours de traitement. La seconde queue contient les analogies de profondeur p_{i+1} pour traitement ultérieur. Lorsque la première queue est vide, le compteur de profondeur i est incrémenté et les queues sont interverties.

2.4.3 *Validation des stratégies de parcours*

Nous avons implanté les parcours en profondeur d'abord et en largeur d'abord. Nous étudions maintenant l'efficacité en espace de chacune de ces stratégies de parcours de façon expérimentale. Dans ce but, nous

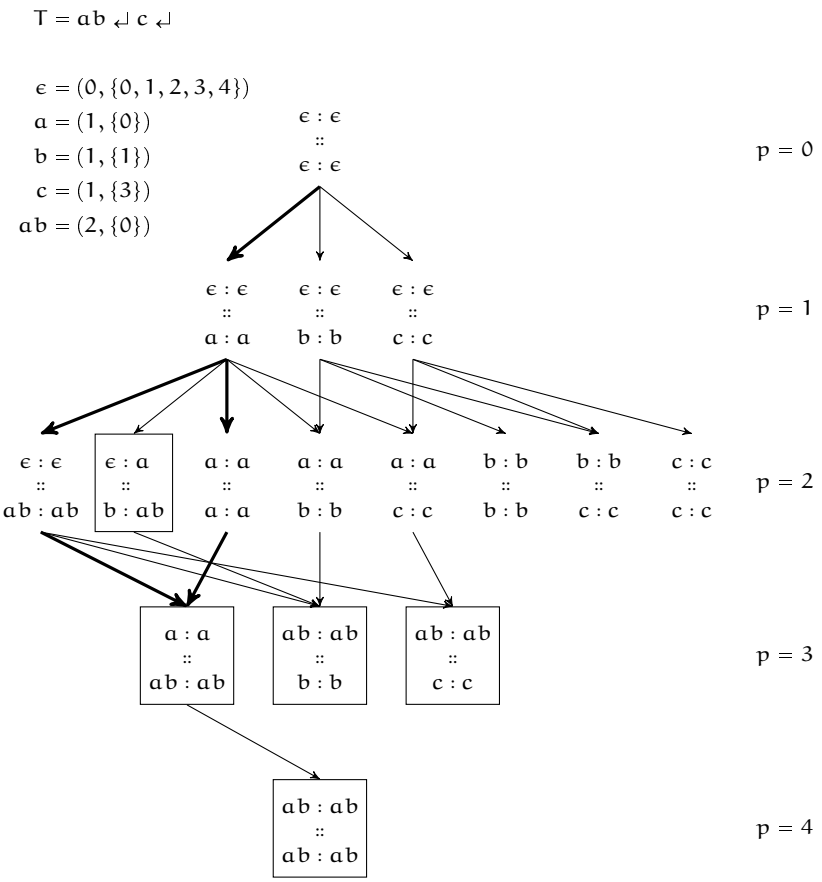


FIGURE 8 : Représentation d'un espace de recherche d'analogies à l'aide d'un graphe orienté acyclique. Cette figure illustre l'espace de recherche parcouru lors de l'énumération exhaustive des analogies du texte ab ↵ c ↵ (nous représentons le symbole de fin de ligne par ↵). Les nœuds sont des analogies canoniques et les arêtes connectent les analogies selon la relation de successeur canonique. Dans cet exemple, seule l'analogie $\epsilon : a :: b : ab$ est non-triviale. Les deux chemins connectant l'analogie racine $\epsilon : \epsilon :: \epsilon : \epsilon$ à l'analogie $a : a :: ab : ab$ sont mis en évidence. Pour la signification des analogies encadrées, voir la section 2.5.

utilisons le petit texte suivant, consistant en trois phrases extraites du BTEC anglais :

```
Please input your pin number.↵
This is my first time diving.↵
I want to have a tight permanent.↵
```

Le parcours de l'espace de recherche est initialisé par la seule analogie $\epsilon : \epsilon :: \epsilon : \epsilon$. Par la suite, cette initialisation est appelée *initialisation par l'analogie de chaînes vides*. Lors du traitement du texte précédent,

le parcours en profondeur d'abord nécessite le stockage de 1 607 436 analogies pour la mémoïsation. Pour le même traitement, le parcours en largeur d'abord implanté à l'aide des deux queues n'a gardé que 74 547 analogies soit $\frac{1}{22^e}$ de l'espace du parcours en profondeur d'abord. De cette expérience, nous pouvons conclure que le parcours en largeur d'abord économise 95 % de mémoire par rapport à un parcours en profondeur d'abord. Des expériences sur de plus longs textes ont mises en évidence des économies encore plus importantes.

2.5 STRATÉGIE D'INITIALISATION

2.5.1 *Initialisation par l'analogie de chaînes vides*

Dans l'expérience précédente, nous avons employé l'analogie de chaînes vides $\epsilon : \epsilon :: \epsilon : \epsilon$ afin d'initialiser le parcours. Quel que soit le texte d'entrée, cette analogie est garantie n'être le successeur d'aucune autre analogie (elle n'a pas de préfixes propres), c'est pourquoi cette analogie est un choix évident pour initialiser le parcours de l'espace de recherche.

2.5.2 *Initialisation par les premières analogies non-triviales*

Bien que l'initialisation par l'analogie de chaînes vides garantisse le bon déroulement du parcours, nous remarquons que cette stratégie implique le parcours de nombreuses analogies triviales avant d'obtenir les premières analogies non-triviales. Par définition, les premières analogies non-triviales sont de la forme $A : A \cdot a :: B : B \cdot a$ avec $A \neq B$, $A \neq \epsilon$ et $B \neq \epsilon$. Tous les couples (A, B) vérifiant ces conditions peuvent être énumérés efficacement à l'aide d'un tableau de suffixes (Manber et Myers, 1990) et devenir les graines d'une initialisation par premières analogies non-triviales. Les analogies encadrées dans la figure 8 sont issues de ce calcul et représentent un tiers de l'espace de recherche.

2.5.3 *Optimisation obtenue par l'initialisation par les premières analogies non-triviales*

Nous vérifions maintenant le bénéfice de la stratégie d'initialisation par les premières analogies non-triviales. Nous employons quatre ensembles de phrases extraits de la partie anglaise du corpus IWSLT 2008 (Paul, 2008). Chaque ensemble de phrases contient 10 phrases de longueurs similaires : 10, 20, 30 et 40 caractères (± 10 %). Dans cette expérience, les sous-chaînes ne peuvent être à cheval sur deux lignes, autrement dit les analogies contenant un caractère fin de ligne sont écartées. Cette

contrainte permet d'obtenir des analogies entre sous-chaînes de phrases. Le corpus de 10 phrases de 30 caractères est reproduit ci-dessous.

```
Please input your pin number.↵
This is my first time diving.↵
I want to have a tight permanent.↵
Let me look at the receipt, then.↵
Please open your mouth wide.↵
Where's the nearest perfumery?↵
Can you sew on this button?↵
Go left at the third corner.↵
Would you clean these clothes?↵
You break it, you bought it.↵
```

Pour chaque corpus, et chaque stratégie d'initialisation, nous mesurons le temps nécessaire à l'énumération exhaustive des analogies à l'aide du parcours en largeur d'abord. Le traitement est effectué au niveau des chaînes de caractères.

La réduction relative du temps de traitement obtenue par la stratégie d'initialisation par les premières analogies non-triviales vis à vis de l'initialisation par analogies de chaînes vides est présentée dans le tableau ci-dessous.

Longueur des lignes en caractères (± 10 %)	10	20	30	40
Temps de traitement avec l'initialisation par l'analogie de chaînes vides (s)	9,93	173,43	1 201,95	6 673,41
Temps de traitement avec l'initialisation par les premières analogies non-triviales (s)	3,07	68,23	457,87	1 791,32
Réduction du temps de traitement	69,1 %	60,7 %	61,9 %	**73,2 %**

L'emploi de la stratégie d'initialisation par les premières analogies non-triviales permet donc d'économiser deux tiers du temps de traitement. Le graphe de la figure 9 montre qu'une réduction plus importante du temps de traitement peut être attendue pour des textes plus longs.

2.6 TRAVAUX EN RELATION AVEC L'ÉNUMÉRATION EXHAUSTIVE DES ANALOGIES

Lepage (2004b) mentionne que l'énumération exhaustive, même avec l'aide de méthodes très rapides de compilation de traits, des analogies entre phrases entières d'un corpus d'environ 100 000 phrases prend énormément de temps. En japonais, Lepage et coll. (2007) ont montré qu'il y a des milliers de fois plus d'analogies entre chunks qu'entre les phrases entières dont les chunks ont été extraits. Une étude similaire

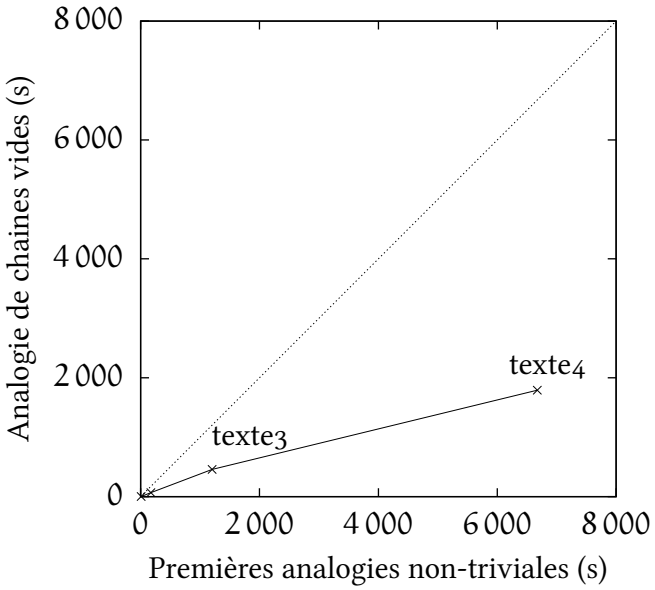


FIGURE 9 : Comparaison du temps de traitement de l'initialisation par l'analogie de chaînes vides (en ordonnée) avec l'initialisation par premières analogies non-triviales (en abscisses) pour différents textes. Les durées sont données en secondes. La courbe reste sous la première diagonale (tracée en pointillés), ce qui indique un temps de traitement inférieur avec l'initialisation par premières analogies non-triviales.

de Takeya et Lepage (2011) mène aux même conclusions pour les onze langues européennes du corpus Europarl. À terme, le travail présenté ici devrait donc permettre d'énumérer les analogies entre chunks en deux langues afin d'améliorer la traduction des chunks dans les systèmes de traduction automatique par analogie. Langlais et Yvon (2008) rapportent une méthode rapide pour l'énumération exhaustive d'analogies entre mots. Les trois travaux précédents considèrent un problème différent de celui qui nous a intéressé ici. L'énumération exhaustive des analogies entre sous-chaînes d'un texte est un problème plus général qui permet nécessairement l'énumération de toutes analogies entre chunks et plus encore.

2.7 CONCLUSION INTERMÉDIAIRE

Dans cette section, nous avons présenté une méthode permettant d'énumérer exhaustivement et sans répétitions toutes les analogies d'un texte. Notre méthode se fonde sur une énumération itérative permettant un parcours exhaustif de l'espace de recherche. L'utilisation de la programmation dynamique permet de réduire l'empreinte mémoire.

Nous avons implanté une méthode de base utilisant une stratégie de parcours en profondeur d'abord à partir de l'analogie de chaînes vides

$\epsilon : \epsilon :: \epsilon : \epsilon$. Nous avons proposé deux optimisations de cette méthode de base. D'une part, un parcours en largeur d'abord permet de réduire la quantité de mémoire utilisée de plus de 95 %. D'autre part, une initialisation à partir des premières analogies non triviales permet de réduire le temps de traitement de plus de 70 %.

Nous sommes donc en mesure d'extraire les analogies d'un texte et d'en déduire des patrons d'analogies.

2.8 PATRONS D'ANALOGIES PRÉSENTES DANS UN CORPUS DE PHRASES FRANÇAISES ET ANGLAISES

Nous appliquons la méthode d'extraction d'analogies présentée précédemment aux corpus monolingues français et anglais du BTEC. Les termes des analogies sont restreints aux n-grammes de mots présents dans les phrases. Nous cherchons donc à observer la structure des analogies dont les termes sont des n-grammes de même ordre, d'abord des bigrammes puis des trigrammes.

Une fois les analogies extraites, nous réunissons les instances d'analogies par patrons afin d'observer les analogies les plus fréquentes.

2.8.1 *Extraction de patrons d'analogies*

L'extraction de patrons d'analogies consiste à extraire une forme symbolique à partir de plusieurs analogies. Par exemple, étant donné les analogies suivantes, extraites d'un corpus de phrases françaises (BTEC, 901 phrases) :

P_1 : sel , : pain , :: de sel : de pain

P_2 : cuisine . : moutarde . :: de cuisine : de moutarde

P_3 : l'huile de : de l'huile :: temps de : de temps

P_4 : pommes de : de pommes :: temps de : de temps

P_5 : , ça : que ça :: ça , : ça que

Les phrases P_1 et P_2 ont le même patron $ab : cb :: da : dc$, de la même manière, les phrases P_3, P_4 et P_5 partagent le même patron $ab : ba :: cb : bc$. Dans le cas de la phrase P_5, c'est la forme équivalente

, ça : ça , :: que ça : ça que

qui respecte le patron $ab : ba :: cb : bc$.

Les patrons d'analogies sont extraits par renommage des mots dans l'ordre de lecture de l'analogie. Par exemple, le patron de l'analogie sel , : pain , :: de sel : de pain permet d'extraire la règle de renommage suivante :

$$\begin{aligned} \text{sel} &\rightarrow a \\ , &\rightarrow b \\ \text{pain} &\rightarrow c \\ \text{de} &\rightarrow d \end{aligned}$$

En suivant cette règle, le patron $ab : cb :: da : dc$ est construit.

Afin d'obtenir un patron sous une forme canonique (voir section 2.3.4), cette fonction de réécriture est effectuée sur toutes les formes équivalentes de l'analogie puis la plus petite est considérée comme le patron de l'analogie.

2.8.2 *Extraction et décompte de patrons d'analogies dans un corpus de phrases françaises et anglaises*

Afin de faire une étude statistique des patrons d'analogies en français et anglais, nous faisons une expérience sur un sous-corpus de 901 phrases françaises et anglaises du BTEC.

Dans un premier temps, nous énumérons les analogies contenues dans chaque corpus de phrases. Nous limitons l'énumération aux analogies dont tous les termes sont des n-grammes de mots. Les analogies triviales (voir section 2.2.1 pour la définition des analogies triviales) de la forme $A : A :: B : B$ sont exclues, ainsi que les analogies ne respectant pas la contrainte sur les distances d'édition $d(A, B) = d(C, D)$ et $d(A, C) = d(B, D)$. De cette manière, les analogies énumérées respectent la formalisation de Lepage (voir section 2.2.2). Nous effectuons l'énumération pour les tailles de n-grammes 1, 2, 3 ... 8 dans les deux langues.

Dans un second temps, les patrons d'analogies sont extraits des analogies obtenues précédemment pour chaque taille de n-gramme et chaque langue. Nous effectuons un décompte des patrons différents. Le résultat obtenu est présenté dans la figure 10. Une courbe est tracée par langue avec la taille des n-grammes en abscisse et le nombre de patrons en ordonnée.

Aucun patron n'est extrait à partir des analogies d'unigrammes puisqu'elles sont toutes triviales et donc exclues du traitement. De la taille 2 à la taille 5, le nombre de patrons d'analogies croît quasi-proportionnellement à la taille des n-grammes. Pour les tailles supérieures à 6, le nombre de patrons d'analogies décroît dans le cas de l'anglais et continue à croître puis décroît pour le français. Ce phénomène est difficilement explicable en l'état, il faudrait rapprocher ces nombres de patrons à d'autres décomptes comme la taille du vocabulaire, le décompte de n-grammes différents dans chaque langue ou la distribution des longueurs de phrases dans chaque langue.

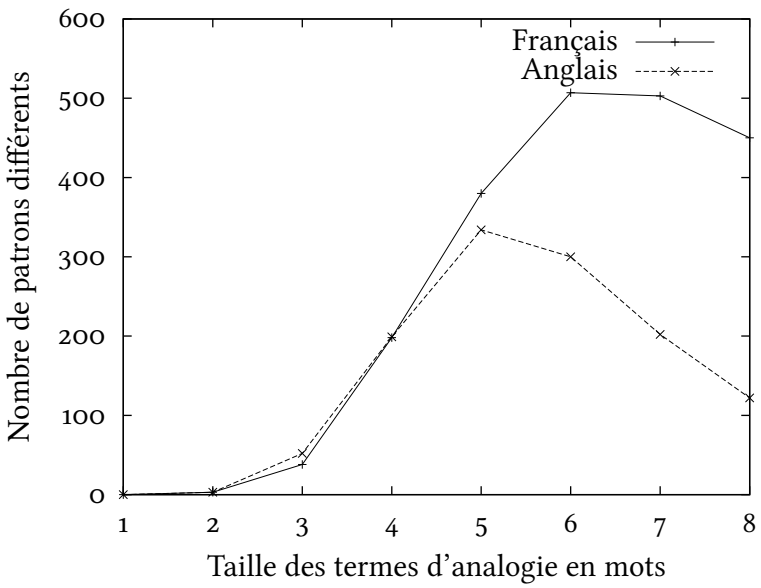


FIGURE 10 : Décompte des patrons d'analogies de n-grammes selon la taille des n-grammes.

Les tableaux 3 et 5 donnent des exemples d'analogies extraites du texte français pour chaque patron d'analogie constaté pour les tailles de termes d'analogies 2 et 3. Quelques exemples d'analogies sont présentés pour chaque patron d'analogie en troisième colonne. Le tableau 4 détaille les mêmes données concernant les analogies présentes dans le corpus de phrases anglaises pour les analogies entre bigrammes.

TABLEAU 3 : Patrons d'analogie de bigrammes en français.

Nombre d'analogies	Patron d'analogie	Exemples
7 595	a b : a c :: d b : d c	place ? : place , :: café ? : café , Excusez-moi , : Excusez-moi . :: frit , : frit . Bonjour , : Bonjour . :: poisson , : poisson . bon , : bon et :: frite , : frite et
5 482	a b : c b :: d a : d c	sel , : pain , :: de sel : de pain du thé : un thé :: Juste du : Juste un , à : ou à :: Grand , : Grand ou expresso . : yaourt . :: un expresso : un yaourt
30	a b : b a :: c b : b c	moyenne , : ketchup , :: , moyenne : , ketchup deux , : , deux :: ça , : , ça pommes de : de pommes :: temps de : de temps , ça : que ça :: ça , : ça que

TABLEAU 4 : Patrons d'analogie de bigrammes en anglais.

