Extraction et recherche d'information en langage naturel dans les documents semi-structurés

Xavier Tannier

▶ **To cite this version:**

Xavier Tannier. Extraction et recherche d'information en langage naturel dans les documents semi-structurés. domain_stic.othe. Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006. Français. ⟨NNT : ⟩. ⟨tel-00121721⟩

HAL Id: tel-00121721

https://theses.hal.science/tel-00121721v1

Submitted on 21 Dec 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

N° d'ordre : 410 I

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

THÈSE

présentée par

Xavier TANNIER

pour obtenir le grade de
Docteur de l'Ecole Nationale Supérieure
des Mines de Saint-Etienne
et de l'Université Jean Monnet

spécialité informatique

Extraction et recherche d'information en langage naturel dans les documents semi-structurés

Soutenue à Saint-Etienne le 27 septembre 2006
en présence d'un jury composé de :

Brigitte GRAU	Maître de Conférence HDR, LIMSI-CNRS rapporteur
Yves CHIARAMELLA	Professeur, Université Joseph Fourier, Grenoble rapporteur
Sylvie LAINÉ-CRUZEL	Professeur, Université Jean Moulin, Lyon 3 examinatrice
François JACQUENET	Professeur, Université Jean Monnet, Saint-Etienne examinateur
Jean-Jacques GIRARDOT	Maître de Recherche, Ecole des Mines directeur de thèse
Mihaela JUGANARU-MATHIEU	Maître-Assistant, Ecole des Mines co-directrice de thèse
Shlomo GEVA	Senior Lecturer Queensland University of Technology, Brisbane, Australie invité

Table des matières

Introduction générale 1

I Recherche d'information, structure et langage naturel 7

1 Les documents semi-structurés 9

1.1 Le langage XML . . . 9

1.2 Du document plat au document structuré . . . 11

1.3 La sémantique du balisage . . . 12

1.3.1 Généralités . . . 12

1.3.2 Balises dures, balises de saut, balises transparentes . . . 14

2 Recherche d'information et structure 17

2.1 Introduction . . . 17

2.2 Les spécificités de la recherche d'information semi-structurée . . . 20

2.2.1 L'unité d'information pertinente . . . 20

2.2.2 Recherche sur contenu et structure . . . 21

2.2.3 Interprétation de la requête . . . 22

2.2.4 Bilan . . . 22

2.3 Indexation . . . 23

2.3.1 Pondération des termes . . . 24

2.3.2 Indexer le contenu et la structure . . . 24

2.4 Interrogation . . . 26

2.4.1 Modèles traditionnels . . . 27

2.4.2 Les langages de requêtes . . . 28

2.4.2.1 XPath . . . 29

2.4.2.2 NEXI . . . 31

2.4.3 Le traitement des requêtes . . . 32

2.4.3.1 Extension des modèles traditionnels . . . 32

2.4.3.2 Le système GPX . . . 33

2.5 Evaluation . . . 34

2.5.1 L'évaluation ou la dictature de l'humain . . . 35

2.5.2 Mesures d'évaluation des systèmes classiques . . . 36

2.5.3 Mesures d'évaluation des systèmes de RI structurée 38
2.5.4 La campagne d'évaluation INEX 38
2.5.4.1 Les documents utilisés par INEX en 2005 39
2.5.4.2 Les requêtes 39
2.5.4.3 Les jugements de pertinence 43
2.5.4.4 Les métriques d'évaluation 44
2.5.4.5 Les différentes tâches 46

3 Traitement automatique de la langue 47
3.1 Les différents niveaux du langage 48
3.2 Morphologie 49
3.2.1 Catégories grammaticales 49
3.2.2 Les autres informations morphosyntaxiques 50
3.3 Syntaxe 51
3.3.1 Du mot à la phrase 51
3.3.1.1 Constituants 51
3.3.1.2 Phrases 52
3.3.1.3 Les fonctions grammaticales 53
3.3.2 Les ambiguïtés syntaxiques 53
3.3.3 L'analyse syntaxique 55
3.3.3.1 L'analyse superficielle 55
3.3.3.2 Les grammaires hors-contexte 56
3.3.3.3 L'analyse en dépendance 57
3.3.3.4 Les traits 58
3.3.3.5 Les grammaires à clauses définies 59
3.3.3.6 Les formalismes plus évolués 60
3.4 Sémantique 61
3.4.1 La représentation du sens 61
3.4.1.1 La représentation logique 61
3.4.1.2 Un exemple de représentation, la DRT 62
3.4.1.3 Les événements 63
3.4.2 L'analyse sémantique 64
3.4.2.1 L'analyse profonde par compositionnalité 64
3.4.2.2 L'interprétation sémantique des relations grammaticales 66
3.4.2.3 Les grammaires sémantiques 67
3.4.2.4 Les patrons sémantiques 68
3.4.3 Les ambiguïtés sémantiques 68
3.4.3.1 L'anaphore 68
3.4.3.2 L'ellipse 70
3.4.3.3 La portée des quantificateurs 71

3.4.4 La sémantique lexicale . . . 72
3.4.5 Vers la pragmatique . . . 73
3.5 Le traitement de la langue dans la recherche d'information . . . 74
3.5.1 Variations morphologiques . . . 75
3.5.2 Variations syntaxiques . . . 75
3.5.3 Variations sémantiques . . . 76
3.6 Les interfaces de requêtes en langage naturel pour les bases de données . 76
3.6.1 Intérêts . . . 77
3.6.2 Architectures . . . 78
3.6.2.1 Les patrons sémantiques . . . 78
3.6.2.2 Les grammaires sémantiques . . . 79
3.6.2.3 Les langages de représentation intermédiaires . . . 80
3.6.3 Inconvénients et limites des interfaces . . . 81
3.7 Les interfaces en langage naturel pour la recherche d'information semi-structurée . . . 83
3.7.1 Motivation . . . 83
3.7.1.1 Complexité des langages de requêtes . . . 83
3.7.1.2 Connaissances de la structure . . . 84
3.7.1.3 Les collections hétérogènes . . . 84
3.7.1.4 Une utilisation "intelligente" de la structure . . . 85
3.7.2 Ce que les ILN pour XML ne sont pas . . . 85
3.7.2.1 Une technique de plus pour la recherche d'information . 85
3.7.2.2 Des interfaces pour les bases de données . . . 85
3.7.2.3 Des systèmes de question/réponse . . . 87
3.7.3 La tâche "Langage Naturel" d'INEX . . . 87
3.7.4 Les premières approches . . . 88
3.7.4.1 Pré-traitement . . . 88
3.7.4.2 Analyse . . . 89
3.7.4.3 Formulation de la requête NEXI . . . 89
3.7.4.4 Limites . . . 90

II Extraction et recherche d'information en langage naturel dans les documents semi-structurés 91

4 Les contextes de lecture 93
4.1 Problématique . . . 94
4.1.1 Rattachement des mots . . . 94
4.1.2 Proximité logique . . . 95
4.1.3 Traitement automatique du langage . . . 96
4.2 Le contexte de lecture . . . 97

4.2.1 Définition . . . 97
4.2.2 Contexte de lecture et classification des balises . . . 98
4.2.2.1 Balises transparentes . . . 98
4.2.2.2 Balises de saut . . . 99
4.2.2.3 Balises dures . . . 100
4.2.2.4 Commentaires . . . 100
4.3 Classification automatique . . . 101
4.3.1 Description de l'approche . . . 101
4.3.1.1 Approche globale et nouvelles définitions . . . 101
4.3.1.2 Analyse syntaxique . . . 102
4.3.1.3 Algorithme de détermination automatique . . . 103
4.3.1.4 Expérimentations . . . 105
4.3.2 Résultats . . . 106
4.3.2.1 Commentaires préliminaires . . . 106
4.3.2.2 Balises transparentes . . . 106
4.3.2.3 Balises de saut . . . 108
4.3.2.4 Résultats finaux . . . 108
4.3.2.5 Portée . . . 109
4.3.3 Taille du corpus . . . 109
4.4 L'outil XGTagger . . . 111
4.5 Conclusion . . . 111