Nombre d'analogies	Patron d'analogie	Exemples
8 904	a b : c b :: d a : d c	ketchup . : yogurt . :: , ketchup : , yogurt grapes . : ketchup . :: some grapes : some ketchup flour . : mustard . :: some flour : some mustard bread ? : takeout ? :: for bread : for takeout
8 903	a b : a c :: d b : d c	available ? : available . :: all ? : all . a coffee : a sandwich :: and coffee : and sandwich a banana : a sugar :: some banana : some sugar change . : change is :: it . : it is
144	a b : a c :: b a : c a	with ham : with bacon :: ham with : bacon with or bacon : bacon or :: or ham : ham or , bread : , yogurt :: bread , : yogurt , , milk : , onions :: milk , : onions ,

TABLEAU 5 : Patrons d'analogie de trigrammes en français.

Nombre d'analogies	Patron d'analogie	Exemples
4 283	a b c : a b d :: e f c : e f d	du lard ? : du lard et :: la mayonnaise ? : la mayonnaise et du beurre . : du beurre ? :: un hot-dog . : un hot-dog ? coca moyen , : coca moyen . :: du raisin , : du raisin . café noir , : café noir . :: un coca , : un coca .
3 084	a b c : a d e :: b c f : d e f	et deux cocas : et des frites :: deux cocas . : des frites . avec du fromage : avec vos oeufs :: du fromage ? : vos oeufs ? voudrais des pommes : voudrais du thé :: des pommes . : du thé . Voulez-vous de l'oignon : Voulez-vous du lard :: de l'oignon ? : du lard ? voudrais du vin : voudrais un chocolat :: du vin , : un chocolat ,
1 004	a b c : d b c :: e f a : e f d	lard et des : jambon et des :: Voulez-vous du lard : Voulez-vous du jambon oignons , s'il : épinards , s'il :: et des oignons : et des épinards , de la : avec de la :: du thé , : du thé avec café , s'il : coca , s'il :: et un café : et un coca , une grande : et une grande :: un hamburger , : un hamburger et
921	a b c : a b d :: b c e : b d e	voudrais du sucre : voudrais du beurre :: du sucre . : du beurre . voudrais du jambon : voudrais du poisson :: du jambon et : du poisson et sac de chips : sac de popcorn :: de chips , : de popcorn , de la confiture : de la mayonnaise :: la confiture ? : la mayonnaise ? vais prendre du : vais prendre un :: prendre du café : prendre un café
874	a b c : a e c :: b c d : e c d	des frites , : frites , s'il :: des bananes , : bananes , s'il Un cheeseburger , : cheeseburger , s'il :: Un hamburger , : hamburger , s'il du fromage , : fromage , s'il :: du sel , : sel , s'il de seltz , : seltz , s'il :: de thé , : thé , s'il la bière , : bière , s'il :: la table , : table , s'il
824	a b c : a d e :: f b c : f d e	prendre un sandwich : prendre du pain :: avoir un sandwich : avoir du pain Un décaféiné , : Un jus d'orange :: un décaféiné , : un jus d'orange et de l'eau : et des oeufs :: voudrais de l'eau : voudrais des oeufs cheeseburgers , s'il : cheeseburgers et un :: coca , s'il : coca et un cheeseburgers , s'il : cheeseburgers et un :: hot-dogs , s'il : hot-dogs et un
455	a b c : a d c :: b e f : d e f	la crème . : la bière . :: crème , s'il : bière , s'il moutarde , des : moutarde et des :: , un muffin : et un muffin de ceux-là . : de frites . :: ceux-là , s'il : frites , s'il du jambon ou : du ketchup ou :: jambon , s'il : ketchup , s'il des bonbons . : des oranges . :: bonbons , s'il : oranges , s'il
310	a b c : a b d :: a e c : a e d	des oignons ? : des oignons , :: des frites ? : des frites , du jambon avec : du jambon ou :: du lard avec : du lard ou du café , : du café . :: du pain , : du pain . un café , : un café ? :: un couvercle , : un couvercle ? du beurre , : du beurre . :: du thé , : du thé .
264	a b c : a d c :: e f b : e f d	moutarde et des : moutarde , des :: la mayonnaise et : la mayonnaise , le poivre , : le sel , :: Avez-vous du poivre : Avez-vous du sel moutarde , des : moutarde et des :: du vin , : du vin et hamburgers , un : hamburgers et un :: Du ketchup , : Du ketchup et hamburgers , un : hamburgers et un :: grande frite , : grande frite et
223	a b c : a b d :: e b c : e b d	voudrais du pain : voudrais du beurre :: avoir du pain : avoir du beurre Un hamburger , : Un hamburger et :: un hamburger , : un hamburger et frites , du : frites , s'il :: hamburger , du : hamburger , s'il À emporter , : À emporter ? :: à emporter , : à emporter ? Pour emporter . : Pour emporter ? :: à emporter . : à emporter ?
150	a b c : c a b :: d e c : c d e	eh bien , : des oignons , :: , eh bien : , des oignons du jambon , : , du jambon :: à emporter , : , à emporter des frites , : , des frites :: un muffin , : , un muffin au seigle , : , au seigle :: eh bien , : , eh bien au seigle , : , au seigle :: des frites , : , des frites

Tableau 5 : Patrons d'analogie de trigrammes en français (suite).

Nombre d'analogies	Patron d'analogie	Exemples
89	a b c : a d c :: e b f : e d f	un coca , : un cheeseburger , :: Un coca et : Un cheeseburger et de ketchup , : de sel , :: du ketchup ? : du sel ? de café , : de thé , :: du café glacé : du thé glacé du ketchup , : du sel , :: le ketchup et : le sel et Un cheeseburger et : Un coca et :: un cheeseburger . : un coca .
72	a b c : a d c :: e b c : e d c	le sel , : le poivre , :: du sel , : du poivre , de sel , : de thé , :: du sel , : du thé , Un chocolat , : Un coca , :: un chocolat , : un coca , Un coca , : Un hamburger , :: petit coca , : petit hamburger , Un déca , : Un décaféiné , :: un déca , : un décaféiné ,
37	a b c : a b e :: c a d : e a d	et un hamburger : hamburger et des :: et un cheeseburger : cheeseburger et des du lard avec : avec du ketchup :: du lard et : et du ketchup , du jambon : jambon , s'il :: , du rôti : rôti , s'il du jambon avec : avec du ketchup :: du jambon et : et du ketchup un cheeseburger , : , un muffin :: un cheeseburger et : et un muffin
35	a b c : a b d :: e a c : e a d	oeufs brouillés ? : oeufs brouillés , :: des oeufs ? : des oeufs , un petit hamburger : un petit jus :: et un hamburger : et un jus coca moyen , : coca moyen . :: petit coca , : petit coca . un grand cheeseburger : un grand hamburger :: et un cheeseburger : et un hamburger un petit café : un petit jus :: voudrais un café : voudrais un jus
35	a b c : a e c :: b d a : e d a	du thé , : thé avec du :: du café , : café avec du , du jus : du sel , :: , le jus : le sel , de ketchup ? : ketchup et de :: de l'oignon ? : l'oignon et de un cheeseburger , : cheeseburger et un :: un menu , : menu et un un cheeseburger , : cheeseburger et un :: un coca , : coca et un
27	a b c : a b e :: c d b : e d b	et du thé : thé avec du :: et du café : café avec du moutarde et de : de jambon et :: moutarde et du : du jambon et hamburger et de : de jambon et :: hamburger et des : des jambon et beaucoup de ketchup : ketchup et de :: beaucoup de moutarde : moutarde et de avoir un hamburger : hamburger et un :: avoir un hot-dog : hot-dog et un
27	a b c : a d e :: b c d : d e d	voudrais deux boules : voudrais de l'huile :: deux boules de : de l'huile de voudrais un peu : un peu de :: voudrais de l'eau : de l'eau de voudrais des pommes : des pommes de :: voudrais de l'huile : de l'huile de voudrais des sachets : des sachets de :: voudrais de l'huile : de l'huile de voudrais du rôti : du rôti de :: voudrais de l'huile : de l'huile de
25	a b c : a d b :: e b c : e d b	frites s'il vous : frites , s'il :: carte s'il vous : carte , s'il carte s'il vous : carte , s'il :: table s'il vous : table , s'il Un coca et : Un grand coca :: un coca et : un grand coca coca s'il vous : coca , s'il :: table s'il vous : table , s'il l'eau , s'il : l'eau chaude , :: pommes , s'il : pommes chaude ,
24	a b c : b d c :: e a b : e b d	petites frites , : frites moyennes , :: Deux petites frites : Deux frites moyennes grand coca , : coca moyen , :: Un grand coca : Un coca moyen petit café , : café noir , :: un petit café : un café noir petit café , : café crème , :: un petit café : un café crème peu de moutarde : de la moutarde :: avec peu de : avec de la
23	a b c : a c d :: e b f : e f d	un grand cheeseburger : un cheeseburger et :: Un grand coca : Un coca et peux en avoir : peux avoir un :: vais en prendre : vais prendre un Un petit coca : Un coca , :: un petit hamburger : un hamburger , peux en avoir : peux avoir du :: vais en prendre : vais prendre du Un petit hot-dog : Un hot-dog , :: un petit hamburger : un hamburger ,

TABLEAU 5 : Patrons d'analogie de trigrammes en français (suite).

Nombre d'analogies	Patron d'analogie	Exemples
21	a b c : a b d :: c e a : d e a	un grand hamburger : un grand coca :: hamburger et un : coca et un de la moutarde : de la tomate :: moutarde et de : tomate et de un grand cheeseburger : un grand coca :: cheeseburger et un : coca et un d'orange et un : d'orange et votre :: un jus d'orange : votre jus d'orange de la Budweiser : de la mayonnaise :: Budweiser et de : mayonnaise et de
17	a b c : a b d :: a c e : a d e	Un petit hot-dog : Un petit coca :: Un hot-dog , : Un coca , un grand cheeseburger : un grand coca :: un cheeseburger et : un coca et un petit café : un petit coca :: un café , : un coca , un grand coca : un grand hamburger :: un coca , : un hamburger , un grand cheeseburger : un grand hamburger :: un cheeseburger et : un hamburger et
17	a b c : a b d :: e c a : e d a	un cheeseburger et : un cheeseburger , :: hamburgers et un : hamburgers , un et un coca : et un hamburger :: Un coca et : Un hamburger et du thé , : du thé avec :: bœuf , du : bœuf avec du du lard , : du lard avec :: bœuf , du : bœuf avec du et un cheeseburger : et un hamburger :: Un cheeseburger et : Un hamburger et
14	a b c : a d c :: c a b : c a d	un muffin , : un coca , :: , un muffin : , un coca du jambon , : , du jambon :: du lait , : , du lait une grande , : , une grande :: une saucisse , : , une saucisse un coca , : , un coca :: un yaourt , : , un yaourt un cheeseburger et : et un cheeseburger :: un coca et : et un coca
14	a b c : c a b :: d b c : c d b	lard ou du : jambon ou du :: du lard ou : du jambon ou ketchup et de : de ketchup et :: terre et de : de terre et gâteau et un : hamburger et un :: un gâteau et : un hamburger et jambon et des : des jambon et :: oeufs et des : des oeufs et cheeseburger et un : hamburger et un :: un cheeseburger et : un hamburger et
13	a b c : b d c :: e a f : e f d	prendre un café : un petit café :: pour prendre le : pour le petit grand coca à : coca , à :: un grand cheeseburger : un cheeseburger , la carte s'il : carte , s'il :: de la tomate : de tomate , la tomate et : tomate , et :: de la vanille : de vanille , grand coca à : coca , à :: un grand hamburger : un hamburger ,
12	a b c : a b d :: c e c : c e d	des pommes de : des pommes , :: de l'eau de : de l'eau , autre chose que : autre chose ? :: que c'est que : que c'est ? du jambon , : du jambon ou :: , moyenne , : , moyenne ou Sur place , : Sur place ou :: , moyenne , : , moyenne ou du lard , : du lard ou :: , moyenne , : , moyenne ou
11	a b c : d a b :: e a c : e d a	un grand coca : et un grand :: , un coca : , et un un grand jus : prendre un grand :: voudrais un jus : voudrais prendre un un petit coca : et un petit :: , un coca : , et un le petit déjeuner : prendre le petit :: pour le déjeuner : pour prendre le un grand cheeseburger : prendre un grand :: voudrais un cheeseburger : voudrais prendre un
8	a b c : a b d :: c a b : d a b	du jambon ou : ou du jambon :: du jambon , : , du jambon ou du jambon : jambon ou du :: ou du lard : lard ou du du lard avec : avec du lard :: du lard ou : ou du lard des oeufs , : , des oeufs :: des oeufs et : et des oeufs et un cheeseburger : cheeseburger et un :: et un hamburger : hamburger et un
5	a b c : d b e :: f a c : f d e	seigle , ou : lait , s'il :: au seigle ou : au lait s'il café noir , : pain noir . :: du café , : du pain . vais prendre le : voudrais prendre un :: Je vais le : Je voudrais un vais prendre le : voudrais prendre un :: je vais le : je voudrais un carte , s'il : tomate , et :: la carte s'il : la tomate et

TABLEAU 5 : Patrons d'analogie de trigrammes en français (suite).

Nombre d'analogies	Patron d'analogie	Exemples
5	a b c : b a b :: d e c : b d e	moyenne , ou : du jambon ou :: , moyenne , : , du jambon cafétéria la plus : la cafétéria la :: moutarde en plus : la moutarde en moyenne , ou : , moyenne , :: au seigle ou : , au seigle moyenne , ou : , moyenne , :: en gaufre ou : , en gaufre moyenne , ou : , moyenne , :: un sandwich ou : , un sandwich
3	a b c : a d e :: c b c : c d e	des pommes de : des jambon et :: de pommes de : de jambon et des pommes de : de pommes de :: des frites . : de frites . des pommes de : de pommes de :: des coquillages sur : de coquillages sur
2	a b c : d b c :: e a e : e d e	pommes chaude , : l'eau chaude , :: de pommes de : de l'eau de l'eau , s'il : pommes , s'il :: de l'eau de : de pommes de
2	a b c : a e c :: b d c : e d c	du pain . : pain noir . :: du café . : café noir . au chocolat , : chocolat chaud , :: au lait , : lait chaud ,
1	a b c : a b d :: b c b : b d b	voudrais de l'huile : voudrais de l'eau :: de l'huile de : de l'eau de
1	a b c : b a e :: d b c : b d e	moyen , s'il : ketchup , s'il :: , moyen ou : , ketchup ou
1	a b c : a d b :: b e c : b d e	coca , à : coca moyen , :: , ou à : , moyen ou

2.9 CONCLUSION

Dans ce chapitre, nous avons présenté une méthode permettant l'énumération de toutes les analogies d'un texte. Nous avons aussi étudié les analogies de bigrammes et de trigrammes extraites à l'aide de cette méthode dans un corpus français et anglais. Notre méthode d'énumération présente un intérêt allant au delà de notre préoccupation initiale (voir section 2.6). Nous allons maintenant étudier en détail les patrons d'analogies entre trigrammes les plus fréquents obtenus à la section 2.8. Cette étude va nous permettre de répondre à la question suivante : tout mot d'un texte est-il couvert par un n-gramme entrant dans une analogie entre n-grammes avec n fixé ?

3

STRUCTURE DES TRIGRAMMES INCONNUS

En appliquant l'outil d'énumération des analogies sur des textes en plusieurs langues, nous avons été en mesure de déduire des régularités. Nous allons maintenant nous focaliser sur les trigrammes de mots et les analogies entre eux. Nous allons distinguer certains patrons d'analogies entre trigrammes de mots à partir de leurs fréquences d'apparition. Nous allons montrer que, dans leur grande majorité, les trigrammes inconnus sont analogues aux trigrammes hapax : leurs structures et leurs effectifs sont semblables. Les résultats présentés dans ce chapitre ont fait l'objet d'une publication (Gosme et Lepage, 2011).

3.1 INTRODUCTION

Dans le chapitre précédent, nous avons élaboré un outil d'énumération des analogies entre sous-chaînes d'un texte. Cet outil nous a permis d'extraire par filtrage les patrons d'analogies entre n-grammes de mots pour n=2 et n=3. Dans ce chapitre, nous étudions les propriétés des termes trigrammes des analogies. Nous focalisons cette étude sur la fréquence d'apparition des trigrammes et montrons comment l'analogie permet de reconstruire des trigrammes inconnus. Ce travail nous permettra de définir une technique de lissage par analogie dans le chapitre suivant.

En guise d'illustration, dans une de nos expériences préliminaires, le trigramme de mots *opportunité de servir* était un trigramme de notre jeu de test absent du corpus d'entraînement. Il se trouvait que ce trigramme pouvait être reconstruit par analogie à l'aide de trois trigrammes du corpus d'entraînement de la manière suivante :

$$\textit{opportunité de servir} : \textit{opportunité de modifier} :: \textit{qui pourrait servir} : \textit{qui pourrait modifier}$$

La ligne précédente se lit ainsi : le trigramme inconnu *opportunité de servir* est au trigramme connu *opportunité de modifier* ce qu'un autre trigramme connu, *qui pourrait servir*, est à un dernier trigramme connu, *qui pourrait modifier*. Les différents éléments du trigramme inconnu sont obtenus par similarité avec le second et le troisième trigrammes (*opportunité de* et *servir*) et peuvent être assemblés par différence avec le quatrième trigramme. En plus de permettre la reconstruction, les trois trigrammes ci-dessus étaient tous hapax dans le corpus d'entraînement.

La définition de l'analogie que nous utilisons dans ce travail est celle de Lepage (voir définition 2 du Chapitre 2). Nous l'appliquons aux trigrammes de mots. Un quadruplet de trigrammes de mots A, B, C et D est une analogie lorsque les contraintes suivantes sont vérifiées :

$$\begin{cases} d(A,B) &= d(C,D) \\ d(A,C) &= d(B,D) \\ \forall m, |A|_m - |B|_m &= |C|_m - |D|_m \end{cases}$$

Ici, $|A|_m$ est le nombre d'occurrences du mot m dans le trigramme A.

En reprenant l'exemple précédent :

$$\begin{aligned} A &= \textit{opportunité de servir} \\ B &= \textit{opportunité de modifier} \\ C &= \textit{qui pourrait servir} \\ D &= \textit{qui pourrait modifier} \end{aligned}$$

on peut vérifier que $d(A,B) = d(C,D) = 2$ et $d(A,C) = d(B,D) = 4$. La relation entre nombres d'occurrences est vérifiée pour chaque mot :

mot m	$\lvert A\rvert_m - \lvert B\rvert_m = \lvert C\rvert_m - \lvert D\rvert_m$
opportunité	$1 - 1 = 0 - 0$
de	$1 - 1 = 0 - 0$
servir	$1 - 0 = 1 - 0$
modifier	$0 - 1 = 0 - 1$
qui	$0 - 0 = 1 - 1$
pourrait	$0 - 0 = 1 - 1$

Le bon sens veut que les trigrammes inconnus apparaissant dans un jeu de test qui peuvent être reconstruits par analogie avec des trigrammes d'un corpus d'entraînement soient considérés plus sûrs que ceux qui ne peuvent pas l'être. Une technique de lissage basée sur de simples décomptes devrait donc donner une plus forte ré-estimation aux trigrammes pouvant être reconstruits par analogie et une plus faible ré-estimation aux autres. Si, en plus, les trigrammes reconstruits le sont à l'aide de trigrammes hapax, la ré-estimation de leur effectif devrait être proche de 1 puisqu'ils sont alors proches structurellement des trigrammes hapax.

Nous allons vérifier l'hypothèse que les trigrammes inconnus sont structurellement analogues aux trigrammes hapax. Des expériences successives menées sur quatre langues européennes vont confirmer, les unes après les autres, cette hypothèse. Dans le chapitre suivant, nous allons présenter une technique de lissage reposant sur cette propriété, et directement inspirée des techniques élémentaires de Lidstone et de Laplace. Des comparaisons effectuées avec trois autres techniques de lissage classiques sur onze langues européennes vont montrer son efficacité, voire sa supériorité.

STRUCTURE DES TRIGRAMMES INCONNUS D'UN JEU DE TEST

Nous menons des expériences sur les quatre langues suivantes : l'anglais, le français, l'allemand et le finnois. Ces langues ont été choisies pour leurs différentes richesses morphologiques. Sur une échelle croissante, on peut en effet placer successivement l'anglais, le français, l'allemand puis le finnois qui a la morphologie la plus riche.

Le corpus Europarl (Koehn, 2005) offre des textes alignés dans ces quatre langues,[1] ce qui permet de mener des expériences véritablement comparables. Pour les expériences de cette section, de l'ensemble de toutes les phrases correspondantes dans toutes les langues, nous avons extrait aléatoirement 100 000 phrases. Parmi elles, 90 000 phrases ont été sélectionnées aléatoirement, les mêmes dans toutes les langues, pour

1 http://www.statmt.org/europarl

servir de corpus d'entraînement. Le jeu de test est constitué des 10 000 phrases restantes.

3.2 PROPORTION DE TRIGRAMMES DE MOTS INCONNUS RECONSTRUITS

Pour vérifier dans quelle mesure les trigrammes inconnus d'un jeu de test peuvent être reconstruits par analogie, nous effectuons une première série d'expériences par validation croisée. Nous comptons simplement le nombre de trigrammes inconnus du jeu de test reconstructibles par analogie à l'aide de trois trigrammes du corpus d'entraînement. Dès lors que la reconstruction est possible, le processus est interrompu pour ce trigramme.

Les résultats obtenus, reportés dans le tableau 6, indiquent que la proportion de trigrammes inconnus dans le jeu de test reconstructibles par analogie avec trois autres trigrammes du corpus d'entraînement est supérieure à 80 % en anglais et en français et supérieure à 70 % en allemand. Cette proportion, appelée μ ici, est donc importante. Elle est prise par rapport à l'ensemble des trigrammes inconnus, dont la proportion relativement à l'ensemble des trigrammes du jeu de test est elle appelée λ. Des expériences non rapportées ici montrent qu'en augmentant la taille des données, les valeurs de λ baissent tandis que les valeurs de μ augmentent. Les paramètres λ et μ seront exploités dans la section 4.3.1.

TABLEAU 6 : Nombre de trigrammes inconnus dans le jeu de test et proportion de trigrammes inconnus reconstructibles par analogie à l'aide de trois trigrammes du corpus d'entraînement.

	Trigrammes inconnus	
	du jeu de test (λ)	reconstruits (μ)
anglais	114,566 (60,04 %)	83,67 %
français	116,922 (57,81 %)	81,87 %
allemand	140,226 (68,97 %)	72,14 %
finnois	132,931 (83,33 %)	44,93 %

En finnois, la proportion de trigrammes inconnus reconstruits est faible avec seulement 45 %. Cette faible valeur est certainement explicable par la richesse morphologique de cette langue et donc l'absence relative de mots-fonctions permettant plus de commutations de mots dans les trigrammes. Nous pouvons dès à présent nous attendre à des résultats différents en finnois dans toute la suite de notre étude.

3.3 PATRONS D'ANALOGIE LES PLUS FRÉQUENTS

Un trigramme de mots donné peut être obtenu par analogie à l'aide d'autres trigrammes de plusieurs façons. Par exemple, pour le trigramme *opportunité de servir*, on a, entre autres, les deux analogies suivantes qui utilisent des trigrammes différents du corpus d'entraînement et respectent bien la définition de l'analogie vue en introduction :

opportunité de servir : *opportunité de modifier* :: *qui pourrait servir* : *qui pourrait modifier*
opportunité de servir : *opportunité pour dire* :: *de servir le* : *pour dire le*

Ces deux analogies exemplifient deux patrons d'analogie différents donnés dans le tableau 7 et numérotés 1 et 2. Le patron 1 correspond au remplacement du bigramme correspondant à la partie gauche du premier trigramme par un autre bigramme et le remplacement de l'unigramme restant à droite par un autre unigramme : *opportunité de* est remplacé par *qui pourrait*, et *servir* est remplacé par *modifier*. Le patron 2 revient à trouver deux bigrammes dans les mêmes contextes droit et gauche : *de servir* et *pour dire* existent dans les mêmes contextes *opportunité* ∼ et ∼ *le*.

Dans le double but d'énumérer les patrons existant réellement en corpus et d'en déterminer les fréquences d'apparition respectives, nous énumérons simplement toutes les analogies existantes entre trigrammes à partir d'un échantillon aléatoire de 10 000 phrases dans chaque langue. Pour cela, nous avons contraint la méthode d'énumération de toutes les analogies d'un texte présentée dans le chapitre précédant pour n'énumérer que les analogies entre trigrammes de mots. Ensuite, nous regroupons les instances d'analogies obtenues par patron et les comptons.

Le tableau 7 donne les patrons d'analogie listés par ordre décroissant de fréquences pour l'anglais. Un résultat remarquable est que les cinq patrons d'analogie les plus fréquents dans les quatre langues apparaissent dans le même ordre avec des proportions semblables. L'étude similaire décrite en section 2.8.2 sur un corpus de plus petite taille et dans un domaine différent (extrait de 901 phrases du BTEC) a mené aux mêmes patrons d'analogie dans le même ordre. Seul le patron 5 a eu un effectif légèrement supérieur au patron 4.

3.4 PATRONS D'ANALOGIE LES PLUS RENTABLES

Jusqu'à présent, les expériences soutiennent l'hypothèse qu'une grande majorité des trigrammes inconnus peut être reconstruite par analogie à l'aide de trigrammes issus du corpus d'entraînement (section 3.2) ; et nous avons identifié, d'autre part, les patrons d'analogie de trigrammes

TABLEAU 7 : Patrons d'analogie entre trigrammes de mots dans un échantillon anglais de 10 000 phrases du corpus Europarl triés par proportions relatives sur l'ensemble des analogies entre trigrammes. Les symboles utilisés dans l'écriture des patrons d'analogie sont distincts deux à deux. Ces patrons respectent la définition de l'analogie donnée en introduction.

N°	A : B :: C : D	Proportion
1	*a b c* : *a b d* :: *e f c* : *e f d*	12,6 %
2	*a b c* : *a d e* :: *b c f* : *d e f*	9,1 %
3	*a b c* : *d b c* :: *e f a* : *e f d*	3,1 %
4	*a b c* : *a e c* :: *b c d* : *e c d*	2,7 %
5	*a b c* : *a b d* :: *b c e* : *b d e*	2,6 %
6	*a b c* : *a d e* :: *f b c* : *f d e*	2,4 %
7	*a b c* : *a d c* :: *b e f* : *d e f*	1,3 %
8	*a b c* : *a b d* :: *a e c* : *a e d*	0,9 %
⋮	⋮	⋮

de mots les plus fréquents dans un même corpus (section 3.3). L'étape suivante est d'identifier les patrons d'analogie qui permettent de reconstruire le plus de trigrammes inconnus d'un jeu de test à l'aide de trigrammes du corpus d'entraînement, autrement dit, les patrons les plus rentables. À cette fin, nous conduisons une nouvelle série d'expériences. En raison de la lourdeur en temps de calcul, nous limitons notre expérience aux cinq patrons d'analogie les plus fréquents listés dans le tableau 7, et nous procédons de la sorte : pour chaque trigramme inconnu du jeu de test, chaque patron d'analogie est essayé successivement dans l'ordre du tableau 7. Dès lors qu'un patron d'analogie permet de reconstruire le trigramme inconnu en question, nous notons son rang et passons au trigramme inconnu suivant.

Les résultats sont présentés dans les tableaux 8(a)–8(d). Ils montrent les contributions cumulées des patrons d'analogie à la reconstruction des trigrammes inconnus. Le patron 1 contribue seul à la majorité de la reconstruction des trigrammes inconnus : plus de 70 % en anglais, français et allemand, mais seulement 61,5 % pour le finnois. Les patrons 1 et 2 suffisent à reconstruire environ 95 % des trigrammes inconnus en anglais, français et allemand, et presque 90 % en finnois. Pour les quatre langues, les cinq patrons suffisent à reconstruire l'intégralité des trigrammes ; notre restriction se justifie donc a posteriori. Quelques exemples de reconstructions de trigrammes sont donnés dans les figures 11 et 12.

TABLEAU 8 : Cumul des contributions des cinq patrons d'analogie les plus fréquents à la reconstruction des trigrammes inconnus dans les quatre langues étudiées. Les pourcentages présentés sont relatifs au nombre total de trigrammes inconnus.

N° de patron	Trigrammes reconstruits (μ)	Proportion cumulée
1	72 426 (63,22 %)	75,55 %
2	19 952 (17,42 %)	96,37 %
3	3 411 (2,98 %)	99,93 %
4	46 (0,04 %)	99,97 %
5	25 (0,02 %)	100,00 %
Total	95 860 (83,67 %)	100,00 %

(a) Anglais

N° de patron	Trigrammes reconstruits (μ)	Proportion cumulée
1	71,466 (61,98 %)	74,66 %
2	20,475 (17,51 %)	96,05 %
3	3 655 (3,13%)	99,87 %
4	92 (0,08 %)	99,97 %
5	35 (0,03 %)	100,00 %
Total	95 723 (81,87%)	100,00 %

(b) Français

N° de patron	Trigrammes reconstruits (μ)	Proportion cumulée
1	71 150 (50,74 %)	70,34 %
2	23 810 (16,98 %)	93,87 %
3	6 003 (4,28 %)	99,81 %
4	156 (0,11 %)	99,96 %
5	37 (0,03 %)	100,00 %
Total	101 156 (72,14 %)	100,00 %

(c) Allemand

N° de patron	Trigrammes reconstruits (μ)	Proportion cumulée
1	36 717 (27,62 %)	61,48 %
2	16 064 (12,08 %)	88,37 %
3	6 227 (4,68 %)	98,80 %
4	548 (0,41 %)	99,72 %
5	169 (0,13 %)	100,00 %
Total	59 725 (44,93 %)	100,00 %

(d) Finnois

3.5 EFFECTIF SUFFISANT POUR LA RECONSTRUCTION D'UN TRIGRAMME INCONNU

Puisque l'hypothèse de la reconstruction massive des trigrammes inconnus par analogie est confirmée par les expériences précédentes, nous passons maintenant à l'étude des effectifs des trigrammes impliqués dans les reconstructions. Nous cherchons donc à savoir quels effectifs ont les trigrammes qui permettent la reconstruction des trigrammes inconnus. Une hypothèse naturelle serait que les trigrammes d'effectifs semblables aient tendance à apparaître dans les mêmes analogies. Suivant cette idée, les trigrammes inconnus, c'est-à-dire apparaissant zéro fois dans le corpus d'entraînement, seraient reconstruits à l'aide de trigrammes apparaissant une fois dans le corpus, c'est-à-dire les trigrammes hapax. Nous allons confirmer ici cette hypothèse.

en justice et : *en est ,* :: *justice et de* : *est , de*
débat en tant : *débat de ce* :: *en tant qu'* : *de ce qu'*
coûts de la : *coûts et les* :: *de la plus* : *et les plus*

FIGURE 11 : Exemples de trigrammes de mots du corpus français d'Europarl respectant le patron 2, c'est-à-dire $a\ b\ c : a\ d\ e :: b\ c\ f : d\ e\ f$.

debate and we : *debate and far* :: *but as we* : *but as far*
Union have set : *Union have that* :: *a committee set* : *a committee that*
but they do : *but they must* :: *so we do* : *so we must*

FIGURE 12 : Exemples de trigrammes de mots du corpus anglais d'Europarl respectant le patron 1, c'est-à-dire $a\ b\ c : a\ b\ d :: e\ f\ c : e\ f\ d$.

Nous effectuons une nouvelle série d'expériences afin d'obtenir les effectifs des trigrammes en relation d'analogie avec les trigrammes inconnus. En raison de la lourdeur des calculs, nous nous limitons à l'analyse du patron 1 : $a\ b\ c : a\ b\ d :: e\ f\ c : e\ f\ d$. Pour chaque instance de ce patron, nous définissons son *effectif maximum* comme l'effectif du trigramme le plus fréquent parmi les quatre trigrammes de l'analogie (comme le premier trigramme est inconnu, son effectif dans le corpus d'entraînement est évidemment zéro). Pour chaque trigramme inconnu reconstruit, nous mémorisons le minimum sur les effectifs maximums de toutes les analogies permettant de le reconstruire (effectif *min-max*). De cette mémorisation, et par inversion, pour chaque effectif *min-max*, nous pouvons compter le nombre de trigrammes inconnus reconstruits. Chaque effectif *min-max* est donc l'effectif suffisant à considérer pour trouver à coup sûr des trigrammes permettant la reconstruction de tant de trigrammes inconnus.

Ces décomptes sont donnés dans le tableau 9. Pour chaque effectif *min-max*, le tableau présente la quantité de trigrammes reconstruits et un pourcentage cumulé. Selon ces résultats, les instances d'analogie du patron 1 impliquant trois trigrammes hapax (effectif *min-max* = 1) permettent la reconstruction de plus de 96 % des trigrammes inconnus en anglais, 94 % en français ou en allemand et 91 % en finnois.

L'ensemble des résultats expérimentaux précédents soutiennent la conclusion que les hapax jouent un rôle majeur dans la reconstruction des trigrammes inconnus par analogie. En d'autres termes, non seulement les analogies entre trigrammes structurent les trigrammes inconnus, mais en plus, la reconstruction des trigrammes inconnus est massivement possible avec des trigrammes d'effectif semblable, c'est-à-dire d'effectif 1.

TABLEAU 9 : Pourcentages cumulés des trigrammes reconstruits, classés par effectif suffisant des trigrammes formant analogie pour leur reconstruction (colonne *effectif min-max*).

Effectif min-max	Trigrammes reconstruits	Pourcentage cumulé
1	54 227 (96,24 %)	96,24 %
2	1 288 (2,29 %)	98,53 %
3	345 (0,61 %)	99,14 %
4	127 (0,23 %)	99,36 %
5	99 (0,18 %)	99,54 %
⋮	⋮	⋮
523	1 (0,00 %)	100,00 %
TOTAL	56 345(100,00 %)	—

(a) Anglais

Effectif min-max	Trigrammes reconstruits	Pourcentage cumulé
1	59 050 (94,07 %)	94,07 %
2	2 167 (3,45 %)	97,52 %
3	608 (0,97 %)	98,49 %
4	302 (0,48 %)	98,97 %
5	167 (0,26 %)	99,24 %
⋮	⋮	⋮
576	1 (0,00 %)	100,00 %
TOTAL	62 771(100,00 %)	—

(b) Français

Effectif min-max	Trigrammes reconstruits	Pourcentage cumulé
1	41 272 (94,01 %)	94,01 %
2	1 475 (3,36 %)	97,36 %
3	465 (1,06 %)	98,42 %
4	219 (0,50 %)	98,92 %
5	124 (0,28 %)	99,21 %
⋮	⋮	⋮
412	1 (0,00 %)	100,00 %
TOTAL	43 904(100,00 %)	—

(c) Allemand

Effectif min-max	Trigrammes reconstruits	Pourcentage cumulé
1	13 382 (91,02 %)	91,02 %
2	760 (5,217 %)	96,18 %
3	238 (1,62 %)	97,80 %
4	101 (0,68 %)	98,49 %
5	56 (0,38 %)	98,87 %
⋮	⋮	⋮
458	1 (0,01 %)	100,00 %
TOTAL	56 542(100,00 %)	—

(d) Finnois

3.6 CONCLUSION

La méthode d'énumération des analogies d'un texte présentée au chapitre précédent nous a permis d'induire les patrons d'analogies entre trigrammes les plus importants. En fait, les patrons se distinguent à la fois par leurs fréquences dans les textes mais également par leur pouvoir à reconstruire des trigrammes inconnus. L'étude empirique menée dans ce chapitre conforte la validité de la propriété suivante : dans leur grande majorité les trigrammes inconnus sont analogues aux trigrammes hapax car leurs structures et leurs effectifs sont semblables. Ces propriétés permettent d'entrevoir une capacité de généralisation des exemples de traduction pouvant servir à un système de traduction par l'exemple. Dans le chapitre suivant nous allons exploiter ces propriétés dans une méthode de lissage par analogie.