5 Analyse des requêtes en langage naturel 113
5.1 Enjeux, problèmes et choix . . . 114
5.1.1 Les contraintes . . . 114
5.1.2 Les libertés . . . 115
5.1.3 L'heure du choix . . . 116
5.1.3.1 Les accessits . . . 116
5.1.3.2 Le premier prix . . . 117
5.2 Analyse syntaxique . . . 118
5.2.1 Les besoins . . . 118
5.2.2 La représentation . . . 118
5.2.3 Avant l'analyse : pré-traitement des syntagmes nominaux . . . 119
5.2.3.1 Les syntagmes nominaux simples . . . 120
5.2.3.2 Les syntagmes nominaux complexes . . . 121
5.2.4 L'analyse . . . 122
5.2.4.1 Le fonctionnement général . . . 123
5.2.4.2 L'attribution des relations grammaticales . . . 125
5.2.4.3 La négation . . . 126
5.2.4.4 Les anaphores . . . 127

5.2.4.5 Robustesse . . . 127
5.3 Règles sémantiques . . . 128
5.3.1 Références explicites à la structure . . . 129
5.3.1.1 Synonymie . . . 130
5.3.1.2 Polysémie et conceptémie . . . 130
5.3.1.3 Représentation des éléments structurels . . . 132
5.3.2 Représentation des règles . . . 132
5.3.3 Règles fixes . . . 133
5.3.3.1 Détection de la cible de la requête . . . 134
5.3.3.2 Autres liens verbaux . . . 134
5.3.3.3 Liens prépositionnels . . . 136
5.3.3.4 Exemples d'application . . . 137
5.3.4 Règles dépendant de la structure . . . 138
5.3.4.1 Liens verbaux . . . 139
5.3.4.2 Liens prépositionnels . . . 141
5.3.4.3 Autres règles . . . 142
5.3.5 Etablissement d'un modèle de la collection . . . 142
5.3.5.1 Le modèle . . . 142
5.3.5.2 Les ambiguïtés du modèle . . . 144
5.3.6 La désambiguïsation par la sémantique . . . 145
5.3.6.1 Nombre de règles utilisées . . . 145
5.3.6.2 Elément ou texte ? . . . 145
5.3.6.3 Elément et contenu . . . 148
5.3.6.4 Les autres ambiguïtés . . . 148
5.4 Obtention d'une requête formelle . . . 149
5.4.1 Disposition des éléments structurels . . . 149
5.4.1.1 Cible . . . 149
5.4.1.2 Support . . . 149
5.4.1.3 Eléments subordonnés . . . 149
5.4.2 Le contenu . . . 149
5.4.2.1 Le cas général . . . 150
5.4.2.2 Les syntagmes nominaux simples . . . 151
5.4.2.3 Traitement des relations . . . 152
5.4.2.4 Le texte non relié à un élément . . . 152
5.4.2.5 Les disjonctions, conjonctions et négations . . . 152
5.5 Conclusion . . . 153

6 Utilisation de la structure pour l'amélioration de la requête 155
6.1 Recherche contextuelle . . . 156
6.1.1 Problématique . . . 156

6.1.2 Modification des requêtes pour la recherche contextuelle 157
6.1.2.1 Aspects théoriques 158
6.1.2.2 Quelle recherche, quel contexte ? 158
6.1.2.3 Premières réponses pratiques 159
6.2 Les liens entre documents 160
6.2.1 De nouvelles règles sémantiques 160
6.2.2 Le rôle du langage formel 161
6.3 Etude concernant la taille des éléments à retourner 162
6.4 Le langage XOR 162
6.4.1 Limites de NEXI 163
6.4.2 Description du langage XOR 164
6.4.2.1 Opérateur de négation 164
6.4.2.2 Requêtes multiples 165
6.4.2.3 Extension des chemins et des mots-clés 165
6.4.2.4 Prédicats additionnels 166
6.4.3 Implémentation 166
6.5 Construire des requêtes avec le langage XOR 166
6.5.1 Recherche contextuelle 166
6.5.1.1 Les relations prépositionnelles 167
6.5.1.2 La négation 167
6.5.2 Autres traitements 168
6.5.2.1 Le prédicat *contains* 168
6.5.2.2 Les liens entre documents 168
6.5.2.3 La taille des éléments 169
6.5.2.4 Les ambiguïtés du modèle 169
6.6 Conclusion 170

7 Expérimentations et résultats 171
7.1 Les conditions d'expérimentations 171
7.1.1 La collection INEX 2005 171
7.1.2 La tâche CO+S 172
7.1.3 La tâche CAS 172
7.1.4 La méthode d'évaluation 173
7.1.5 Les métriques d'évaluation 173
7.2 Traduction en NEXI 173
7.2.1 Traduction simple 173
7.2.1.1 Les requêtes CAS 173
7.2.1.2 Les requêtes CO+S 174
7.2.1.3 Le regroupement des groupes nominaux simples 176
7.2.1.4 La désambiguïsation syntaxique par les règles sémantiques 179

7.2.1.5 Conclusion . . . 180
7.2.2 NEXI et recherche contextuelle . . . 182
7.2.3 Les résultats officiels de la campagne INEX 2005 . . . 182
7.2.3.1 Les requêtes CAS . . . 183
7.2.3.2 Les requêtes CO+S . . . 183
7.2.3.3 Conclusion . . . 183
7.3 Expérimentations avec le langage XOR . . . 187
7.3.1 Les requêtes multiples . . . 187
7.3.2 Autres . . . 188
7.4 Quelques exemples d'analyse . . . 188
7.4.1 Topic 141. . . . 188
7.4.2 Topic 156. . . . 189
7.4.3 Topic 162. . . . 190
7.4.4 Topic 164. . . . 190
7.5 Conclusion . . . 193

Conclusion générale **197**

Annexes **199**

A Le langage XML **203**
A.1 Les bases de XML . . . 203
A.2 La DTD . . . 203
A.3 XLink . . . 204

B Traitement automatique de la langue **207**
B.1 Les catégories grammaticales . . . 207
B.1.1 Aperçu rapide . . . 207
B.1.2 Les étiquettes de la *Penn Treebank* . . . 207
B.2 Quelques éléments de grammaire . . . 208
B.2.1 Les groupes nominaux . . . 208
B.2.2 Les groupes verbaux . . . 209
B.2.3 Les propositions relatives . . . 209
B.2.4 La phrase . . . 209
B.2.5 Les abréviations utilisées . . . 210
B.3 L'analyse sémantique et le lambda-calcul . . . 210

C XGTagger **213**
C.1 Description . . . 213
C.2 Exemples . . . 213
C.2.1 Analyse morphosyntaxique . . . 213

C.2.2 Enrichissement lexical 215
C.2.3 Analyse syntaxique 216
C.2.4 Regroupement des contextes de lecture 218
C.2.5 Conclusion 218
C.3 Mise en œuvre 219
C.3.1 Contraintes 219
C.3.2 Fonctionnement 220

D Interfaces en langage naturel **223**
D.1 Contenu seulement 223
D.2 Contenu et structure 225

Glossaire **229**

Bibliographie **233**

Remerciements

La thèse est une école, une decouverte, une aventure, un tremplin aussi. Ce mémoire en est la conclusion, et cette page, malgré sa position, le point final.