4

APPLICATION DES ANALOGIES DE TERMES FIXES : LISSAGE DE MODÈLES DE LANGUE N-GRAMMES

Une série d'expériences sur quatre langues, sur des échantillons du corpus Europarl, a permis de conforter l'hypothèse suivante : dans leur grande majorité, les trigrammes inconnus d'un jeu de test peuvent être reconstruits par analogie avec des trigrammes hapax du corpus d'entraînement. De ce résultat, nous allons maintenant dériver une méthode de lissage simple pour les modèles de langue par trigrammes. Cette méthode va nous permettre d'obtenir de meilleurs résultats que les lissages de Witten-Bell, de Good-Turing et de Kneser-Ney dans des expériences menées en onze langues sur la partie commune d'Europarl, sauf pour le finnois.

4.1 INTRODUCTION

L'étude menée dans le chapitre précédent nous a conduit à la conclusion suivante : dans leur grande majorité, les trigrammes inconnus d'un jeu de test peuvent être reconstruits par analogie avec des trigrammes hapax du corpus d'entraînement. Ce résultat marquant va nous permettre de dériver une méthode de lissage simple pour les modèles de langue par trigrammes. Nous allons proposer une première formalisation en distinguant trois cas de figure et en introduisant deux paramètres. En étudiant la relation entre ces paramètres, nous allons constater qu'ils sont intimement liés et proposer une seconde formalisation. Enfin, nous allons évaluer notre méthode de lissage par analogie et montrer que cette méthode très compétitive par rapport aux techniques de lissage classiques conne Witten-Bell, Good-Turing et Kneser-Ney.

TECHNIQUES DE LISSAGE

Les techniques de lissage de modèles de langue reposent habituellement sur des hypothèses purement statistiques pour estimer la probabilité des évènements inconnus. Il y a dix ans, Rosenfeld (2000) constatait que :

> *Ironically, the most successful SLM techniques use very little knowledge of what language really is. The most popular language models (*n*-grams) take no advantage of the fact that what is being modeled is language.*

Nous présentons ici une technique de lissage pour les modèles de langue trigrammes qui repose sur la structure des évènements inconnus, c'est-à-dire la manière dont les trigrammes inconnus peuvent être construits à partir des trigrammes connus en utilisant une opération structurelle linguistiquement justifiée, l'analogie.

Le but du lissage des modèles de langue est d'attribuer des probabilités non-nulles aux évènements inconnus. Habituellement, les probabilités attribuées dépendent d'une caractérisation théorique des évènements inconnus. L'hypothèse à l'origine de notre travail est que les trigrammes inconnus peuvent être caractérisés, dans une large mesure, par la similitude de leurs structures avec des trigrammes rares. Plus précisément, nous avons montré dans le chapitre précédent que, dans une large mesure, les trigrammes inconnus sont analogues aux trigrammes connus et plus particulièrement aux trigrammes hapax, c'est-à-dire aux trigrammes apparaissant exactement une fois dans un corpus d'entraînement de taille donnée.

4.2 PREMIÈRE PROPOSITION DE LISSAGE DE MODÈLES TRIGRAMMES PAR ANALOGIE

4.2.1 *Notations*

Les méthodes de lissage présentées dans ce chapitre utilisent les notations suivantes :

- Le corpus d'entraînement T de longueur N est composé des mots $m_i : T = m_1 m_2 \dots m_N$.
- Les n-grammes sont notés : $m_{i,i+n-1} = m_i m_{i+1} \dots m_{i+n-1}$.
- Le vocabulaire des mots du corpus d'entraînement est noté V.
- Le décompte observé des n-grammes identiques à $m_{i,i+n-1}$ est noté $C(m_{i,i+n-1})$.

4.2.2 *Lissage de Laplace*

Nous présentons en premier lieu le lissage de Laplace puisque le lissage de Lidstone dérive de celui-ci. Le lissage de Laplace fait l'hypothèse que tout trigramme aurait dû être observé une fois de plus qu'il ne l'a été réellement. Ce qui permet de dire qu'un trigramme apparaissant pour la première fois en test est aussi observé une fois de plus que dans le jeu d'entraînement. La formule de la probabilité conditionnelle $p(m_i|m_{i-n+1,i-1})$[1] effectuant le lissage de Laplace est donc :

$$p(m_i|m_{i-n+1,i-1}) = \frac{C(m_{i-n+1,i}) + 1}{C(m_{i-n+1,i-1}) + |V|}$$

Le problème de ce lissage est qu'il peut attribuer une masse de probabilité trop importante aux trigrammes inconnus (Gale et Church, 1994). Cependant, cette méthode de lissage a l'avantage d'être simplissime à implanter et ne nécessite aucun paramètre.

4.2.3 *Lissage de Lidstone*

Le lissage de Lidstone est une généralisation de la méthode de lissage de Laplace. L'incrément aux décomptes des n-grammes observés est remplacé par un paramètre γ usuellement compris entre 0 et 1. La formule de la probabilité conditionnelle $p(m_i|m_{i-n+1,i-1})$ effectuant le lissage de Lidstone devient donc :

1 Nous utilisons ici les notations de Chen et Goodman (1999).

$$p(m_i|m_{i-n+1,i-1}) = \frac{C(m_{i-n+1,i}) + \gamma}{C(m_{i-n+1,i-1}) + \gamma|V|}$$

En raison de la critique mentionnée ci-dessus envers le lissage de Laplace, la valeur choisie pour le paramètre γ est beaucoup plus petite que 1 (cas du lissage de Laplace), et fixée à des valeurs non nulles mais très proches de 0.

4.2.4 *Lissage par analogie*

Habituellement, lorsqu'on utilise directement des outils tels que *SRILM* (Stolcke, 2002), on utilise les techniques de lissage classiques connues pour donner des résultats acceptables. Des techniques de lissage plus élaborées que Laplace ou Lidstone ont été proposées afin de réduire la taille des modèles de langue, nous pensons en particulier au *clustering* (Brown et coll., 1992). Cependant, de telles techniques requièrent une phase de pré-traitement complexe, ce qui accroît le coût de calcul (Matsuzaki et coll., 2003). En comparaison, la méthode que nous proposons ici extrait peu de connaissances supplémentaires des données d'entraînement : la proportion de trigrammes inconnus et la proportion relative de trigrammes inconnus reconstructibles par analogie. La structure des trigrammes inconnus est vérifiée au fil du calcul. Les principaux avantages de cette méthode sont sa simplicité et sa facilité d'utilisation.

Nous allons dériver le lissage par analogie du lissage de Lidstone. Dans cette méthode, nous considérons une partition du vocabulaire des trigrammes et leurs proportions :

- λ est la proportion de trigrammes inconnus du jeu de test (absents du corpus d'entraînement). Les trigrammes composés d'un ou de plusieurs mots inconnus tombent évidemment dans ce cas. Parmi les trigrammes inconnus, nous distinguons deux sous-ensembles ayant les proportions suivantes :
  - μ est la proportion relative de trigrammes inconnus reconstruits par analogie (absents du corpus d'entraînement mais reconstruits par analogie à partir des trigrammes du jeu d'entraînement).
  - $1 - \mu$ est la proportion relative de trigrammes inconnus non-reconstruits par analogie (absents du corpus d'entraînement et non-reconstruits par analogie). En particulier, les trigrammes composés d'un ou de plusieurs mots inconnus ne peuvent pas être reconstruits par analogie et par conséquent comptabilisé dans ce cas.

- $1 - \lambda$ est la proportion de trigrammes connus dans le jeu de test (présents à la fois dans le corpus d'entraînement et dans le corpus de test).

On peut avoir l'impression ici que λ et μ sont des paramètres dépendant du jeu de test. C'est théoriquement vrai. Mais en pratique, nous les estimerons à partir du corpus d'entraînement, ce qui les rendra indépendants du jeu de test. Deux paramètres sont utilisés dans cette méthode. Le premier, α_1, correspond à l'incrément appliqué aux décomptes des n-grammes reconstruits. Le second, α_2, correspond à l'incrément appliqué aux décomptes des n-grammes non-reconstruits. Les n-grammes connus doivent être privilégiés par rapport aux n-grammes inconnus, c'est pourquoi l'incrément appliqué aux décomptes des n-grammes connus doit être supérieur à la fois à l'incrément des n-grammes reconstruits mais aussi à l'incrément des n-grammes non-reconstruits. Nous choisissons de privilégier les n-grammes connus par rapport aux n-grammes reconstruits, c'est pourquoi nous utilisons la solution la plus simple qui consiste à ajouter la somme $\alpha_1 + \alpha_2$ aux décomptes des n-grammes connus. Les décomptes réestimés notés C^* peuvent alors s'écrire de la manière suivante :

- trigrammes connus :

$$C^*(m_{i-2,i}) = C(m_{i-2,i}) + \alpha_1 + \alpha_2$$

- trigrammes inconnus pouvant être reconstruits par analogie

$$C^*(m_{i-2,i}) = C(m_{i-2,i}) + \alpha_1$$

- trigrammes inconnus ne pouvant être reconstruits par analogie :

$$C^*(m_{i-2,i}) = C(m_{i-2,i}) + \alpha_2$$

Il est nécessaire de déterminer la somme des décomptes réestimés pour tout mot du vocabulaire m dans un historique donnée $m_{i-2,i-1}$ afin de proposer une formalisation de la probabilité conditionnelle d'un mot m_i sachant son historique $m_{i-2,i-1}$. En notant $\text{inc}(m_{i-2,i})$ l'incrément aux décomptes du trigramme $C(m_{i-2,i})$, nous obtenons :

$$\begin{aligned}\sum_{m \in V} C^*(m_{i-2,i-1} \cdot m) &= \sum_{m \in V} C(m_{i-2,i-1} \cdot m) + \text{inc}(m_{i-2,i-1} \cdot m) \\ &= C(m_{i-2,i-1}) + \sum_{m \in V} \text{inc}(m_{i-2,i-1} \cdot m)\end{aligned}$$

Nous faisons l'hypothèse que la somme des incréments dépend uniquement de la nature des trigrammes et non de leurs derniers mots.

Aucune expérience n'a apporté d'éléments permettant de réfuter cette hypothèse. Sous cette hypothèse, nous sommes en mesure d'utiliser la répartition de trigrammes connus, inconnus reconstructibles et inconnus non-reconstructibles pour exprimer la somme des incréments.

$$\sum_{m \in V} \text{inc}(m_{i-2,i-1} \cdot m) = \big((1-\lambda)\times(\alpha_1+\alpha_2) + \lambda\mu\times\alpha_1 + \lambda(1-\mu)\times\alpha_2\big)|V|$$

Nous posons $\zeta = (1-\lambda)\times(\alpha_1+\alpha_2) + \lambda\mu\times\alpha_1 + \lambda(1-\mu)\times\alpha_2$. Finalement, nous obtenons les probabilités conditionnelles suivantes :

- trigrammes connus :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \alpha_1 + \alpha_2}{C(m_{i-2,i-1}) + \zeta|V|}$$

- trigrammes inconnus pouvant être reconstruits par analogie

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \alpha_1}{C(m_{i-2,i-1}) + \zeta|V|}$$

- trigrammes inconnus ne pouvant être reconstruits par analogie :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \alpha_2}{C(m_{i-2,i-1}) + \zeta|V|}$$

La section suivante discute de l'avantage de la méthode de lissage par analogie par rapport à la méthode de Lidstone.

4.2.5 *Entropie croisée*

La mesure utilisée pour rendre compte de la qualité d'une méthode de lissage est l'entropie croisée. Chen et Goodman (1999) entre autres en donnent la définition. Étant donné un modèle de langue lissé par la méthode M dont la fonction de probabilité est notée p_M, un jeu de test constitué de l_{phrase} phrases notées t_i dont la longueur totale en mots est l_{mot}, l'entropie croisée est définie par la formule suivante :

$$\frac{1}{l_{mot}} \sum_{i=1}^{l_{phrase}} -\log_2 p_M(t_i)$$

Selon le logarithme employé, l'unité de l'entropie croisée est mesurée en bit/mot avec le logarithme en base 2, nat/mot avec le logarithme népérien, *etc.* Dans la pratique, le logarithme utilisé est le logarithme en base 2.

4.2.6 *Effets disjoints des paramètres du lissage par analogie*

Afin de comparer la méthode de lissage par analogie avec la méthode de lissage de Lidstone, nous utilisons une partie des corpus Europarl totalisant 100 000 phrases en chaque langue. Nous faisons cette étude sur les quatre langues suivantes : l'anglais, le français, l'allemand et le finnois. À partir de corpus d'entraînement de 90 000 phrases et de corpus de test de 10 000 phrases issues d'Europarl, nous cherchons à reconstruire les trigrammes inconnus. Les cinq patrons d'analogies de termes trigrammes les plus fréquents ont été utilisés pour la reconstruction des trigrammes inconnus. Pour rappel, la proportion de trigrammes reconstruits est de l'ordre de 80 % pour les trois premières langues étudiées : l'anglais, le français et l'allemand. Cette proportion est bien moins importante pour le finnois : seulement 45 % de trigrammes reconstruits par analogie.

La qualité des méthodes de lissage est mesurée à l'aide de l'entropie croisée, mesurée en bits/mots. La qualité d'une méthode de lissage augmente lorsque l'entropie croisée mesurée diminue. Les valeurs des paramètres γ, α_1 et α_2 sont prises dans l'intervalle $]0;1[$. En premier lieu, nous nous intéressons à l'effet des paramètres de la méthode de lissage par analogie pris indépendamment sur l'entropie croisée.

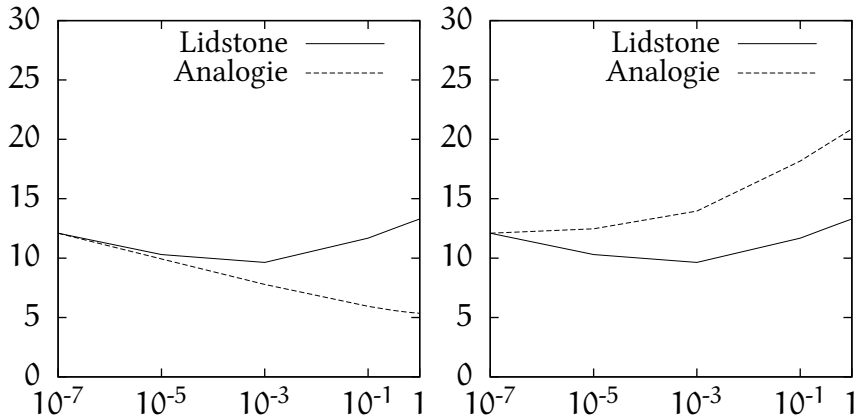
4.2.6.1 *Effet du paramètre α_1*

Les graphes (a) des figures 13, 14, 15 et 16 tracent l'évolution de l'entropie croisée des modèles lissés par Lidstone et lissés par analogie en fonction de la valeur du paramètre γ pour Lidstone et α_1 pour l'analogie. Les valeurs des paramètres γ et α_1 sont égales et le paramètre α_2 est fixé à une valeur supposée négligeable : 10^{-7}. Ainsi les graphes (a) permettent de comparer l'effet des paramètres γ et α_1. Pour plus de clarté, les courbes sont tracées sur une échelle logarithmique pour les valeurs des paramètres (en abscisses).

Nous constatons le même effet pour toutes les langues : plus la valeur du paramètre α_1 est proche de 1, plus l'entropie croisée décroît. Nous en concluons que les trigrammes reconstruits par analogie peuvent être considérés comme hapax. La valeur optimale pour le lissage de Lidstone est de 10^{-3} pour les quatre langues étudiées. Quelle que soit la langue, la méthode de lissage par analogie permet d'obtenir un gain absolu d'environ 5 bits/mots par rapport à la méthode de Lidstone.

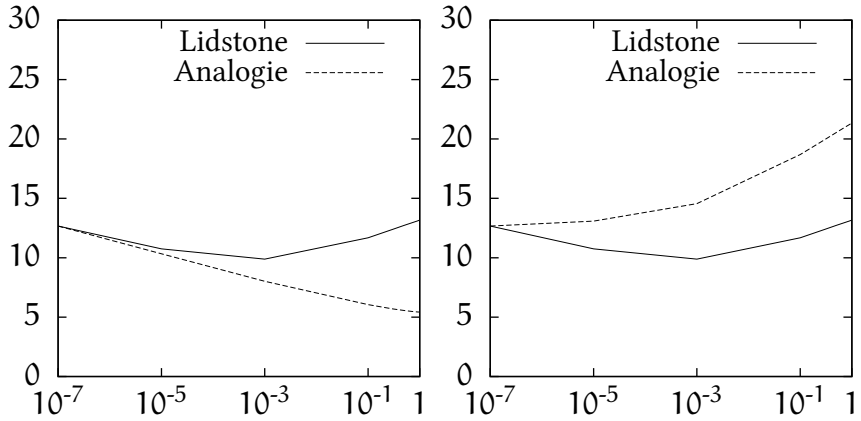
4.2.6.2 *Effet du paramètre α_2*

Les graphes (b) des figures 13, 14, 15 et 16 tracent l'évolution de l'entropie croisée des modèles lissés par Lidstone et lissés par analogie en fonction de la valeur du paramètre γ pour Lidstone et α_2 pour l'analogie. Le



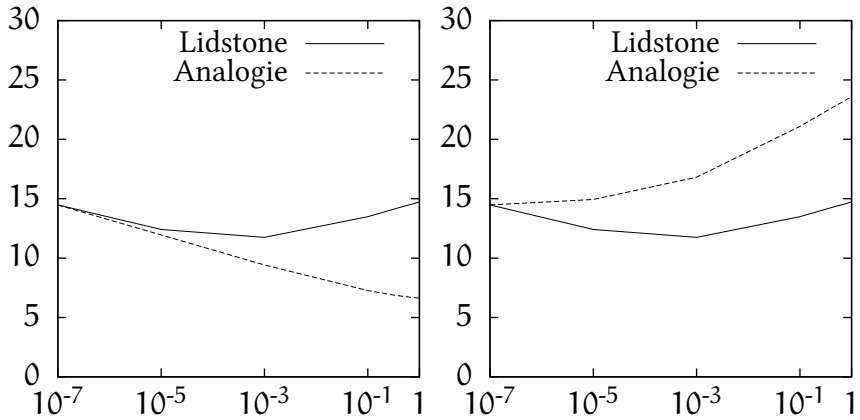
(a) $\gamma = \alpha_1$ et $\alpha_2 = 10^{-7}$ (b) $\gamma = \alpha_2$ et $\alpha_1 = 10^{-7}$

FIGURE 13 : Entropie croisée des modèles de langue du français lissés en fonction du paramètre γ pour le lissage de Lidstone et α_1 ou α_2 pour le lissage par analogie.



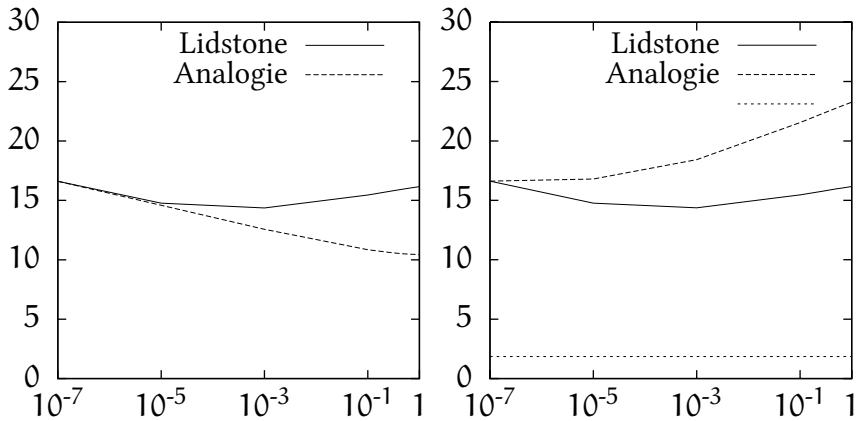
(a) $\gamma = \alpha_1$ et $\alpha_2 = 10^{-7}$ (b) $\gamma = \alpha_2$ et $\alpha_1 = 10^{-7}$

FIGURE 14 : Même chose qu'en figure 13, mais pour l'anglais.



(a) $\gamma = \alpha_1$ et $\alpha_2 = 10^{-7}$ (b) $\gamma = \alpha_2$ et $\alpha_1 = 10^{-7}$

FIGURE 15 : Même chose qu'en figure 13 (voir page précédente), mais pour l'allemand.



(a) $\gamma = \alpha_1$ et $\alpha_2 = 10^{-7}$ (b) $\gamma = \alpha_2$ et $\alpha_1 = 10^{-7}$

FIGURE 16 : Même chose qu'en figure 13 (voir page précédente), mais pour le finnois.

paramètre α_1 est fixé à une valeur supposée négligeable : 10^{-7}. Ainsi les graphes (b) permettent de comparer l'effet des paramètres γ et α_2. Comme précédemment, les courbes sont tracées sur une échelle logarithmique pour les valeurs des paramètres (en abscisses).

Les courbes mettent en évidence que le paramètre α_2 seul a un effet négatif sur l'entropie croisée puisque l'entropie croisée croît avec la valeur du paramètre α_2. Nous constatons que les courbes du lissage par analogie et celles du lissage de Lidstone sont superposables à une rotation près. Si la courbe de l'entropie croisée pour la méthode de Lidstone est croissante à partir de la valeur 10^{-7} sans doute parce que l'incrément aux décomptes des occurrences des trigrammes inconnus non reconstructibles par analogie devient trop élevé : cas du paramètre α_2 dans la méthode de lissage par analogie.

4.2.6.3 *Conclusion intermédiaire*

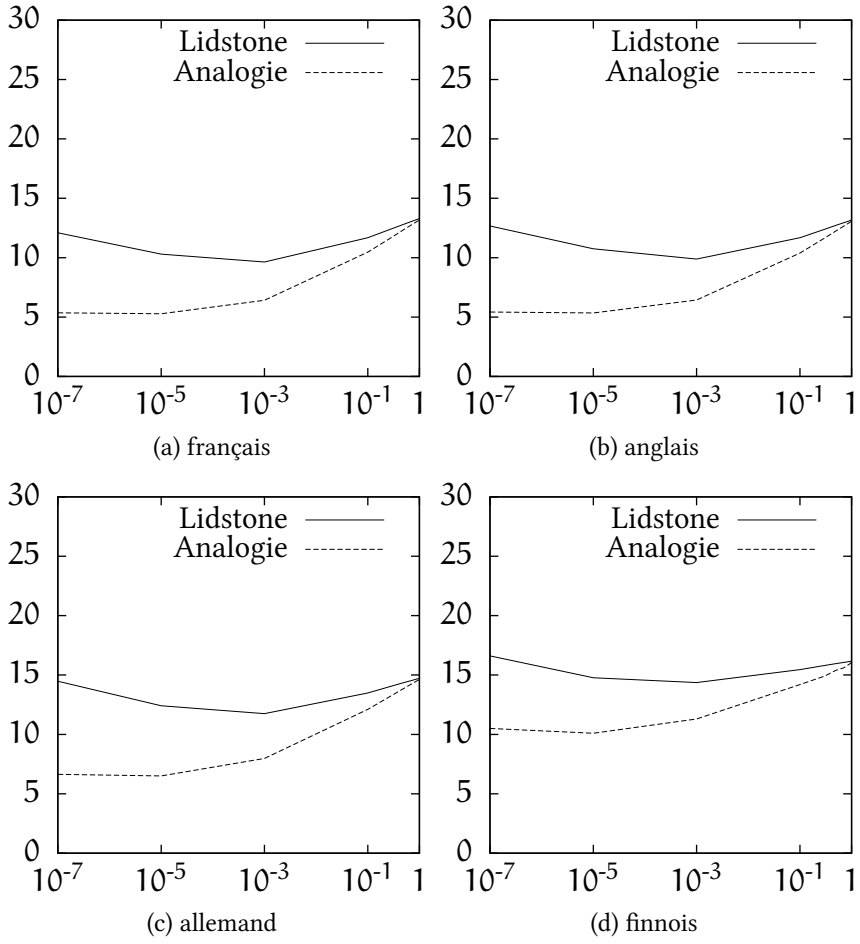
Cette étude montre que la méthode de lissage par analogie est préférable à la méthode de lissage de Lidstone. En effet la distinction entre trigrammes inconnus reconstructibles par analogie ou non permet de lisser plus finement le modèle de langue. Cette expérience confirme encore que les trigrammes inconnus reconstructibles par analogie peuvent être assimilés à des hapax (α_1 doit prendre une valeur très proche de 1) tandis que les trigrammes inconnus non reconstructibles peuvent être quasiment ignorés.

4.2.6.4 *Optimisation du paramètre α_2 de la méthode de lissage par analogie*

Afin de préciser la contribution du paramètre α_2 à la méthode de lissage par analogie, nous nous plaçons dans le cas où le paramètre α_1 est très proche de 1, nous fixons ce paramètre arbitrairement à : $1 - 10^{-6}$. Puis nous effectuons la même expérience que dans la section 4.2.6.2.

Les graphes obtenus sont tracés dans la figure 17. Pour toutes les langues, les valeurs optimales pour le paramètre α_2 sont comprises dans l'intervalle $[10^{-7}; 10^{-5}]$, ce qui justifie *a posteriori* le choix de la valeur 10^{-7} utilisée précédemment.

En conclusion, la méthode de lissage par analogie offre un gain d'environ 5 bits/mots par rapport à la méthode de lissage de Lidstone. De plus, les expériences nous conduisent à choisir systématiquement une valeur proche de 1 pour le paramètre α_1 et une valeur faible d'environ 10^{-6} pour α_2. La valeur, proche de 1, du paramètre α_1 signifie que les trigrammes reconstruits par analogie peuvent être considérés comme des hapax.



(a) français (b) anglais

(c) allemand (d) finnois

FIGURE 17 : Optimisation du paramètre α_2 de la méthode de lissage par analogie : les paramètres $\gamma = \alpha_2$ varient et $\alpha_1 = 1 - 10^{-6}$.

4.2.7 *Effets conjoints des paramètres du lissage par analogie*

Nous reprenons l'expérience précédente pour optimiser conjointement les paramètres α_1 et α_2 de la méthode de lissage par analogie. Nous faisons donc varier en même temps les deux paramètres pour les valeurs 10^{-i} pour i variant de 0 à 7. Les langues étudiées sont le français et l'anglais pour des corpus d'entraînement de 90 000 phrases et des corpus de test de 10 000 phrases (échantillon des corpus d'Europarl). Les résultats sont présentés sous forme de courbes de niveaux dans les figures 18 et 19. À chaque courbe tracée correspond une valeur différente de l'entropie croisée. Le paramètre α_1 varie selon l'axe des abscisses, le paramètre α_2 varie selon l'axe des ordonnées. Les deux axes utilisent une échelle logarithmique. Une courbe de niveau est tracée pour chaque valeur entière d'entropie croisée dans l'intervalle $[6; 22]$.

Les courbes de niveaux permettent de confirmer la conclusion formulée en section 4.2.6.3 pour les deux langues, à savoir que l'entropie croisée est minimale lorsque le paramètre α_1 est proche de 1 et que le paramètre α_2 est d'environ 10^{-6}. L'entropie croisée étant plus faible pour les modèles de langues de meilleure qualité, nous en concluons que notre première proposition de lissage par analogie est meilleure en fixant les paramètres comme suit :

$$\begin{aligned} \alpha_1 &= 1 - 10^{-6} \\ \alpha_2 &= 10^{-6} \end{aligned}$$

4.3 SECONDE PROPOSITION POUR UN LISSAGE DE MODÈLES TRIGRAMMES PAR ANALOGIE

Notre première proposition pour un lissage de modèles trigrammes par analogie employait les paramètres α_1 et α_2, respectivement l'incrément appliqué aux décomptes des trigrammes inconnus reconstruits et l'incrément appliqué aux décomptes des trigrammes inconnus non-reconstruits. Or l'optimisation de ces paramètres pour plusieurs langues conduit systématiquement au résultat suivant $\alpha_1 = 1 - 10^{-6}$ et $\alpha_2 = 10^{-6}$. Nous en concluons que ces paramètres sont liés et donc qu'un seul est nécessaire. Nous remarquons également que l'optimisation des valeurs des paramètres revient à un lissage de Laplace puisque $\alpha_1 + \alpha_2 = 1$. Nous utilisons ce résultat pour proposer une seconde formulation d'un lissage de modèle trigrammes par analogie. Nous avons vu en section 4.2.2 que la critique adressée au lissage de Laplace (dont notre première proposition est proche) est qu'une masse de probabilités trop importante est attribuée aux trigrammes inconnus. C'est aussi sans doute

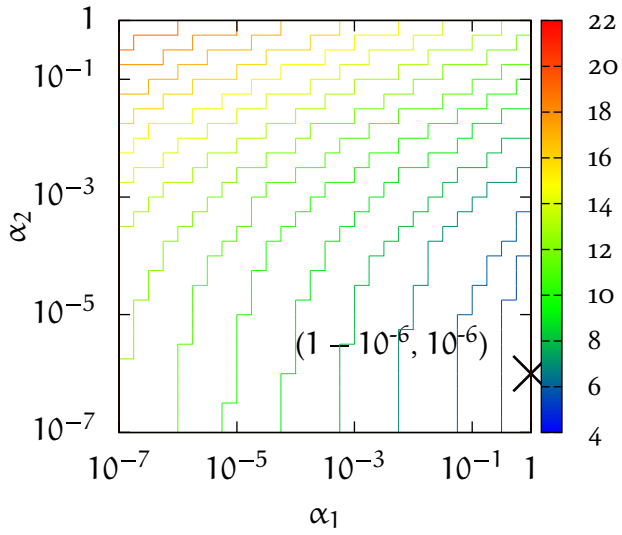


FIGURE 18 : Courbe de niveaux de l'entropie croisée pour l'optimisation des paramètres α_1 et α_2 de la méthode de lissage par analogie : cas du français.

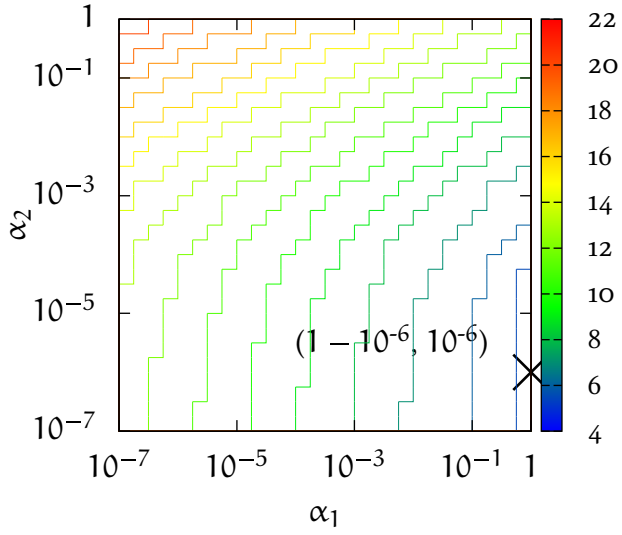


FIGURE 19 : Courbe de niveaux de l'entropie croisée pour l'optimisation des paramètres α_1 et α_2 de la méthode de lissage par analogie : cas de l'anglais.

le cas dans notre première proposition. Notre seconde proposition vise à éviter cet écueil.

Afin d'éviter de surestimer les trigrammes connus, nous introduisons un paramètre γ jouant exactement le même rôle que le paramètre γ de la technique de Lidstone. Afin de raffiner le modèle, nous ajoutons aussi un mécanisme de repli. Finalement, les paramètres de notre seconde formalisation sont liées aux paramètres de la première formalisation par les formules suivantes :

$$\begin{aligned} \gamma &= \alpha_1 + \alpha_2 \\ \alpha &= \alpha_2 \end{aligned}$$

Cependant, les valeurs des paramètres γ et α vont nettement se distinguer des valeurs des paramètres α_1 et α_2 du fait de la mise en place du mécanisme de repli.

4.3.1 *Ré-estimation des effectifs*

Redisons une vérité élémentaire : tout évènement inconnu apparaissant dans le jeu de test a un effectif nul dans le corpus d'entraînement. Immédiatement au-dessus de la classe des évènements d'effectif nul, vient la classe des évènements observés une seule fois dans le corpus d'entraînement : ce sont les hapax. Or, il est classique pour une technique de lissage d'essayer d'estimer la probabilité lissée des évènements inconnus en se basant sur les propriétés des évènements classés selon leurs fréquences d'apparition : c'est la base du lissage de Good-Turing (Gale et Sampson, 1995). Nous exploitons la même idée mais dans une mise en application plus simple.

Dans le lissage de Lidstone, tout évènement voit son effectif augmenté de γ. Nous gardons cet incrément de γ pour les évènements connus. L'essence de notre technique de lissage tient toujours dans la distinction faite entre évènements inconnus selon qu'ils peuvent être reconstruits par analogie ou non.

Nous donnons un fort avantage aux évènements inconnus qui peuvent être reconstruits par analogie au détriment de ceux qui ne peuvent pas l'être. Les résultats des expériences présentées en section 3.5 conduisent à proposer un effectif plus important pour les trigrammes reconstructibles qu'aux trigrammes non-reconstructibles. Nous fixons leur effectif à $\gamma - \alpha$ avec α proche de 0.

En dernier ressort, nous affectons comme estimation des effectifs des trigrammes inconnus qui ne peuvent pas être reconstruits une valeur très proche de 0. Afin de conserver l'égalité $\alpha = \alpha_2$, nous utilisons la valeur α. Ainsi $\gamma - \alpha + \alpha = \gamma$.

Au total donc, la probabilité lissée d'un trigramme $m_{i-2,i}$ ($m_{i-2,i-1}$ représente les deux mots précédant m_i) est ré-estimée selon chacun des trois cas suivants, avec $|V|$ la taille du vocabulaire et δ restant à déterminer :

- trigrammes connus :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \gamma}{C(m_{i-2,i-1}) + \delta \times |V|}$$

- trigrammes inconnus pouvant être reconstruits par analogie :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \gamma - \alpha}{C(m_{i-2,i-1}) + \delta \times |V|}$$

- trigrammes inconnus ne pouvant être reconstruits par analogie :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + \alpha}{C(m_{i-2,i-1}) + \delta \times |V|}$$

Par rapport à la première proposition (voir section 4.2), cette nouvelle formulation permet de mieux distinguer l'incrément γ apporté aux trigrammes connus et reconstruits par analogie de l'incrément α affecté aux trigrammes inconnus.

En reprenant les notations de la section 4.2 et du tableau 6, nous notons λ la proportion de trigrammes inconnus et μ la proportion relative de trigrammes inconnus reconstruits par analogie.

En suivant les mêmes étapes de calcul qu'en section 4.2, nous déterminons la valeur de δ permettant de garantir la normalisation des décomptes réestimés :

$$\delta \quad = \quad (1-\lambda) \times \gamma \quad + \quad \mu\lambda \times (\gamma - \alpha) \quad + \quad (1-\mu)\lambda \times \alpha$$

De plus, lorsque l'historique d'un mot est inconnue, nous utilisons une stratégie de repli semblable à celle décrite par Chen et Goodman (1999) pour le modèle p_{add} : formellement, lorsque $C(m_{i-2,i}) = 0$ alors $p(m_i|m_{i-2,i-1}) = p(m_i|m_{i-1})$.[2] Plus précisément, en notant l'incrément $inc(m_{i-2,i})$, la formule de base :

$$p(m_i|m_{i-2,i-1}) = \frac{C(m_{i-2,i}) + inc(m_{i-2,i})}{C(m_{i-2,i-1}) + \delta \times |V|}$$

est remplacé par

2 Nous avions préalablement considéré de n'employer le mécanisme de repli uniquement pour les cas où l'historique était inconnue ($C(m_{i-2,i}) = 0$). Des résultats préliminaires ont confirmé qu'il est préférable d'utiliser le mécanisme de repli dès lors que l'évènement lui-même est inconnu.

$$p(m_i|m_{i-2,i-1}) = p(m_i|m_{i-1}) = \frac{C(m_{i-1,i}) + inc(m_{i-2,i})}{C(m_{i-1}) + \delta \times |V|}$$

Une solution de repli équivalente est utilisée lorsque $C(m_{i-1,i}) = 0$, dans ce cas la formule utilisée est :

$$p(m_i|m_{i-2,i-1}) = p(m_i) = \frac{C(m_i) + inc(m_{i-2,i})}{N + \delta \times |V|}$$

4.3.2 *Estimation des paramètres*

Dans la pratique, les paramètres λ et μ sont estimés dans une phase de pré-traitement. Le corpus d'entraînement est divisé en deux parties, l'une comprenant les neuf dixièmes du corpus, l'autre comprenant le dixième restant. Le pourcentage de trigrammes inconnus dans la plus petite partie ainsi que la part du pourcentage de trigrammes inconnus reconstruits par analogie sont estimés par échantillonnage. Ces estimations deviennent les valeurs des paramètres λ et μ.