Je tiens à remercier mes deux rapporteurs, Brigitte Grau et Yves Chiaramella, dont les remarques (toujours pertinentes), les conseils (toujours bienvenus) et la gentillesse (qui ne gâte rien) m'ont été utiles à tous points de vue. Les avoir comme rapporteurs fut un honneur et un plaisir.

Un grand merci également à Sylvie Lainé-Cruzel et François Jacquenet, qui ont accepté de consacrer une partie de leur été à l'examen de ma thèse, qui ne valait probablement pas un bon roman. J'ai été heureux de les compter parmi les membres de mon jury.

Je remercie également mon directeur Jean-Jacques Girardot et ma co-directrice Mihaela Mathieu pour m'avoir accueilli, pour avoir encadré mes recherches et pour la confiance qu'ils ont su me porter. Merci aussi à l'ensemble de l'équipe COCRI et du département RIM.

Je ne remercierai jamais assez Shlomo Geva. Il faut savoir être bref, et je me contenterai de dire que sans lui, sans son enthousiasme et son désintéressement, ce mémoire aurait été beaucoup plus court. J'ai l'audace de penser que cela aurait été dommage. Mon seul regret est de n'avoir pas pu profiter de notre collaboration pour visiter l'Australie.

La thèse n'est pas qu'une aventure scientifique. Elle fut également l'occasion de rencontrer des gens différents, de tous horizons et toutes nationalités, toutes ambitions, tous caractères. Citer des noms ne servirait qu'à en oublier, et j'espère qu'ils savent, de toute façon.

Merci enfin à ceux qui ne m'ont ni renié ni oublié pendant mon exil à Saint-Etienne, de ma famille à mes amis. Par ailleurs, je me refuse à sombrer dans le mélo, mais je n'en pense pas moins.

Résumé

La recherche d'information (RI) dans des documents semi-structurés (écrits en XML en pratique) combine des aspects de la RI traditionnelle et ceux de l'interrogation de bases de données. La structure a une importance primordiale, mais le besoin d'information reste vague. L'unité de recherche est variable (un paragraphe, une figure, un article complet...). Par ailleurs, la flexibilité du langage XML autorise des manipulations du contenu qui provoquent parfois des ruptures arbitraires dans le flot naturel du texte.

Les problèmes posés par ces caractéristiques sont nombreux, que ce soit au niveau du pré-traitement des documents ou de leur interrogation. Face à ces problèmes, nous avons étudié les solutions spécifiques que pouvait apporter le traitement automatique de la langue (TAL). Nous avons ainsi proposé un cadre théorique et une approche pratique pour permettre l'utilisation des techniques d'analyse textuelle en faisant abstraction de la structure. Nous avons également conçu une interface d'interrogation en langage naturel pour la RI dans les documents XML, et proposé des méthodes tirant profit de la structure pour améliorer la recherche des éléments pertinents.

Notes

Les termes marqués d'une étoile * sont définis dans le glossaire, à la page 229, et les numéros entre crochets font référence à la bibliographie. Ces marques, ainsi que d'autres éléments, sont en couleurs et "cliquables" dans la version électronique.

Index

étiquetage morphosyntaxique, 50

adjectif, 207
adverbe, 207
ambiguïtés
    sémantiques, 68
    syntaxiques, 54
analyse en dépendance, 57
analyse profonde, 55
analyse superficielle, 55
anaphore, 69, 127
approche
    orientée document, 12
    orientée données, 12
auto-explicativité, 21, 85, 156
auxiliaire, 207

balisage, 11
balise
    de saut, 15, 99, 101
    dure, 14, 100, 101
    transparente, 15, 98, 101
bases de données, 85

Cass, 55, 102
catégorie grammaticale, 49
CFG, 56
chevauchement, 38
collections hétérogènes, 84
conceptémie, 131
conjonction, 207
constituant, 51
contenu et structure, 21, 39, 42
contenu seul, 21, 39, 42
contexte de lecture, 97

déterminant, 207
DCG, 59
document
    plat, 11
    semi-structuré, 11
    structuré, 11
données *vs.* info, 17
doxel, 20

DRS, 62
DRT, 62

ellipse, 70
exhaustivité, 44
extraction d'information, 20

flexion, 51
focus, 70
fonction d'appariement, 27
fonction grammaticale, 53

GPX, 33, 166, 173, 178
grammaire à clauses définies, 59
grammaire hors-contexte, 56
grammaires sémantiques, 67, 79, 116

holonymie, 72
hyponymie, 72

ief, 24
indexation, 23
INEX, 38
    NLQ2NEXI, 87
interrogation
    stricte, 85
    vague, 86

lambda-calcul, 210
langage naturel, 47
lemmatisation, 51
lemme, 51

méronymie, 72
métriques, 36, 38, 44
    ep/gr, 45
    MAep, 45
    nxCG, 45
    précision, 37
    rappel, 37
marquage, 11
modèle
    booléen, 27
    de document, 23
    de requête, 26

de RI, 18
probabiliste, 27
vectoriel, 27
morphologie, 48, 49
variations morphologiques, 75

NEXI, 31
NLQ2NEXI, 182
nom, 207

patrons sémantiques, 78
pertinence, 36
polysémie, 72
pooling, 36
préposition, 207
pragmatique, 48, 73
pronom, 207
proximité logique, 95

question/réponse, 20, 87

référent, 62
réification, 64
résolution des pronoms, 70
résumé automatique, 20
rôle thématique, 63, 66
recherche conditionnelle, 158
recherche contextuelle, 158
recherche d'information, 18
relations grammaticales, 66, 117
requête, 18
cible, 31
support, 31

sémantique, 48, 61
ambiguïtés, 68
analyse sémantique, 64
sémantique lexicale, 72
variations sémantiques, 76
sous-arbres imbriqués, 25
spécificité, 44
stemming, 75
synonymie, 72
syntagme, 51
syntaxe, 48, 51
ambiguïtés, 54
analyse profonde, 55
analyse superficielle, 55
grammaires hors-contexte, 56
grammaires à clauses définies, 59
variations syntaxiques, 75

tête, 51
template matching, 68, 116
tf - idf, 24
topic, 39
traits, 58

unité d'information, 20
unités disjointes, 25

verbe, 207
intransitif, 209
transitif, 209

WordNet, 73

XGTagger, 111
XML, 9
élément, 9
attribut, 9
balise, 9
commentaires, 203
DTD, 203
feuilles, 9
hiérarchie, 9
racine, 9
XLink, 204

Introduction générale

"Quand on proclama que la Bibliothèque comprenait tous les livres, la première réaction fut un bonheur extravagant. Tous les hommes se sentirent maîtres d'un trésor intact et secret."

"Sur quelque étagère de quelque hexagone, raisonnait-on, il doit exister un livre qui est la clef et le résumé parfait de tous les autres *: il y a un bibliothécaire qui a pris connaissance de ce livre et qui est semblable à un dieu. Dans la langue de cette zone persistent encore des traces du culte voué à ce lointain fonctionnaire. Beaucoup de pélerinages s'organisèrent à sa recherche, qui un siècle durant battirent vainement les plus divers horizons. Comment localiser le vénérable et secret hexagone qui l'abritait ? [...] Il est certain que dans quelque étagère de l'univers ce livre total doit exister."*

"Comme tous les hommes de la Bibliothèque, j'ai voyagé dans ma jeunesse ; j'ai organisé des pélerinages à la recherche d'un livre et peut-être du catalogue des catalogues."

Jorge Luis Borges, *La Bibliothèque de Babel*

Dans cette nouvelle étourdissante de Borges, cette Bibliothèque, qui contient *tous* les livres, des chaînes de caractères sans aucun sens jusqu'au volume qui, pense-t-on, délivre le sens de la vie, en passant par l'histoire complète de l'existence et de la mort de chacun d'entre nous, cet ensemble fini mais immense gardera tous ses secrets[1].