Concernant le paramètre α, les résultats d'expériences présentés en section 4.2.6.3 nous ont conduit à le fixer à 10^{-6} pour toutes les langues.

Enfin, une optimisation de la valeur du paramètre γ nous conduit à le fixer à 2×10^{-5} en employant le premier patron d'analogie et 10^{-5} avec les deux premiers patrons d'analogies.

4.3.3 *Temps d'exécution*

Afin de déterminer si un trigramme inconnu peut être reconstruit ou non par analogie, le corpus d'entraînement est mémorisé sous forme de deux tableaux de suffixes (sens de lecture normal et miroir). Lorsque la reconstruction d'un trigramme doit être testée, pour chaque patron d'analogie, une recherche appropriée à ce patron est effectuée dans ces tableaux de suffixes.

Par exemple, pour le patron 1 ($a\ b\ c : a\ b\ d :: e\ f\ c : e\ f\ d$), le trigramme candidat $a\ b\ c$ est décomposé en une partie gauche $a\ b$ et une partie droite c. La recherche de ces séquences dans les tableaux de suffixes est très rapide. Il suffit de prendre l'intersection en termes de positions de l'ensemble des unigrammes d qui suivent $a\ b$ et de l'ensemble des bigrammes $e\ f$ qui précèdent c. Dès qu'une position est trouvée dans l'intersection, nous en déduisons qu'il existe au moins un trigramme $e\ f\ d$ dans le corpus d'entraînement et nous pouvons conclure en l'existence d'une analogie $a\ b\ c : a\ b\ d :: e\ f\ c : e\ f\ d$. Cela signifie que le trigramme $a\ b\ c$ peut être reconstruit par analogie à l'aide de trigrammes du corpus

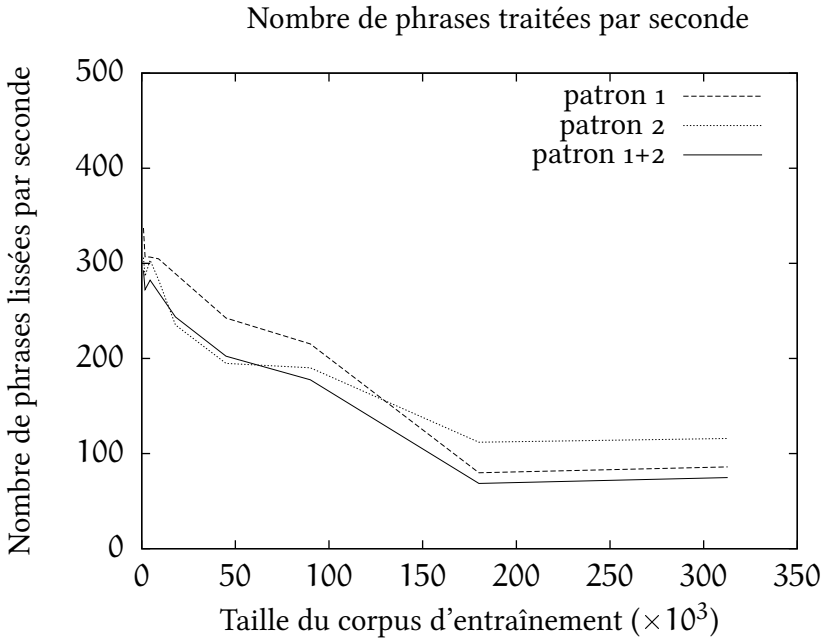


FIGURE 20 : Vitesse de la méthode de lissage par analogie pour différentes tailles du corpus d'entraînement, pour deux variantes de la méthode : patron 1 et patrons 1 et 2.

d'entraînement. Une procédure similaire a été implantée pour le patron 2 ($a\ b\ c : a\ d\ e :: b\ c\ f : d\ e\ f$).

L'implantation des lissages classiques de *SRILM* permet de lisser environ 1 000 phrases par seconde en français quelle que soit la méthode de lissage et quelle que soit la taille du corpus d'entraînement avec une machine équipée d'un processeur 32 bits cadencé à 2 GHz et 4 Go de mémoire.

Notre méthode de lissage effectue des recherches dans des tableaux de suffixes et nous nous attendons à ce que la vitesse de lissage dépende de la taille du corpus d'entraînement. Sur le même type de machine, nous mesurons la vitesse de notre méthode de lissage en fonction de la taille du corpus d'entraînement pour deux variantes : patron 1 seul et patrons 1 et 2. Nous utilisons des échantillons de la partie française d'Europarl avec des tailles variant de 900 à 320 000 phrases. Dans la seconde variante, le patron 2 est utilisé en deuxième instance dans le cas où le patron 1 n'a pas permis de reconstruire le trigramme.

Les courbes de la figure 20 donnent le nombre de phrases du jeu de test traitées par seconde en fonction de la taille du corpus d'entraînement. La vitesse de lissage de notre méthode dépend naturellement de la taille du corpus d'entraînement. Pour de petits corpus, notre implantation traite 300 phrases par seconde. Cette vitesse chute à 100 phrases par seconde pour les corpus de plus grande taille et n'évolue plus vraiment à partir de 180 000 phrases. Les deux variantes sont similaires, ce qui signifie que

la variante patrons 1 et 2 n'engendre qu'un faible surcoût de temps de traitement.

L'implantation actuelle de notre méthode de lissage par analogie, en Python, est donc dix fois plus lente que les implantations de *SRILM* en C++ des méthodes classiques de lissage. On peut raisonnablement espérer des temps bien inférieurs avec une implantation en C++ si l'on se fie aux règles très grossières donnant des accélérations par cinquante lors de réécritures de Python en C++.[3]

4.3.4 *Évaluation des performances*

Nous comparons maintenant notre méthode de lissage par analogie avec quatre méthodes de lissage classiques : Lidstone (Chen et Goodman, 1999), Witten-Bell (Witten et Bell, 1991), Good-Turing (Gale et Sampson, 1995), et la version originale de Kneser-Ney (Kneser et Ney, 1995). Cette dernière méthode est souvent considérée comme la meilleure en pratique.[4] Pour ces trois méthodes, nous utilisons l'implantation de *SRILM* (Stolcke, 2002).

Dans le cas du lissage de Lidstone, après optimisation, nous utilisons la valeur 10^{-3} pour le paramètre γ pour chaque langue.

Le critère d'évaluation utilisé est la perplexité en mots. La perplexité en mots est définie comme la moyenne géométrique des inverses des probabilités réestimées. En notant $|\mathrm{test}|$ le nombre de mots du jeu de test, la perplexité en mots est :

$$\mathrm{PPL} = \exp_2 \frac{-\sum_{i=1}^{|\mathrm{test}|} \log_2 p(m_i|m_{i-2,i-1})}{|\mathrm{test}|}$$

Dans ces formules, $p(m_i|m_{i-2,i-1})$ est la probabilité conditionnelle obtenue par le jeu de test, avec m_i le mot à la position i et $m_{i-2,i-1}$ son historique, c'est-à-dire les deux mots précédant m_i.

Les modèles de langues sont configurées de telles sorte qu'ils utilisent deux marqueurs de début de phrase. Dans la formule précédente, ces marqueurs sont aux indices -1 et 0.

D'autre part, les mots inconnus sont considéré comme un nouveau mot du vocabulaire. Les mots inconnus sont simplement remplacés par le mot `<unk>`. Ce mot vient donc s'ajouter au vocabulaire et la taille de ce dernier est donc augmenté de 1.

3 Voir par exemple : `http://shootout.alioth.debian.org/u32q/benchmark.php?test=all&lang=gpp&lang2=python`

4 « Kneser-Ney (1995) smoothing and its variants are generally recognized as having the best perplexity of any known method for estimating N-gram language models. » (Moore et Quirk, 2009).

TABLEAU 10 : Statistiques des corpus d'entraînement et des jeux de tests utilisés pour la comparaison.

	Corpus d'entraînement : 347 613 phrases			
Langue	Taille			
	Mo	mots ($\times 10^6$)	mots / phrase	taille du vocabulaire
da	53,39	9,46	27,21	153 425
de	60,02	9,51	27,36	167 942
el	105,37	10,00	28,76	149 247
en	52,39	9,94	28,60	67 819
es	57,90	10,47	30,12	100 410
fi	55,79	7,18	20,65	299 116
fr	60,61	10,95	31,51	86 567
it	57,50	9,88	28,42	99 252
nl	57,42	10,01	28,80	125 565
pt	58,56	10,29	29,59	102 800
sv	54,45	8,99	25,86	157 116

	Jeux de test : 38 624 phrases			
Langue	Taille			
	Mo	mots ($\times 10^6$)	mots / phrase	taille du vocabulaire
da	5,96	1,06	27,36	46 117
de	6,70	1,06	27,48	51 398
el	11,76	1,12	28,89	52 671
en	5,85	1,11	28,76	25 854
es	6,46	1,17	30,27	37 128
fi	6,22	0,80	20,74	84 964
fr	6,77	1,22	31,65	33 403
it	6,41	1,10	28,54	36 624
nl	6,42	1,12	29,00	39 728
pt	6,53	1,15	29,73	38 041
sv	6,08	1,00	25,98	48 327

TABLEAU 11 : Proportion de trigrammes inconnus (λ) et proportion de trigrammes inconnus reconstructibles par analogie (μ) estimées à partir d'un échantillon de 10 % du corpus d'entraînement pour chaque langue, et valeurs correspondantes de δ ($\alpha = 10^{-6}$, $\gamma = 2 \times 10^{-5}$ avec un patron, $\gamma = 10^{-5}$ avec deux patrons).

	Trigrammes inconnus				
	(λ)	Reconstruits (μ)		(δ) $\times 10^{-6}$	
		Patron 1	Patrons 1 et 2	Patron 1	Patrons 1 et 2
da	55,54 %	63,91 %	79,67 %	15,84	8,541
de	60,42 %	60,44 %	76,52 %	15,09	8,261
el	55,20 %	64,10 %	78,99 %	15,88	8,520
en	50,99 %	65,30 %	78,50 %	16,31	8,613
es	50,87 %	68,54 %	84,14 %	16,61	8,846
fi	**75,02 %**	**44,67 %**	**56,93 %**	**11,78**	**6,665**
fr	47,91 %	53,40 %	85,98 %	15,50	8,984
it	53,65 %	49,82 %	83,82 %	14,62	8,769
nl	54,78 %	52,00 %	82,44 %	14,72	8,683
pt	52,44 %	47,46 %	85,20 %	14,52	8,855
sv	58,90 %	47,25 %	77,93 %	13,82	8,371

TABLEAU 12 : Comparaison de la technique de lissage par analogie (patron 1 et patrons 1 et 2) avec quatre techniques de lissage classiques en onze langues selon la perplexité en mots.

	da	de	el	en	es	fi	fr	it	nl	pt	sv
Lidstone	143,2	206,7	147,7	98,6	97,2	871,2	75,6	125,9	143,1	113,6	200,3
Witten-Bell	98,0	143,8	105,9	82,6	79,3	495,3	63,5	102,2	108,6	90,9	136,7
Good-Turing	104,9	154,6	111,8	84,2	81,5	581,8	64,9	105,4	113,7	94,0	147,5
Kneser-Ney	90,7	130,7	99,1	77,9	74,6	**433,5**	59,4	95,6	101,6	85,0	127,0
Patron 1	85,1	122,9	86,6	62,0	62,8	561,5	49,5	76,4	88,2	70,4	116,7
Patron 1 et 2	**83,7**	**120,8**	**85,0**	**61,5**	**62,1**	549,0	**49,1**	**75,5**	**87,0**	**69,6**	**114,6**

La comparaison est effectuée sur des données extraites d'Europarl en onze langues. Pour chaque langue, les phrases ayant une traduction en anglais sont retenues. Nous obtenons de cette manière onze corpus alignés de 383 237 phrases. Chaque corpus est ensuite divisé en deux parties : 90 % du corpus pour l'entraînement, les 10 % restants servant de jeu de test. De cette manière, nos expériences sont véritablement comparables entre langues. Les statistiques concernant le corpus d'entraînement et le jeu de test pour chaque langue sont présentées dans le tableau 10. En particulier, la taille du vocabulaire du corpus d'entraînement pour chaque langue correspond à la valeur de $|V|$ dans les formules de probabilité décrivant le lissage par analogie.

Les estimations des paramètres λ et μ nécessaires à notre méthode de lissage sont détaillées dans le tableau 11. Nous rappelons que λ est la proportion de trigrammes inconnus et que μ est la proportion relative de trigrammes inconnus qui peuvent être reconstruits par analogie. Nous considérons deux variantes de notre méthode : la première n'utilise que le patron 1, la seconde utilise les patrons 1 et 2.

Afin de rendre la technique de lissage indépendante du jeu de test, pour chaque langue les paramètres λ et μ ont été estimés automatiquement à partir d'un échantillon aléatoire formé d'un dixième du corpus d'entraînement comme décrit dans la section 4.3.2. La normalisation des formules de probabilité conditionnelle est donc valide tant que les paramètres λ et μ dépendent uniquement de la langue considérée. Dans nos expériences, les corpus d'entraînement et de test ont été obtenus par échantillonnage d'un corpus, il est donc raisonnable de penser que les paramètres estimés à partir du corpus d'entraînement sont identiques pour le corpus de test. Nos expériences ont permis de vérifier que les paramètres sont bien les mêmes pour le corpus d'entraînement et le jeu de test.

Comme le montrent les chiffres du tableau 11, l'utilisation du patron 2 en plus du patron 1 augmente sensiblement la valeur du paramètre μ : plus de 10 % en valeurs absolues. À l'exception du finnois, l'utilisation conjointe des patrons 1 et 2 permet la reconstruction de 75 % à 85 % des trigrammes inconnus (paramètre μ). Les valeurs pour le finnois, en gras dans le tableau, sont nettement différentes des valeurs pour les autres langues.

Les résultats de l'évaluation des deux variantes de la méthode proposée sont présentés dans le tableau 12 :

- à l'exception du finnois, le patron 1 est suffisant pour atteindre de bien meilleurs résultats que ceux des trois méthodes de lissage classiques ;
- l'ajout du patron 2 permet d'améliorer encore un peu les résultats avec une réduction d'environ 1 bit/mot ;

- en finnois, le lissage par analogie obtient des résultats situés entre ceux de Good-Turing et ceux de Witten-Bell.

La contre-performance sur le finnois n'est pas surprenante si on considère le nombre important de trigrammes inconnus et la faible proportion de ces trigrammes qui peuvent être reconstruits par analogie (voir table 11). Afin de remédier à ce problème, plutôt que d'accroître la quantité de données d'entraînement, il serait sans doute plus judicieux de segmenter les mots en morphèmes.

4.4 DÉTAILS SUR L'IMPLANTATION

L'ensemble des calculs nécessaires à l'implantation de la technique de lissage par analogie profite grandement de l'utilisation de tableaux de suffixes comme structure de données principale. L'avantage de cette structure de données est de permettre une implantation efficace en espace et en temps. Les procédures principales du modèle de langue par analogie utilisant des tableaux de suffixes sont :

1. le décompte du nombre d'occurrences des n-grammes ;
2. la vérification de l'existence d'une analogie entre trois n-grammes hapax du corpus d'entraînement et un n-gramme inconnu :
3. le lissage des probabilités.

Aucune implantation des tableaux de suffixes existante ne convenait tout à fait à notre travail. Il existe des implantations efficaces et élégantes en langage C telle que celle de Kärkkäinen et coll. (2006).[5] Cependant, nous souhaitons obtenir un prototype aisément modifiable, ce que le langage C ne permet pas. Il en existe une traduction en Python.[6] Bien qu'écrite dans un langage haut niveau, cette implantation n'offre aucune facilité d'utilisation des tableaux produits. En particulier, aucune fonction de recherche n'y est proposée.

Nous avons décidé de partir de l'implantation en Python car ce langage satisfait nos exigences de simplicité de lecture et de facilité d'écriture. Puis nous avons ajouté les fonctions de base pour la recherche, à savoir le calcul des plus longs préfixes communs ou *LPC*, et une fonction de recherche par dichotomie. La recherche par dichotomie permet de trouver un suffixe du texte parmi ceux qui ont la chaîne recherchée pour préfixe. Le résultat de la recherche est étendu à l'ensemble des suffixes de mêmes

5 Disponible à l'adresse `http://www.mpi-sb.mpg.de/~sanders/programs/suffix/drittel.C` (dernier accès le 21 février 2011)

6 Disponible à l'adresse `http://code.google.com/p/pysuffix/` (dernier accès le 21 février 2011)

préfixes à l'aide des *LPC* en un temps linéaire en le nombre de résultats. L'algorithme utilisé pour effectuer le calcul des *LPC* n'est pas linéaire dans le pire des cas. L'article de Kärkkäinen et Sanders montre que théoriquement le calcul des *LPC* peut être effectué en même temps que la construction du tableau de suffixes et permettre un algorithme linéaire en temps. Nous avons constaté que dans la pratique, la construction des *LPC* avec notre algorithme naïf est suffisamment efficace pour traiter la partie française du corpus Europarl de plus de 12,5 millions de mots en 8 minutes soit un surcoût de 15 % seulement par rapport au temps de construction du tableau de suffixes seul. L'efficacité de notre implantation est présentée en annexe B.

4.5 CONCLUSION

En nous appuyant sur le fait qu'en majorité les trigrammes inconnus dans un jeu de test sont structurellement analogues aux trigrammes hapax d'un corpus d'entraînement, nous avons proposé une méthode de lissage pour modèles de langue trigrammes. L'effectif des trigrammes connus y est ré-estimé en appliquant un incrément de γ inférieur à 1 comme dans le lissage de Lidstone. Les trigrammes inconnus qui peuvent être reconstruits par analogie voient leurs effectifs ré-estimés à une valeur proche de γ. Les trigrammes inconnus qui ne peuvent être reconstruits par analogie sont presque ignorés, leurs effectifs étant fixés à une valeur proche de 0 comme dans le lissage de Lidstone. En comparaison de techniques de lissage utilisant des techniques de *clustering*, cette méthode est simple. Elle ne construit que deux classes de trigrammes inconnus : ceux qui peuvent être reconstruits par analogie et les autres. Il s'agit d'un lissage de Lidstone qui ré-estime très faiblement les trigrammes impossibles. Ces derniers sont définis par l'impossibilité de les reconstruire par analogie. Notre travail se rapproche donc des idées promues dans des travaux comme (Langlois et coll., 2003).