Dans la vie réelle, une masse de données moins impressionnante, mais néanmoins de plus en plus considérable, et ayant pour sa grande majorité plus de sens, est désormais disponible, et cela pour un nombre croissant d'individus. Et s'il est heureux que le sens de la vie (ou le texte de ce mémoire) ne puisse être retrouvé par une simple exploration combinatoire de chaînes de caractères, les nécessités plus terre-à-terre de la vie quotidienne imposent de fournir des mécanismes efficaces pour rechercher de l'information utile dans un ensemble de documents.

Les enjeux

La recherche d'information automatisée permet à un utilisateur de formuler un *besoin d'information*, à l'aide d'une *requête*, pour obtenir une réponse issue d'un ensemble de documents. L'idée est apparue dès la naissance des premiers ordinateurs. Elle s'est développée dans les années 1960 avec la possibilité de stocker et d'analyser des masses importantes de données. Depuis, avec la gigantesque augmentation des connaissances produites et conservées numériquement, elle est devenue un secteur stratégique pour

[1] La Bibliothèque de Babel est composée de l'ensemble des livres formés par toutes les combinaisons de caractères appartenant à un alphabet fini, ces combinaisons ayant elles-mêmes une longueur maximale. L'ensemble obtenu est donc fini également ; il suffit pourtant de mettre bout à bout plusieurs ouvrages pour obtenir n'importe quel texte plus long.

beaucoup d'entreprises et a indiscutablement bouleversé, par l'intermédiaire de l'Internet, le rapport de tout un chacun avec la connaissance. Avec "toujours plus d'aiguilles dans toujours plus de foin" [100], mais aussi avec toujours plus d'utilisateurs, les enjeux de la recherche d'information sont devenus considérables.

Un autre aspect important, et relativement nouveau, est que ces textes sont de plus en plus formulés de façon structurée. Cette structure est formalisée **explicitement** dans le texte lui-même par des *balises*, à l'aide de langages de représentations spécifiques. Parmi ces langages, XML (pour eXtended Markup Language) est désormais devenu un standard universellement utilisé. Il permet l'échange et le stockage de l'information avec une flexibilité qui l'a rendu extrêment populaire. Les documents obtenus sont dits *semi-structurés*, car ils sont en quelque sorte un intermédiaire entre les documents classiques et les bases de données.

Effectuer une recherche d'information sur ces nouveaux supports induit de nouvelles problématiques, auxquelles la recherche doit apporter de nouvelles solutions. Notamment, la présence de la structure entraîne deux nouvelles exigences : d'une part, celle de retourner des unités d'information plus petites (et donc plus précises) qu'un document entier, si le besoin s'en fait sentir, et d'autre part, celle de permettre la prise en compte de la structure dans la requête. Ceci a conduit à l'élaboration de *langages de requêtes* structurés spécifiques à ce domaine.

Par ailleurs, un système de recherche d'information, au fonctionnement immuable, déterministe et dénué d'émotions, a pour particularité d'avoir affaire en entrée et en sortie à un être humain subjectif, versatile et peu enclin à se laisser dicter un comportement par les limites d'une machine. La réduction nécessaire et pourtant impossible de ce "choc des cultures" est somme toute ce qui occupe et agrémente le quotidien des chercheurs du domaine. Parmi les attentes des utilisateurs, l'une des plus importantes, dès que le langage de requêtes devient complexe[1], est de pouvoir formuler leur besoin d'information en langage *naturel*, c'est-à-dire dans leur langage de tous les jours.

Pour résoudre ce problème, on ne tente pas encore de combler le fossé entre la machine et l'humain ; le moteur de recherche doit toujours recevoir une requête exprimée dans un langage *formel* (c'est-à-dire possédant une syntaxe et une sémantique définies à l'avance et non ambiguës, par opposition au langage naturel). L'idée est plutôt de construire un pont, une *interface*, capable de traduire du mieux possible les requêtes en langage naturel vers un langage formel. Les recherches dans ce domaine ont été intenses pour l'interrogation des bases de données. Cependant la problématique est sensiblement différente lorsque l'on parle de recherche d'information d'une part, et de documents semi-structurés d'autre part.

Le contexte

Le traitement automatique de la langue (TAL) recouvre de nombreuses applications dans les domaines de l'accès à l'information. Notamment, la recherche d'information en fait grand usage : extension de la requête par les synonymes ou les mots apparentés, traitement des groupes nominaux, lemmatisation... Parallèlement, de nombreuses interfaces d'accès aux informations contenues dans les bases de données ont été conçues. Ces interfaces en langage naturel permettent des manipulations de données évoluées mais très dépendantes d'un domaine précis d'application, et sont souvent peu robustes.

En recherche d'information (RI) traditionnelle (sur des documents non structurés),

[1] Un langage de requêtes *non complexe* est par exemple celui consistant en une simple suite de mots-clés. C'est le cas pour la recherche simple proposée par les moteurs de recherche sur l'Internet.

le besoin d'une assistance à la recherche par une interface en langage naturel se fait peu ressentir. En effet, les requêtes sont constituées d'une liste de mots-clés relativement aisée à construire ; le confort procuré par la possibilité de formuler des phrases complètes ne compenserait pas les imprécisions allant de pair avec leur traitement linguistique. De plus, une bonne connaissance du comportement d'un moteur de recherche permet d'ajouter et ou modifier des mots-clés selon le but que l'on cherche à atteindre, flexibilité que, paradoxalement, n'autorise que difficilement une expression en langage naturel.

L'hypothèse qui a gouverné le début de nos travaux est que la situation est radicalement différente dans le cadre de la recherche d'information dans des documents semi-structurés. Tout d'abord, les requêtes formelles devenant longues et complexes, les avantages apportés par la possibilité de formuler son besoin d'information dans une langue naturelle sont plus importants. Ensuite, dans le cas où les documents dans lesquels on recherche l'information sont de natures diverses, et en particulier de structurations diverses, chaque structure différente doit faire l'objet d'une requête différente. Effectuer son interrogation à un niveau plus conceptuel, puis générer automatiquement les formes adéquates, peut simplifier fortement ce problème. L'avantage procuré est double, puisqu'il évite à la fois l'effort d'écrire de nombreuses requêtes et celui de retenir autant de structures. Enfin, même en considérant comme bien connus le langage de requêtes et la structure des documents, plusieurs paramètres nouveaux viennent compliquer la tâche de l'utilisateur devant décrire son besoin d'information : les éléments potentiellement pertinents sont de tailles variables, ils sont également imbriqués les uns dans les autres (par exemple, une figure dans un chapitre, un chapitre dans un livre). De plus, les élément sont compris dans un ensemble faisant un tout et, par conséquent, ne sont plus *auto-explicatifs*, c'est-à-dire que la connaissance du contexte est indispensable pour les comprendre. Ces particularités ont pour résultat une élaboration de la requête qui n'est pas toujours en adéquation avec le fonctionnement du moteur de recherche qui la traite.

Notre contribution

Dans notre travail, nous n'avons pas cherché à utiliser les techniques "classiques" du TAL en RI, pour les appliquer aux documents semi-structurés, mais plutôt à identifier de quelle manière les problèmes posés en RI par les spécificités des documents semi-structurés pouvaient trouver une solution grâce au TAL.

Cette réflexion a tout d'abord apporté des réponses dans le cadre du pré-traitement des documents XML, avec une classification automatique des éléments structurels. En effet, les balises insérées dans le texte pour le structurer, outre qu'elles ont chacune leur signification propre, ne sont pas toutes traitées de la même façon par un lecteur humain. Notamment, certaines, comme les indications de forme (gras, italiques, etc.) n'interrompent pas le texte, tandis que d'autres (comme les notes de bas de page, les commentaires) indiquent une discontinuité dans l'énoncé. Il est possible de distinguer trois classes principales de balises dénotant chacune un comportement spécifique lors de la lecture. Nous nous sommes intéressé à la détermination automatique de ces trois classes en fonction de la structure des documents, ainsi qu'à la réalisation d'un outil utilisant cette classification pour permettre l'application des procédés habituels d'analyse textuelle à des documents semi-structurés.

En outre, le sujet que nous avons exploré en priorité est la réalisation d'une interface permettant d'analyser des requêtes exprimées en langage naturel et de les traduire dans un langage de requêtes formel. Nous avons montré que la première phase, l'analyse de la requête elle-même, pouvait être effectuée de façon très robuste et assez efficace, étant

données les simplifications autorisées par le contexte. Notons que ces simplifications ne consistent pas en une restriction des constructions linguistiques autorisées, mais plutôt en une certaine flexibilité concernant les informations recueillies par l'analyseur. De plus, la résolution des ambiguïtés syntaxiques, qui représente un problème majeur en analyse automatique de textes, est facilitée, pour les plus importantes (en ce qui concerne notre application), par les informations structurelles des documents.

Quant à la transformation en un langage formel, elle est l'occasion de prendre en compte des caractéristiques propres aux documents XML, voire propres à une structure donnée, éventuellement en formulant des requêtes bien plus complexes que celles que pourrait concevoir un être humain, même expert. Ainsi, il est possible, grâce à certaines constructions employées dans une interrogation en langage naturel, de distinguer les mots-clés faisant partie de la cible de la requête (par exemple, un paragraphe recherché), ou de son contexte (par exemple, l'article contenant ce paragraphe).

Il est ainsi possible de transformer un besoin d'information exprimé en langage naturel en une requête formelle analysable par un moteur de recherche, mais aussi de participer activement à la recherche elle-même. L'interface n'est alors plus seulement un module indépendant que l'on branche au système de recherche d'information, mais devient un **composant à part entière du processus de recherche**.

Les expérimentations concernant ce type d'interfaces sont facilitées par le fait que de tels systèmes sont capables de générer une requête dans un langage déjà existant. Les résultats que nous obtenons ainsi sont très bons, mais mettent également en évidence une perte d'information importante entre, d'une part, la représentation que l'on peut générer à l'issue de l'analyse linguistique d'une requête, et, d'autre part, les capacités d'expression des langages formels utilisés.

C'est pourquoi une autre voie de recherche a consisté en la création d'un nouveau langage de requêtes, plus adapté à nos besoins. Le caractère générique de l'interface en pâtit, puisque seul un moteur de recherche capable d'interpréter notre nouveau langage pourra être utilisé, mais les performances de recherche sont améliorées. Nous avons également participé à l'adaptation d'un moteur de recherche pour le rendre capable de traiter des requêtes exprimées dans ce nouveau langage.

Organisation du mémoire

Ce document est divisé en deux parties. La première est une présentation détaillée du contexte de recherche de nos travaux. La seconde décrit la contribution que nous venons d'exposer brièvement.

Recherche d'information, structure et langage naturel

Cette première partie comprend trois chapitres. Le premier, relativement succinct, est une présentation générale des documents semi-structurés et du langage XML. La problématique de la sémantique des balises y est abordée (section 1.3) et la classification entre balises dures, balises transparentes et balises de saut y est présentée (section 1.3.2).

Le deuxième chapitre est une synthèse concernant la recherche d'information (RI), orientée vers la discipline spécifique qui nous intéresse, à savoir la recherche d'information dans les documents semi-structurés. Les spécificités de celle-ci sont exposées (section 2.2) ; il s'agit tout particulièrement des différences concernant l'unité d'information pertinente (2.2.1), la recherche sur la structure (2.2.2) et l'interprétation de la requête (2.2.3). Par la suite, trois phases distinctes sont explorées, à commencer par

l'indexation des documents (section 2.3), qu'il est nécessaire de bien comprendre, notamment pour construire les requêtes automatiques les plus adaptées possibles. Puis la phase d'interrogation elle-même est décrite (section 2.4), à travers un survol des modèles traditionnels (2.4.1), des langages de requêtes (2.4.2) et du traitement de ces requêtes dans le cadre de la RI semi-structurée (2.4.3). Enfin les méthodes d'évaluation de systèmes de recherche d'information, que nous utiliserons nous-mêmes pour tester notre approche, sont exposées (section 2.5), et le chapitre se termine avec une présentation de la campagne INEX (INitiative for the Evaluation of XML Retrieval) à la section 2.5.4.

Dans le chapitre 3, nous donnons un aperçu circonstancié des techniques du traitement automatique de la langue (TAL). Nous distinguons dans un premier temps les différents niveaux d'étude du langage (section 3.1), avant de consacrer trois sections à ceux qui nous intéressent tout particulièrement dans notre travail : la morphologie (section 3.2), la syntaxe (section 3.3) et la sémantique (section 3.4). Dans chacune de ces sections, loin d'offrir une description exhaustive, nous nous attachons à présenter les enjeux et les techniques dont la connaissance est utile pour la compréhension de notre travail. Nous nous orientons ensuite vers les approches plus spécialisées dans le traitement de l'information, en évoquant le traitement de la langue dans la recherche d'information (section 3.5) et les interfaces de requêtes en langage naturel pour l'interrogation des bases de données (section 3.6). Enfin, en guise de transition vers notre contribution, nous présentons une vision personnelle des tenants et des aboutissants des recherches naissantes concernant les interfaces de requêtes pour la recherche d'information semi-structurée (section 3.7).

Traitement automatique de la langue pour la recherche d'information semi-structurée

La seconde partie du mémoire présente nos travaux. Le chapitre 4 détaille tout d'abord une proposition concernant la définition et l'application de la notion de *contexte de lecture*. Ce concept apporte une solution aux problèmes posés par la flexibilité du langage XML pour la conservation du texte original lors de l'analyse automatique du contenu des documents.

Dans les derniers chapitres, nous nous intéressons à la production automatique de requêtes formelles à partir d'un énoncé en langage naturel. Le chapitre 5 décrit la méthode que nous utilisons pour analyser la requête et obtenir une représentation formelle susceptible d'être traduite dans n'importe quel langage de requêtes dédié à la recherche d'information. La section 5.1 pose le problème, énumère les contraintes et les libertés, justifie les choix. Puis les phases principales du processus sont développées : les analyses morphosyntaxique et syntaxique d'une part (section 5.2) et, d'autre part, l'application de règles sémantiques, déduites du contexte ou de la structure des documents, permettant de construire les relations appropriées entre les éléments structurels et le contenu textuel des documents (section 5.3). Enfin le mécanisme d'obtention de la requête finale est expliqué (section 5.4).

Le chapitre 6 se situe dans la continuité du précédent, mais se place à un niveau d'analyse très différent. Il relate nos études concernant les possibilités de mise à profit des formulations employées par l'utilisateur dans sa demande, dans le but de construire des requêtes formelles plus fines et plus détaillées, et par conséquent plus efficaces. Ces réflexions et leur application concernent le contexte des éléments recherchés (section 6.1), leur taille (section 6.3) et les liens qu'ils entretiennent explicitement, par des références, avec des autres documents (section 6.2). Comme nous l'avons déjà dit, ceci nous a conduit à la conception d'un langage de requêtes plus adapté à nos besoins et à l'adaptation d'un moteur de recherche existant pour traiter ce nouveau langage

(section 6.4). Les modifications apportées à notre interface pour générer ces nouvelles requêtes sont exposées à la section 6.5.