Des mesures sur onze langues ont montré que cette méthode de lissage donne de bons résultats en comparaison des techniques de lissage classiques, sauf dans le cas du finnois.

5

CONCLUSION GÉNÉRALE

POINT DE DÉPART : LE PROBLÈME DE LA SÉLECTION DES EXEMPLES

Le point de départ de ce travail de thèse a été l'identification du problème de la sélection des exemples dans les systèmes de traduction par l'exemple. Ce problème est d'autant plus évident dans le système de traduction par analogie *Aleph* où aucune sélection n'intervient à proprement parler mais où les exemples sont ordonnés selon une heuristique. Le problème de la sélection des exemples est un problème de généralisation des exemples qui se manifeste par la manipulation de nombreux exemples équivalents. En étudiant le cas de la sous-langue des dates et en organisant un corpus d'exemples en sous-grammaires par l'exemple, nous avons montré comment le système *Aleph* pourrait cesser d'être confronté au problème de la sélection des exemples. D'une part nous avons montré que ce système était capable de traduire parfaitement l'ensemble d'un corpus de 4 018 dates organisé en sous-grammaires. D'autre part nous avons montré que sans cette organisation en sous-grammaires, ce système avait besoin d'un échantillon aléatoire de 750 exemples, contre 58 exemples avec cette organisation.

POUR Y RÉPONDRE : BESOIN D'ÉNUMÉRER TOUTES LES ANALOGIES D'UN TEXTE

La résolution du problème de la sélection des exemples nous a conduit à étudier les régularités de structure entre exemples. À cette fin, nous avons dû mettre au point une méthode pour l'énumération des analogies présentes dans un texte. Étant donné qu'aucune méthode efficace n'avait été proposée pour répondre auparavant, nous avons mis au point une méthode originale, optimisable en deux temps. Cette méthode a été conçue pour être la plus générique possible, afin d'être adaptée simplement à la fois à nos besoins mais également à tout problème d'énumération d'analogies sans restriction *a priori* ni sur les termes d'analogies, ni sur la formulation exacte de l'analogie. La méthode originale emploie une stratégie de parcours en profondeur d'abord à partir de l'analogie de chaînes vides. Deux optimisations sont alors possibles. Une première optimisation consiste à employer un parcours en largeur d'abord. Elle réduit la mémoire utilisée de plus de 95 %. Une seconde optimisation

consiste à débuter le parcours à partir des premières analogies non triviales. Elle réduit le temps de traitement de plus de 70 %.

APPLICATION : ÉTUDE DE LA STRUCTURE DES ANALOGIES ENTRE TRIGRAMMES DE MOTS

La généralité de notre méthode d'énumération des analogies d'un texte, nous permis d'étudier en détail la structure des analogies entre trigrammes de mots d'un corpus. Nous avons pu induire les patrons d'analogies entre trigrammes de mots et avons constaté une concomitance entre structure des trigrammes de mots et fréquence d'apparition. En particulier les trigrammes hapax sont fortement structurés par l'analogie : les trois patrons d'analogie les plus fréquents en français, anglais, allemand et finnois sont identiques et permettent de reconstruire plus de 70 % des trigrammes inconnus en français, anglais et allemand et près de 45 % en finnois. Nous avons également montré qu'il est inutile de considérer un grand nombre de patrons d'analogies pour la reconstruction des trigrammes inconnus puisque les deux patrons les plus fréquents permettent de reconstruire 90 % des trigrammes reconstructibles.

UNE SOLUTION AU PROBLÈME DE DÉPART : LISSAGE DES MODÈLES DE LANGUES TRIGRAMMES PAR ANALOGIE

La structure des analogies entre trigrammes nous a permis de confirmer l'hypothèse selon laquelle deux patrons d'analogies suffisaient à généraliser quasiment toutes les instances de trigrammes de mots. La définition d'une technique de lissage des modèles de langues trigrammes exploitant cette propriété nous a permis d'appliquer cette hypothèse à un problème concret. Nous avons proposé une technique de lissage inspirée des techniques de lissage de Laplace et de Lidstone. Dans la technique proposée, nous avons distingué trois cas : le premier pour les trigrammes connus, le deuxième pour les trigrammes inconnus reconstruits par analogie et le troisième pour les trigrammes inconnus non-reconstruits par analogie. En optimisant les paramètres du lissage par analogie, nous avons découvert que les trigrammes inconnus reconstruits par analogie peuvent être considérés comme quasi-connus. Les expériences menées ont prouvé que ce lissage par analogie est compétitif pour 10 langues européennes sur 11.

PERSPECTIVES

Cette thèse offre plusieurs perspectives d'étude. Les travaux préliminaires nous ont conduit à entrevoir la possibilité de compartimenter les

exemples des traductions en sous-grammaires. Il est peut-être possible de généraliser ce concept de sous-grammaire en relâchant certaines contraintes. Par exemple, dans notre étude de la sous-langue des dates, nous avons supposé l'existence d'un prototype de dates dont les exemples des sous-grammaires ne font varier qu'un phénomène à la fois. Par quel moyen pourrait-on mettre en œuvre des sous-grammaires définies sur des prototypes différents ? D'autre part l'automatisation de la production de sous-grammaires reste ouverte. Par manque de temps, nous n'avons pas pu explorer cette piste.

Notre travail ayant permis la découverte des patrons d'analogie entre trigrammes, il offre un protocole d'étude qui peut facilement être mis en œuvre sur des n-grammes de mots d'ordre supérieur voire sur des chunks ou d'autres groupes syntagmatiques. Ce protocole offre un panel d'investigation tellement vaste qu'il sort nécessairement du cadre de ce mémoire. Mentionnons que nous avons commencé les expérimentations sur les 4-grammes et les 5-grammes de mots sans pouvoir la terminer. Nous avons préféré achever l'étude sur les trigrammes de mots avec la proposition de notre méthode de lissage par analogie. L'étude des n-grammes d'ordre supérieur (4, 5, *etc.*) aboutira-t-elle à trouver des propriétés semblables à celles qui existent entre trigrammes ? Ou bien conduira-t-elle à conclure en la singularité des trigrammes de mots vis-à-vis de ces propriétés ?

Enfin il aurait été souhaitable d'évaluer la performance de notre modèle de lissage par analogie dans une tâche de traduction par approche statistique. Par manque de temps nous n'avons pu mener cette étude à terme. Il aurait été souhaitable de mener cette évaluation sur un grand nombre de couples de langues voire l'intégralité des couples de langues du corpus Europarl utilisé dans notre évaluation de la technique de lissage par analogie. Comme il est connu que de simples réordonnements des hypothèses de traduction n'apporte généralement quasiment rien, l'intégration de notre technique de lissage aurait demandé carrément une modification du décodeur de Moses, tâche peu envisageable à quelques mois des échéances administratives. Redisons aussi que notre propos était de contribuer plus à la traduction par l'exemple qu'à la traduction statistique.

A

RÉIMPLANTATION D'*ALEPH* : *HOMOMORPHISM*

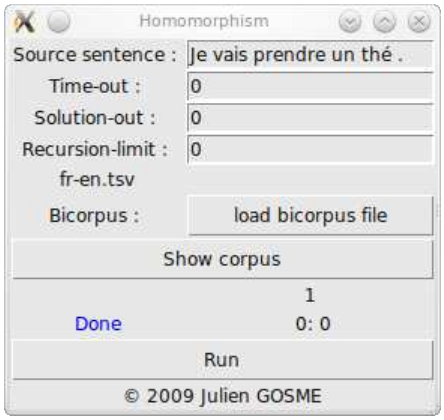
Homomorphism est une réimplantation en Python du système de traduction par l'exemple *Aleph*. Cette implantation est disponible à l'adresse suivante :

`http://jgosme.perso.info.unicaen.fr/homomorphism.html`.

Elle offre le choix du grain de traitement des termes des analogies : les termes peuvent être traités soit comme chaînes de caractères comme dans l'implantation originale, soit comme chaînes de mots.

La motivation principale de cette implantation est de proposer une visualisation des analogies manipulées en cours de traduction. Les analogies sont stockées sur deux tableaux noirs, un premier tableau noir pour les analogies en langue source et un second tableau noir pour les analogies en langue cible. Les traductions sont représentées dans une troisième structure. L'interface présente chacune de ces structures dans une fenêtre indépendante.

Pour illustrer l'utilisation d'*Homomorphism*, considérons la phrase « Je vais prendre un thé . » à traduire en anglais. Nous écrivons cette phrase dans la fenêtre de contrôle d'*Homomorphism* comme illustré ci-après.



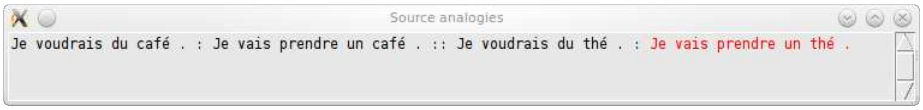
Nous chargeons alors le corpus d'exemples de traduction suivant :

```
Je voudrais du café .     ⟷ I'd like some coffee .
Je voudrais du thé .      ⟷ I'd like some tea .
Je vais prendre un café . ⟷ I'll have some coffee .
```

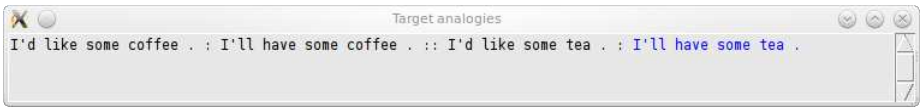
Les corpus sont de simples fichiers texte contenant une traduction par ligne. Chaque ligne est composée d'une phrase en langue source, d'une tabulation servant de séparateur (ici représentée par le symbole « ⟷ ») et de la traduction en langue cible de la première phrase.

Au lancement du traitement, les fenêtres inspectant l'état des structures internes sont affichées et mises à jour au fur et à mesure du traitement.

La première fenêtre, représentée ci-après, affiche les analogies en langue source et distingue la phrase à traduire (en rouge) des phrases du corpus d'exemple (en noir).

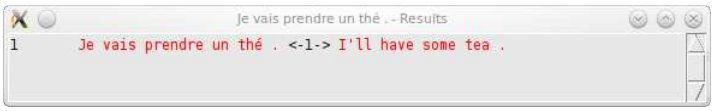


La deuxième fenêtre, représentée ci-dessous, affiche les analogies en langue cible et distingue la phrase obtenue par la résolution de l'équation analogique (en bleu) des autres phrases.



Enfin la troisième fenêtre, représentée ci-dessous, affiche les traductions induites par correspondance entre une analogie en langue source et une analogie en langue cible. Cette correspondance est établie dès lors que les trois premiers termes des analogies sont en relation de traduction.

À la fin du traitement, une dernière fenêtre affiche l'ensemble des traductions de la phrase source :



Grâce à cette visualisation des étapes de traitement, nous avons constaté que le système *Aleph* consacre la majorité de son temps de traitement à résoudre des analogies en langue source. C'est ce phénomène que nous désignons par « problème de la sélection des exemples » en section 1.2 et qui constitue le point de départ de cette thèse.

B

TABLEAUX DE SUFFIXES : *LINSUFFARR*

Nous avons été amené à manipuler des tableaux de suffixes dans nos travaux. Nous avons utilisé cette structure de données une première fois lors de la définition d'une méthode d'énumération des analogies d'un texte en section 2.5.2 afin de proposer l'initialisation par les premières analogies non triviales. Nous avons de nouveau utilisé cette structure de données lors de la proposition d'une méthode de lissage par analogie en section 4.4 afin d'implanter la reconstruction des trigrammes inconnus termes d'un patron d'analogie.

Les raisons invoquées en section 4.4 nous ont amené à implanter en Python la construction des tableaux de suffixes décrite par Kärkkäinen et coll. (2006). Nous avons ajouté le calcul des plus longs préfixes communs à cette implantation, puisque nous avions besoin de cette information pour la reconstruction des trigrammes inconnus par patron d'analogie.

DÉCOMPTE DE N-GRAMMES

Pour illustrer l'intérêt des tableaux de suffixes, considérons la tâche consistant à établir le décompte des bigrammes et 10-grammes d'un texte. La méthode naïve utilise une fenêtre glissante de taille n pour les n-grammes et met à jour le décompte des bigrammes au fur et à mesure. Le décompte des n-grammes avec les tableaux de suffixes est effectué en suivant l'algorithme décrit par Yamamoto et Church (1996). Cet algorithme met à profit le fait que tous les suffixes partageant le même préfixe, dans notre cas le même n-gramme, sont adjacents dans le tableau de suffixes et leur plus long préfixe commun est de longueur supérieure ou égale à l'ordre des n-grammes.

Nous comparons ces deux algorithmes pour effectuer le décompte des bigrammes et 10-grammes de mots contenus dans la partie anglaise du corpus Europarl (Koehn, 2005).[1] Pour comparer les temps de traitement, nous préparons des échantillons de ce corpus allant jusqu'à 4 millions de mots.

Les temps de traitement de ces deux algorithmes sont présentés en figure 21 pour les bigrammes et figure 22 pour les 10-grammes. L'algorithme utilisant un tableau de suffixes est nettement plus économe

1 Nous utilisons la version 5 de ce corpus, disponible à l'adresse `http://www.statmt.org/europarl/archives.html#v5`.

en temps de traitement, d'autant plus que l'ordre des n-grammes est grand. En effet, lors du décompte des 10-grammes de mots, nous pouvons remarquer un comportement quadratique de l'algorithme naïf. Pour la même tâche, l'algorithme utilisant un tableau de suffixe a un comportement linéaire. Cette première comparaison montre l'intérêt indéniable de cette structure de données pour manipuler des n-grammes.

IMPLANTATION DES TABLEAUX DE SUFFIXES EN PYTHON : *linsuffarr*

Notre implantation de l'algorithme de construction de tableau de suffixes de Kärkkäinen et coll. (2006) est écrite en Python et disponible à l'adresse suivante :

`http://jgosme.perso.info.unicaen.fr/linsuffarr.html`

Nous avons repris l'implantation en C de l'article original ainsi que l'implantation en Python *pysuffix*[2] de Romain Brixtel. Ces deux implantations incluent la construction du tableau de suffixes et le calcul du vecteur des plus longs préfixes communs. *Linsuffar* offre en plus la possibilité d'effectuer des recherches en lignes de commandes ainsi que d'ajouter des traits aux suffixes. D'un point de vue vitesse de construction, *pysuffix* et *linsuffarr* sont très proches. Romain Brixtel a mesuré un léger avantage[3] de *pysuffix* pour la vitesse de construction (quelques pourcents avec CPython) sur des textes composés d'une répétition d'un seul symbole (communication personnelle).

Afin de valider notre implantation, nous la comparons à l'algorithme de construction de tableau de suffixes de Larsson et Sadakane (2007) et plus précisément à deux implantations qui en ont été faites par Makoto Hiroi en Python.[4] Cet algorithme utilise astucieusement *quiksort* pour trier les suffixes selon le i-ième symbole à l'étape i du calcul. La première implantation *mksao* respecte exactement l'algorithme décrit dans l'article. La seconde implantation *mksa2* ajoute quelques améliorations pour le traitement des textes contenant de longues chaînes répétées.

Nous utilisons la partie commune du corpus Europarl en anglais, français, allemand et finnois comme texte d'entrée. Nous préparons des échantillons de 10 000, 20 000, 50 000, jusqu'à 10 000 000 octets par langue. En utilisant le même ordinateur, nous construisons le tableau de suffixes d'octets de chaque échantillon par *mksao*, *mksa2* et *linsuffar*. Le temps de construction obtenu par chaque implantation est tracé

2 Disponible à l'adresse suivante `http://code.google.com/p/pysuffix/` (dernier accés le 7 décembre 2011)

3 Voir le résultat de la comparaison de *pysuffix* et *linsuffarr* à l'adresse `http://code.google.com/p/pysuffix/#Bench_pysuffix_please`

4 Disponible à l'adresse `http://www.geocities.jp/m_hiroi/light/pyalgo43.html` (dernier accès le 16 mars 2011)

en figure 23. Pour les quatre langues, nous vérifions que *linsuffarr* construit les tableaux de suffixes en temps linéaire sur de véritables données. D'autre part, l'algorithme de Larsson et Sadakane (2007) a un comportement quasi-linéaire en temps. Cette comparaison donne du crédit à notre implantation puisqu'elle permet d'économiser entre 18 % et 30 % de temps par rapport à la meilleure implantation *mksa2* (cette économie atteint même 75 % par rapport à *mksa0*). Dans le pire des cas, en allemand, *linsuffarr* traite 26 000 octets par seconde.

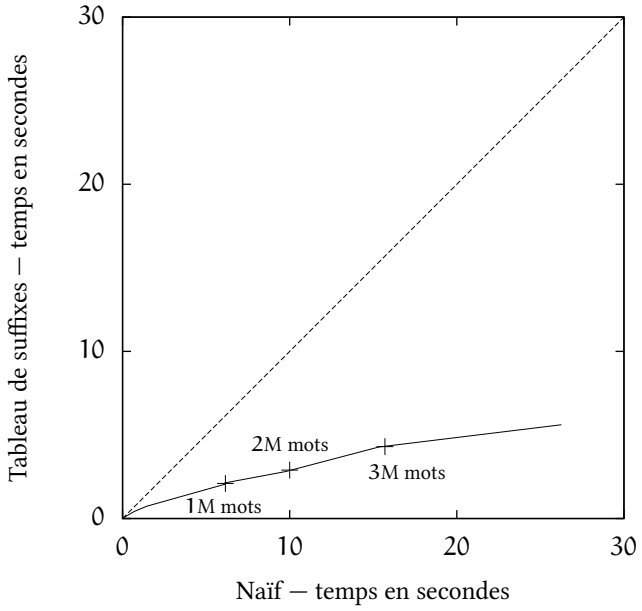


FIGURE 21 : Comparaison du temps de traitement de l'algorithme naïf de décompte des bigrammes et de l'algorithme utilisant un tableau de suffixes.