Enfin, le dernier chapitre a trait aux expérimentations menées et aux résultats obtenus par nos différents systèmes de traitement automatique des requêtes en langage naturel. Les procédures utilisées sont différentes selon la démarche choisie. La construction des requêtes dans un langage existant au préalable permet de comparer les résultats produits avec les performances obtenues lors de l'utilisation de requêtes écrites manuellement. On n'évalue alors que les requêtes, indépendamment du moteur de recherche. En revanche, l'utilisation d'un nouveau langage et de son système de traitement dédié empêche cette simplification. Il est alors nécessaire de confronter notre système de façon globale à d'autres moteurs, pour comparer légitimement leurs résultats. Ces diverses expérimentations ont été rendues possibles par le cadre d'évaluation proposé par la campagne INEX, permettant d'obtenir des collections de documents, des requêtes en langage naturel et en langage formel et des jugements de pertinence des documents par rapport à ces requêtes.

Première partie

Recherche d'information, structure et langage naturel

Chapitre 1

Les documents semi-structurés

1.1 Le langage XML

XML (eXtensible Markup Language [251]) est un standard mis en place par le *World Wide Web Consortium* (W3C [244]). *"Il définit une syntaxe générique utilisée pour marquer les données avec un balisage simple et lisible par les humains."* [1] [102]. *XML*

La structure est représentée en XML par des *éléments*, contenant des *attributs*, du texte ou d'autres éléments. Les éléments ne peuvent pas se chevaucher. Le choix du nom des éléments structurants et des attributs, ainsi que l'organisation des éléments entre eux, est laissé à la libre volonté de l'auteur. C'est pourquoi on dit que le langage XML est *générique*.

Un exemple de document XML, représentant des informations sur des films présents dans une cinémathèque, est donné à la figure 1.1. L'organisation particulière d'un document XML (notamment l'imbrication des éléments sans possibilité de chevauchement) permet de représenter celui-ci sous forme arborescente, comme le montre l'exemple de la figure 1.2.

Nous pouvons illustrer par cet exemple les bases de la terminologie XML :

- "cinémathèque", "film", "titre", etc. sont des *noms* ou des *types de balises* ;
- `<film>` et `</film>` sont des *balises* (respectivement balises de début et de fin d'élément) ; *balise*
- Les balises de début et de fin ainsi que leur contenu (texte et éléments insérés entre les balises) constituent un *élément*, aussi appelé *nœud* ou *sous-arbre* ; *élément*
- `id="1"` est un attribut (id) ayant la valeur "1". *attribut*
- Les parties purement textuelles (*"Woody Allen"*, *"1993"*, etc.) sont des *éléments textuels*.
- L'élément `acteurs` est le *parent* des éléments `acteur`, qui sont donc ses *enfants*. On dit également qu'un parent *contient* un enfant. Par extension, l'élément `film` est l'*ancêtre* des éléments `acteur` (et `titre`, etc.) qui sont ses *descendants*. *hiérarchie*
- L'élément `cinémathèque` est l'élément *racine*, il est l'ancêtre de tous les autres éléments. Par la même analogie avec les arbres, on dit que les éléments situés au plus bas de l'arborescence (`titre`, `acteur`, etc.) sont des *feuilles*. *racine* *feuilles*

Par ailleurs, les liens hiérarchiques (parenté, attributs nécessaires, etc.) sont décrits par une *DTD* (Document Type Definition) ou par un *Schema* [249]. La DTD ou le Schema définissent un "sous-langage" restreignant, imposant avec plus ou moins de contraintes d'utiliser une organisation prédéfinie. Les Schemas permettent de décrire beaucoup plus de caractéristiques que les DTD (notamment concernant les types

[1] "It defines a generic syntax used to mark up data with simple, human-readable tags." [102, p.3]

```
<cinémathèque>
    <film id="1">
        <titre>Meurtre Mystérieux à Manhattan</titre>
        <réalisateur>Woody Allen</réalisateur>
        <année>1993</année>
        <durée unité="min">108</durée>
        <affiche>./affiches/allen_mmm.jpg</affiche>
        <site>http ://www.woodyallen.com/</site>
        <acteurs>
            <acteur>Woody Allen</acteur>
            <acteur>Diane Keaton</acteur>
            <acteur>Anjelica Huston</acteur>
        </acteurs>
    </film>
    <film id="2">
        ...
    </film>
    ...
</cinémathèque>
```

FIG. 1.1 – Exemple de document XML représenté sous forme textuelle.

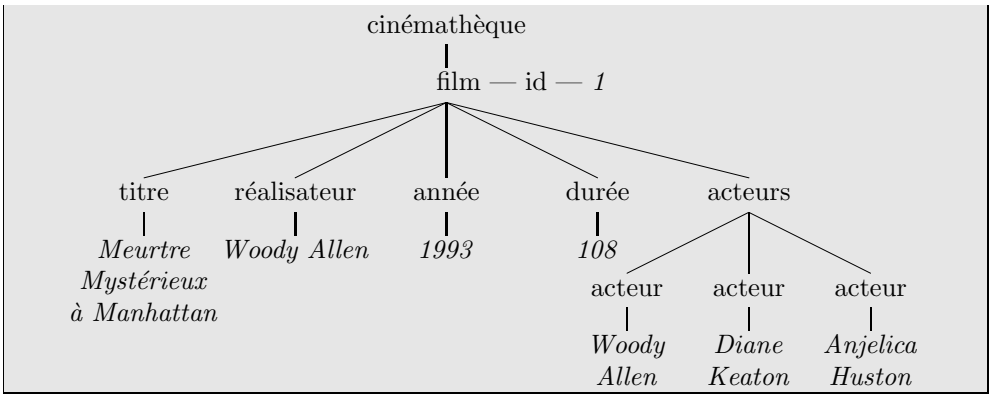


FIG. 1.2 – Exemple de document XML représenté sous forme arborescente.

de données utilisées dans le contenu). La DTD est abordée de façon plus précise en annexe A.2.

Ainsi, les formats DocBook [255], la TEI [112] ou XHTML [250], mais aussi des langages de traitement du XML comme XSLT [246] ou XQueryX [252] obéissent à des DTD et/ou des Schemas spécifiques, ce sont des sous-langages de XML.

On voit dans les exemples que si la syntaxe XML est relativement intuitive, en revanche la représentation textuelle d'un document peut vite devenir très verbeuse, et donc assez difficile à lire pour un humain. Des langages de transformation permettant d'obtenir une apparence plus agréable sont donc utilisés (XSLT, CSS).

Il est également possible de naviguer et de sélectionner des informations à travers les documents XML grâce aux langages XPath [245], XSLT [246] ou XQuery [253].

Plus de détails sur XML peuvent être trouvés à l'annexe A.

1.2 Du document plat au document structuré

Structure *: Manière dont les parties d'un ensemble concret ou abstrait sont arrangées entre elles.*
Dictionnaire Larousse.

marquage, balisage

Coombs et al. [51] distinguent six sortes de marquage, ou balisage :

- La *ponctuation* : virgules, points de toutes sortes, tirets divers. Cette forme de marquage est familière pour tous, au moins dans le monde occidental, relativement stable, mais certaines de ses caractéristiques sont variables selon les langues (espaces autour des signes notamment), les usages ou les habitudes.
- La *présentation* : les auteurs utilisent différents moyens de rendre leurs documents plus clairs. Les passages à la ligne, espacements divers, coupures de pages, énumérations, notes, numérotation, ou même les étiquettes explicites du type "Chapitre 1." entrent dans cette catégorie.
- Le *marquage procédural* : il s'agit d'instructions remplaçant le marquage de présentation, dans le cas où le document est destiné à être formaté par un système. Exemple : `'.sk 3'` pour sauter trois lignes (*"skip 3 lines"*), ou `'.in -10'` pour une indentation négative de dix colonnes.
- Le *marquage descriptif* permet à l'auteur d'identifier le rôle (la sémantique) d'une partie de texte (paragraphe, citation, note). Dans la plupart des documents lus par un être humain, ce marquage est remplacé par des conventions de présentation.
- Le *marquage référentiel* représente des entités externes au document. Une référence à une telle entité sera remplacée par le contenu de celle-ci pendant le formatage du texte.
- Le *méta-marquage* facilite l'interprétation des autres types de marquage, notamment par une description des procédures qu'ils nécessitent.

La ponctuation, ainsi que la présentation dans une moindre mesure, sont des marquages effectués par l'auteur de tout document écrit, numérique ou non. Le lecteur humain en a besoin pour apporter du sens au texte qu'il lit. Dans la suite de cette thèse, on appellera *document plat* tout texte ne comportant que ces deux types de marquage. On dira d'un tel document qu'il est *non structuré*, même si la ponctuation ou les passages à la ligne sont des éléments structurants (séparation en phrases, en paragraphes, etc.).

document plat

On considèrera qu'il existe véritablement une structure lorsque celle-ci est indiquée par des indices conventionnels formels et dénués d'ambiguïté[1].

document structuré

A l'opposé des documents plats, les documents dits *structurés* possèdent une structure régulière prépondérante à base de marquage descriptif. Par exemple, la structure d'une base de données relationnelle est représentée par des tables comportant plusieurs colonnes et plusieurs lignes, ainsi que par des relations entre les champs des différentes tables. L'ordre des éléments n'a généralement pas d'importance. On ne parle alors plus de *texte* mais de *données* ; ces données n'ont habituellement aucune signification intrinsèque, c'est-à-dire qu'il est impossible de les considérer sans examiner la structure dans laquelle elles sont inscrites.

document semi-structuré

Le document *semi-structuré* est un pont entre les données structurées et non structurées [228]. Le langage XML, qui permet de produire des documents à la fois structurés et semi-structurés, est devenu un mode de représentation standard dans le domaine des

[1] On peut remarquer notamment que le point peut signifier une fin de phrase, mais également apparaître dans un sigle ou le nom abrégé d'une personne. La décomposition automatique d'un texte en phrases n'est, de ce fait, pas un problème trivial. Par cette ambiguïté, le point n'est pas considéré comme structurant au sens où nous l'entendons ici.

documents électroniques. D'une part, par un marquage inséré dans le texte, il apporte des éléments sémantiques supplémentaires concernant la structure, le fond ou la forme du document. D'autre part, par le caractère flexible, irrégulier ou incomplet de sa structure [2], ainsi que par son format (textuel), il permet de stocker et d'échanger des données beaucoup plus aisément que les bases de données.

Ainsi, tous les exemples donnés dans ce chapitre témoignent de cette flexibilité. Il suffit par exemple d'écrire une ligne de texte pour ajouter un acteur à un film (figure 1.1) ou un mot-clé à une référence bibliographique (figure 1.4). Si la DTD le permet, il est possible de supprimer la ligne concernant l'éditeur dans la figure 1.3 si cette information est inconnue.

La notion de documents semi-structurés ne recouvre pas une seule réalité. *"Le terme "document" en XML constitue [...] un terme technique, qui ne correspond pas nécessairement à la notion classique d'un document "narratif", c'est-à-dire à un ensemble de données textuelles organisées et mises en forme à l'attention d'un lecteur. Il s'applique également à toute structure de données à vocation d'échange interapplications."* [41]

Comme l'indiquent notamment *Fuhr et Großjohann* [77], il est possible de distinguer deux approches (ou vues) différentes du XML, correspondant à deux besoins différents, et à deux manières d'aborder la recherche d'information :

orienté document

- *l'approche orientée document* considère le document sous sa forme traditionnelle, c'est-à-dire un texte dont la finalité principale reste la lecture par un être humain. Dans cette vue le marquage (ou balisage) sert essentiellement à fournir des informations concernant la structure (chapitres, sections...) et/ou la forme (caractères gras, italiques...) du texte. Des exemples de cette approche sont les manuels, les livres (voir figure 1.3), les articles ou les pages Web statiques.

orienté données

- *l'approche orientée données*, plus proche des bases de données, est utilisée pour représenter et échanger des ensembles structurés de données, comme des horaires de vols, des catalogues, des bibliographies, etc. Cette vue est illustrée par un exemple à la figure 1.4.

En pratique il n'existe pas de frontière nette entre ces deux approches. Ainsi, les premiers éléments du document de la figure 1.3, qui informent sur le titre, l'auteur, l'éditeur (et pourquoi pas d'autres métadonnées*), peuvent être considérés comme étant orientés données, puisque le contenu sans la structure devient beaucoup moins informatif.

1.3 La sémantique du balisage

1.3.1 Généralités

L'ajout d'une structure de type XML aux documents permet de prendre certaines libertés avec le texte lui-même. Notamment, l'ordre *physique* des éléments du fichier XML peut différer de l'ordre réel, *logique* [226], du texte lu par l'humain après mise en forme.

C'est la sémantique particulière de certaines balises, attribuée par l'être humain qui a construit (ou structuré) le document, qui autorise ces manipulations tout en conservant le sens du texte initial.

A l'opposé, pour les processeurs automatiques de documents XML, les balises sont toutes équivalentes, et surtout toutes totalement vides de sens. Nous reviendrons plus en détails sur ce problème au chapitre 4.

La perte d'information entre la création d'un document XML et son analyse est donc manifeste. Pour cette raison, des initiatives telles que l'élaboration des Schemas

```
<livre type="roman">
    <titre>Le tour du monde en 80 jours</titre>
    <auteur>Jules Verne</auteur>
    <éditeur>Hatzel</éditeur>
    <chapitre n="I">
        <titre_chapitre>
            Dans lequel Phileas Fogg et Passepartout s'acceptent réciproquement,
            l'un comme maître, l'autre comme domestique
        </titre_chapitre>
        En l'année 1872, la maison portant le numéro 7 de Saville-row, Burlington
        Gardens – maison dans laquelle Sheridan mourut en 1814 – était habitée
        par Phileas Fogg, esq., l'un des membres les plus singuliers et les plus
        remarqués du Reform-Club de Londres, bien qu'il semblât prendre à tâche
        de ne rien faire qui pût attirer l'attention.
        ...
    </chapitre>
</livre>
```

FIG. 1.3 – Représentation en XML d'un roman. Ici le texte reste l'information essentielle du document, la structure est secondaire.

XML [249] et le Web Sémantique [247], dirigées par le World Wide Web Consortium (W3C [244]), visent à représenter formellement la sémantique structurelle des documents.

D'autre part, d'autres travaux portent sur une simple catégorisation des balises pour apporter un surplus de sémantique aux documents [220, 186]. En 1981, bien avant l'invention du langage XML, *Goldfarb* [91] faisait déjà la distinction entre les approches "procédurale" (instructions de formatage) et "descriptive" (composants logiques, comme les paragraphes, les chapitres ou les titres). Cette distinction est devenue consensuelle avec SGML [92] puis XML.

En HTML, qui concerne essentiellement la forme du document, la dichotomie entre balises "logiques" et balises "physiques" sépare le rôle sémantique joué par un élément d'une stricte description de son apparence.

Par ailleurs, la TEI (Text Encoding Initiative [112]) a mis en place une intéressante répartition entre différents niveaux d'inclusion : "chunk", "phrase-level" et "inter-level". La définition de ces niveaux est purement syntaxique et aucune distinction sémantique n'est faite.