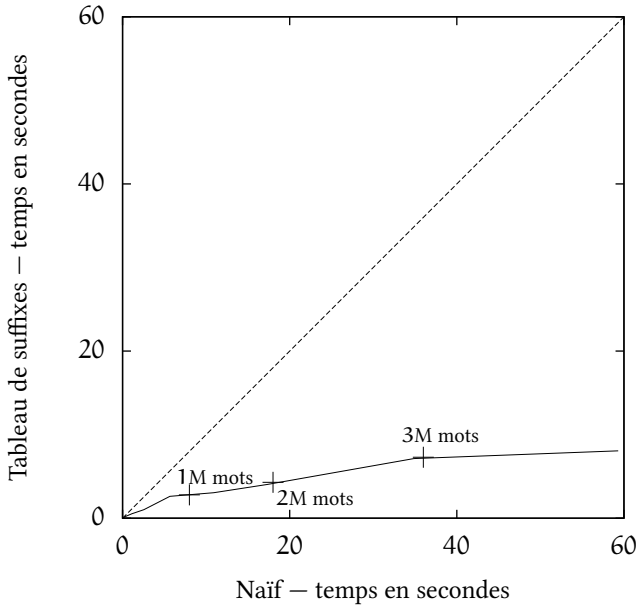


FIGURE 22 : Comparaison du temps de traitement de l'algorithme naïf de décompte des 10-grammes et de l'algorithme utilisant un tableau de suffixes.

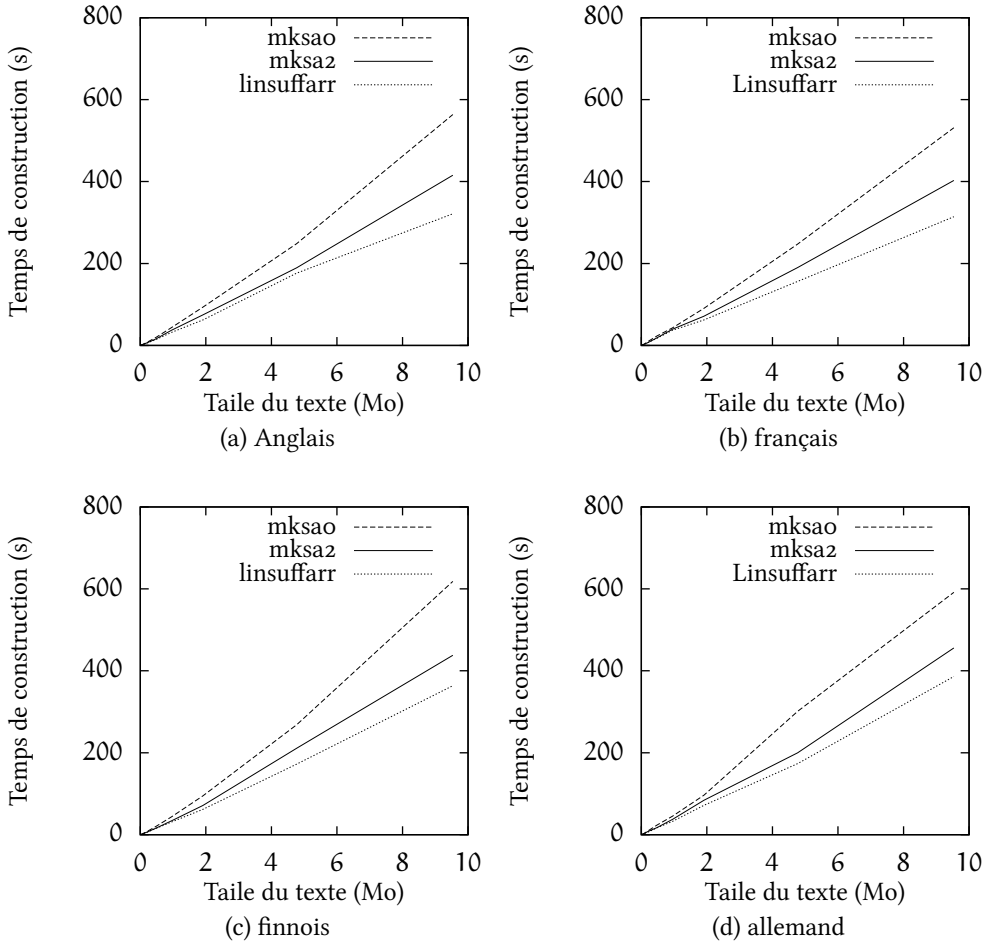


FIGURE 23 : Comparaison du temps de construction des tableaux de suffixes en quatre langues.

BIBLIOGRAPHIE

Boitet, C. 2007, « Corpus pour la TA : types, tailles et problèmes associés, selon leur usage et le type de système », *Revue Française de Linguistique Appliquée*, vol. 12, n° 1, p. 25–38. (Cité en page 34.)

Brown, P., V. Pietra, P. d. deSouza, J. Lai et R. Mercer. 1992, « Class-based *n*-gram models of natural language », *Computational linguistics*, vol. 18, n° 4, p. 467–479. (Cité en page 76.)

Brown, P. F., J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer et P. S. Roossin. 1990, « A statistical approach to machine translation », *Computational Linguistics*, vol. 16, n° 2, p. 79–85. (Cité en page 18.)

Chen, S. F. et J. Goodman. 1999, « An empirical study of smoothing techniques for language modeling », *Computer Speech and Language*, vol. 13, n° 4, p. 359–394. (Cité en pages 75, 78, 87 et 90.)

Claveau, V. et M. L'Homme. 2005a, « Structuring terminology using analogy-based machine learning », dans *Proceedings of the 7th International Conference on Terminology and Knowledge Engineering (TKE)*, Innsbruck, p. 17–18. (Cité en page 20.)

Claveau, V. et M.-C. L'Homme. 2005b, « Terminology by Analogy-Based Machine Learning », dans *Proceedings of the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*, Copenhague, p. 301–312. (Cité en page 42.)

Debili, F. et E. Souissi. 2005, « Y at-il une taille optimale des règles de succession intervenant dans l'étiquetage grammatical », dans *Actes de TALN 2005*, p. 363–372. (Cité en page 23.)

Denoual, E. 2007, « Analogical translation of unknown words in a statistical machine translation framework », dans *Proceedings of Machine Translation Summit XI*, Copenhague, p. 10–14. (Cité en page 42.)

Gale, W. A. et K. Church. 1994, « What's wrong with adding one? », dans *Corpus-Based Research into Language*, édité par N. O. de Haan et P., Language and Computers, Rodolpi, Amsterdam., p. 189–200. (Cité en page 75.)

Gale, W. A. et G. Sampson. 1995, « Good-Turing frequency estimation without tears », *Journal of Quantitative Linguistics*, vol. 2, nº 3, p. 217–237. (Cité en pages 86 et 90.)

Gosme, J. et Y. Lepage. 2011, « Structure des trigrammes inconnus et lissage par analogie », dans *Actes de TALN*, Montpellier, p. 345–356. (Cité en page 63.)

Harris, Z. 1968, *Mathematical structures of language*, Interscience publishers New York:, ISBN 0470353163, 230 p.. (Cité en page 26.)

Harris, Z., M. Gottfried, T. Ryckman, P. Mattick Jr, A. Daladier, T. Harris et S. Harris. 1989, *The form of information in science: analysis of an immunology sublanguage*, Kluwer Academic Publishers, 616 p.. (Cité en page 26.)

Hathout, N. 2009, « Acquisition morphologique à partir d'un dictionnaire informatisé », dans *Actes de la 16e Conférence Annuelle sur le Traitement Automatique des Langues Naturelles (TALN-2009)*, édité par D. e. P. T. Nazarenko, ATALA, p. 1–10. (Cité en page 42.)

Hutchins, J. 2005, « Towards a definition of example-based machine translation », dans *MT Summit X*, Phuket, p. 63–70. (Cité en page 18.)

Isabelle, P. 1987, « Machine translation at the TAUM group », *Machine Translation: The State of the Art*, p. 247–277. (Cité en pages 22 et 27.)

Kärkkäinen, J., P. Sanders et S. Burkhardt. 2006, « Linear work suffix array construction », *Journal of the ACM (JACM)*, vol. 53, nº 6, p. 918–936, ISSN 0004-5411. (Cité en pages 94, 103 et 104.)

Kittredge, R. 1987, « The significance of sublanguage for automatic translation », *Machine Translation: Theoretical and methodological issues*, p. 59–67. (Cité en page 26.)

Kneser, R. et H. Ney. 1995, « Improved backing-off for m-gram language modeling », dans *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, IEEE, ISBN 0780324315, ISSN 1520-6149, p. 181–184. (Cité en page 90.)

Koehn, P. 2005, « Europarl: A Parallel Corpus for Statistical Machine Translation », dans *Proceedings of the tenth Machine Translation Summit (MT Summit X)*, Phuket, p. 79–86. (Cité en pages 65 et 103.)

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin et E. Herbst. 2007, « Moses: Open Source Toolkit for

Statistical Machine Translation », dans *ACL 2007*, Prague, p. 177–180. (Cité en page 29.)

Lafourcade, M. 1994, *Génie logiciel pour le génie linguiciel*, thèse de doctorat, université Joseph Fourier, Grenoble 1. (Cité en page 32.)

Langlais, P. et A. Patry. 2007, « Translating Unknown Words by Analogical Learning », dans *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, p. 877–886. (Cité en page 42.)

Langlais, P. et F. Yvon. 2008, « Scaling up analogical learning », dans *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, Manchester, p. 51–54. (Cité en page 54.)

Langlais, P., F. Yvon et P. Zweigenbaum. 2008, « Analogical Translation of Medical Words in Different Languages », dans *GoTAL*, *Lecture Notes in Computer Science*, vol. 5221, Springer, Berlin, ISBN 978-3-540-85286-5, p. 284–295. (Cité en pages 20 et 42.)

Langlois, D., A. Brun, K. Smaïli et J.-P. Haton. 2003, « Événements impossibles en modélisation stochastique du langage », *Traitement Automatique des Langues*, vol. 44, n° 1, p. 33–61. (Cité en page 95.)

Larsson, N. et K. Sadakane. 2007, « Faster suffix sorting », *Theoretical Computer Science*, vol. 387, n° 3, p. 258–272. (Cité en pages 104 et 105.)

Lavallée, J.-F. et P. Langlais. 2010, « Analyse morphologique non supervisée par analogie formelle », dans *TALN 2010*, Montréal, p. 10 pages. (Cité en page 42.)

Lepage, Y. 2004a, « Analogy and formal languages », dans *Proceedings of the joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language*, vol. 53, Elsevier, ISSN 1571-0661, p. 180–191. (Cité en pages 41, 44 et 45.)

Lepage, Y. 2004b, « Lower and higher estimates of the number of "true analogies" between sentences contained in a large multilingual corpus », dans *Proceedings the 20th International Conference on Computational Linguistics (COLING)*, vol. 1, Genève, p. 736–742. (Cité en pages 20, 42 et 53.)

Lepage, Y. et E. Denoual. 2005a, « ALEPH: an EBMT system based on the preservation of proportional analogies between sentences across languages », dans *IWSLT 2005*, Pittsburgh, p. 47–54. (Cité en page 42.)

Lepage, Y. et E. Denoual. 2005b, « Purest ever example-based machine translation: detailed presentation and assessment », *Machine Translation Journal*, vol. 19, n° 3-4, p. 251–282. (Cité en pages 19, 21 et 29.)

Lepage, Y., J. Migeot et E. Guillerm. 2007, « Analogies of form between chunks in Japanese are massive and far from being misleading », dans *Proceedings of the 3rd Language & Technology Conference (LTC)*, Poznań, p. 503–507. (Cité en page 53.)

Leplus, T., P. Langlais et G. Lapalme. 2004, « Weather report translation using a translation memory », *Lecture notes in computer science*, p. 154–163, ISSN 0302-9743. (Cité en page 22.)

Levenshtein, V. I. 1966, « Binary codes capable of correcting deletions, insertions, and reversals », *Soviet Physics Doklady*, vol. 10, n° 8, p. 707–710. (Cité en page 29.)

Manber, U. et G. Myers. 1990, « Suffix arrays: A new method for on-line string searches », dans *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, San Francisco, p. 319–327. (Cité en page 52.)

Matsuzaki, T., Y. Miyao et J. Tsujii. 2003, « An efficient clustering algorithm for class-based language models », dans *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*, vol. 4, Association for Computational Linguistics, p. 119–126. (Cité en page 76.)

Moore, R. et C. Quirk. 2009, « Improved smoothing for N-gram language models based on ordinary counts », dans *Actes de ACL-IJCNLP 2009 Conference*, Association for Computational Linguistics, p. 349–352. (Cité en page 90.)

Och, F. et H. Ney. 2003, « A Systematic Comparison of Various Statistical Alignment Models », *Computational Linguistics*, vol. 29, n° 1, p. 19–51. (Cité en page 29.)

Och, F. J. 2003, « Minimum Error Rate Training in Statistical Machine Translation », dans *ACL 2003*, Association for Computational Linguistics, Sapporo, p. 160–167. (Cité en page 29.)

Papineni, K., S. Roukos, T. Ward et W. Zhu. 2002, « BLEU: a Method for Automatic Evaluation of Machine Translation », dans *ACL 2002*, p. 311–318. (Cité en page 29.)

Paul, M. 2008, « Overview of the IWSLT 2008 Evaluation Campaign », dans *Proceedings of the 5th International Workshop on Spoken Language Translation (IWSLT)*, Waikiki, p. 1–17. (Cité en pages 43 et 52.)

Rosenfeld, R. 2000, « Two decades of statistical language modelling: where do we go from here? », *Proceedings of the IEEE*, vol. 88, n° 8, p. 1270–1278. (Cité en page 74.)

Snover, M., B. Dorr, R. Schwartz, L. Micciulla et J. Makhoul. 2006, « A study of translation edit rate with targeted human annotation », dans *AMTA 2006*, p. 223–231. (Cité en page 36.)

Solan, Z., E. Ruppin, D. Horn et S. Edelman. 2003, « Automatic acquisition and efficient representation of syntactic structures », dans *Advances in Neural Information Processing Systems 15*, vol. 15, MIT Press, p. 91–98. (Cité en page 38.)

Somers, H. 1999, « Review article: Example-based machine translation », *Machine Translation*, vol. 14, n° 2, p. 113–157, ISSN 0922-6567. (Cité en page 18.)

Stolcke, A. 2002, « SRILM-an extensible language modeling toolkit », dans *Seventh International Conference on Spoken Language Processing*, vol. 3, p. 901–904. (Cité en pages 76 et 90.)

Stroppa, N. et F. Yvon. 2005, « An analogical learner for morphological analysis », dans *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, Ann Arbor, p. 120–127. (Cité en page 44.)

Takeya et Y. Lepage. 2011, « A study of the number of proprtionnal analogies between marker-based chunks in 11 European languages », dans *Procedings of the 5th Language and Technology Conference (LTC)*, Poznań, p. 284–288. (Cité en page 54.)

Turcato, D. et F. Popowich. 2003, « What is example-based machine translation », dans *Recent advances in example-based machine translation*, p. 59–82. (Cité en page 18.)

Turney, P. 2008, « A Uniform Approach to Analogies, Synonyms, Antonyms, and Associations », dans *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Coling 2008 Organizing Committee, Manchester, p. 905–912. (Cité en pages 19 et 42.)

Vauquois, B. et C. Boitet. 1985, « Automated Translation at Grenoble University », *Computational Linguistics*, vol. 11, n° 1, p. 28–36. (Cité en page 34.)

Weaver, W. 1955, « Translation », *Machine translation of languages*, vol. 14, p. 15–23. (Cité en page 17.)

Witten, I. H. et T. C. Bell. 1991, « The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression », *IEEE Transactions on Information Theory*, vol. 37, n° 4, p. 1085–1094. (Cité en page 90.)

Yamamoto, M. et K. Church. 1996, « Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus », dans *Proceedings of the 6th Workshop on Very Large Corpora*, Computational Linguistics. (Cité en page 103.)

Yvon, F. 2003, « Finite-State transducers solving analogies on words », cahier de recherche 2003, École Nationale Supérieure des Télécommunications, Informatique et Réseaux. (Cité en pages 41 et 44.)

Cette thèse a été composée avec LaTeX 2ε
en utilisant le style classicthesis.
La police principale est Libertine.

Résumé

Ce travail de thèse s'inscrit dans le cadre de la traduction automatique. En étudiant les fondements de la traduction automatique par l'exemple, et plus particulièrement le système *Aleph*, nous mettons en évidence le *problème de la sélection des exemples*. Le système *Aleph* emploie exclusivement l'analogie afin de produire de nouvelles phrases et de nouvelles traductions. Le problème est de sélectionner les phrases dans un grand corpus d'exemples afin de produire de nouvelles phrases par analogie. Notre premier apport consiste en l'élaboration d'une méthode permettant d'énumérer l'intégralité des analogies entre chaînes d'un texte. Cette méthode nous permet ensuite de mettre en œuvre une étude statistique des analogies les plus fréquentes entre trigrammes de mots et de mettre en évidence les patrons d'analogie les plus fréquents. Ces résultats permettent alors de concevoir une nouvelle méthode de lissage d'un modèle de langue trigramme basé sur un petit nombre d'analogies. Nos expériences montrent que cette méthode est très compétitive vis-à-vis des méthodes classiques.

Mots-clés : traitement automatique du langage naturel, traduction automatique, analogie, bilinguisme, problème de la sélection des exemples, grammaire par l'exemple, énumération des analogies, modèle de langue n-gramme

Titre en anglais :
« Complete Enumeration and Specific Detection of Proportional Analogies: Studies for Language Models and Machine Translation »

Abstract

The research presented in this PhD thesis is in the machine translation field. By studying the foundations of example-based machine translation, especially in the *Aleph* system, we bring to light the problem of example selection. The *Aleph* system uses exclusively the operation of analogy to produce new sentences and new translations. The problem is to select the adequate sentences from a large corpus of examples to allow for the production of new sentences by analogy. Our first contribution consists in the design of a method for the complete enumeration of all analogies contained in a text. This method allows us to complete a statistical study of the most frequent analogies between word trigrams and to bring to light the most frequent patterns of analogy. These results allow us to design a new smoothing technique for trigram language models based on a small amount of patterns of analogy. We report experiments which show that this new smoothing technique outperforms classical methods.

Keywords : natural language processing, machine translating, analogy, bilinguism