D'autres réflexions ont été menées plus récemment [185, 176, 186, 218], introduisant de nombreux nouveaux concepts, parmi lesquels nous pouvons citer une discrimination entre deux axes (comprenant d'une part les domaines logique et renditionnel, et d'autre part impératif ou indicatif [185]) ainsi que les marquages proleptiques et métaleptiques [176].

Nous allons nous attarder sur une division des balises particulièrement appropriée au domaine de la recherche d'information, division que nous utiliserons par la suite. Elle a été proposée par *Lini et al.* [143], dans le but d'identifier différentes catégories qu'il serait important de distinguer dans le cadre de la recherche dans des documents XML. L'idée de départ était de permettre des traitements différents pendant une recherche de motif (de séquence de caractères) [143, 49].

```
@InProceedings{Boudon99,
    title = {Le 18 Brumaire dans l'histoire},
    author = {Jacques-Olivier Boudon and Yves Bruley},
    booktitle = {Actes du colloque "Brumaire. La prise de pouvoir de
Bonaparte"},
    year = {1999},
    address = {Paris, France},
    pages = {161-173},
    keywords = {Napoléon Bonaparte, Coup d'Etat du 18 Brumaire}
    }
```

a. Représentation en BibTeX [138]

```
<bibitem type="InProceedings" key="Boudon99">
    <title>Le 18 Brumaire dans l'histoire</title>
    <author>Jacques-Olivier Boudon and Yves Bruley</author>
    <booktitle>Actes du colloque "Brumaire. La prise de pouvoir de
    Bonaparte"</booktitle>
    <year>1999</year>
    <address>Paris, France</address>
    <pages>161-173</pages>
    <keywords>
        <item>Napoléon Bonaparte</item>
        <item>Coup d'Etat du 18 Brumaire</item>
    </keywords>
</bibitem>
```

b. Représentation en XML

FIG. 1.4 – Représentation d'un élément bibliographique sous un format BibTeX (a.) ou XML (b.). On parle dans ce cas-là de document semi-structuré, car la structure n'est pas régulière ; des champs peuvent être omis s'ils sont inconnus ou non pertinents (adresse, pages...), d'autres ont une certaine structure interne (par exemple, dans le champ `author`, les noms obéissent à une certaine grammaire et sont séparés par le mot réservé `'and'`). BibTeX permet aussi d'effectuer des références croisées. Les mots-clés (`keywords`) de la version XML contiennent un nombre de champs `item` indéterminé. Toutes ces caractéristiques rendraient inutilement complexe la description de ces informations par des bases de données [2]. Cet élément bibliographique est néanmoins *orienté données*, puisque la structure a autant d'importance que le contenu textuel.

1.3.2 Balises dures, balises de saut, balises transparentes

Les trois différentes classes proposées par *Lini et al.* [143] sont les suivantes :

hard tags

– les *balises "dures"* (*"hard" tags*) sont les plus fréquentes. Elles interrompent la "linéarité" d'un texte et contribuent généralement à la structuration du document. Des exemples de balises dures sont les titres, les chapitres, les paragraphes.

(1.1)
```
...
<titre>Text Retrieval System</titre>
<auteur>Michele Paoli</auteur>
...
```

– Les *balises "transparentes"* (*"soft" tags*) identifient des parties significatives du texte, comme les textes cités, les effets de forme ou les corrections, mais sont "transparentes" lorsqu'on lit le texte. *soft tags*

(1.2) Les ministres de l'<gras>Union Européenne</gras> se sont réunis ...

– Les *balises de "saut"* (*"jump" tags*) sont utilisées pour représenter des éléments particuliers comme les notes de marges, les références bibliographiques ou des définitions. Elles sont détachées du texte les entourant, comme la `'note'` suivante : *jump tags*

(1.3) La mort de Mozart, dans l'indifférence générale<note>Le corps du compositeur sera jeté à la fosse commune, sans même une croix.</note>, l'empêcha de terminer son requiem.

N.B. : Des éléments comme les tableaux ou les listes, qui ne sont pas abordés par les auteurs, semblent particuliers. Les tableaux peuvent être considérés en première approximation comme des éléments "durs", mais leurs spécificités demandent une attention particulière [65]. Quant aux listes, qui possèdent également des caractéristiques propres [7], nous verrons qu'il est possible de les traiter de plusieurs façons différentes.

Par ailleurs, les formules mathématiques, chimiques ou autres peuvent être utilisées dans le texte (éléments transparents) ou dans un contexte à part (éléments durs).

Si ces différentes catégories de balises semblent très pertinentes dans le cadre de la recherche d'information, elles souffrent cependant d'un manque de définitions précises. Nous reviendrons sur ces définitions dans le chapitre 4 au travers d'une description de cadres théorique et pratique utilisant les concepts décrits dans cette section.

Chapitre 2

Recherche d'information et structure

La recherche d'information automatisée est une discipline qui met en jeu le stockage et la représentation de l'information d'une part, l'analyse et la satisfaction d'un besoin d'information d'autre part.

Plusieurs disciplines évoluent dans cette problématique, elles se chevauchent souvent, mais restent distinctes à de nombreux points de vue. C'est pourquoi la première section de ce chapitre est consacrée à décrire ce qu'est la recherche d'information (RI), et ce qu'elle n'est pas. Nous analyserons également les raisons pour lesquelles une discipline relativement récente, la RI dans les documents semi-structurés de type XML, introduit une problématique spécifique (section 2.2). Les concepts de base étant définis, la suite du chapitre sera consacrée à décrire le processus de recherche dans les grandes lignes. Nous passerons en revue les étapes d'indexation (section 2.3) et d'interrogation (section 2.4), en insistant pour chacune d'elles sur les caractéristiques propres aux documents semi-structurés. Enfin nous nous attarderons tout particulièrement sur la question de l'évaluation des systèmes de RI (section 2.5) en présentant notamment la campagne INEX qui s'est spécialisée dans la recherche XML (section 2.5.4).

2.1 Introduction

> Information (INFORM) *: Elément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué.*
> Dictionnaire Larousse.

La recherche d'information (RI), dans le cas général, comporte deux acteurs principaux :

- Un être humain, qui a un besoin d'*information* à un moment donné[1] ;
- Un stock de *données*, fixé au préalable.

données vs. *info*

"Les données sont reçues, stockées et retrouvées par un endosystème. Les données sont impersonnelles ; elles sont disponibles pour tout utilisateur du système. L'information, en revanche, est un ensemble de données qui correspond à un besoin particulier. Le concept d'information a des composantes personnelles et temporelles absentes du concept de donnée." [2] [133]

[1] Ou qui est dans un *"état anormal de connaissance"* ("Anomalous state of knowledge") [24]

[2] "*Data* are received, stored, and retrieved by an information endosystem. The data are impersonal ; they are equally available to any users of the system. *Information*, in contrast, is a set of data that have been matched to a particular information need. That is, the concept of information has both personal

Les besoins d'information des utilisateurs sont divers, et cette diversité influe sur le mode de recherche de cette information. Il est possible de distinguer quatre types génériques de besoins d'information, et donc de stratégies de recherche [159, chap. 13] :

- *La recherche d'un élément connu* : l'utilisateur sait exactement quels éléments il recherche. Il sait reconnaître les éléments désirés s'il les voit. Par exemple, il recherche une citation bibliographique précise et possède assez d'information pour l'identifier sans ambiguïté.
- *La recherche d'une information spécifique* : l'utilisateur recherche une information spécifique mais ignore sous quelle forme elle se présente. Cette information peut d'ailleurs être présente à plusieurs reprises. Exemple : A quelle date le président Kennedy a-t-il été assassiné ? La réponse sera trouvée ou non, mais ne peut être partielle.
- *La recherche d'une information générale* : l'utilisateur recherche une information sur un sujet en général. Il existe de nombreuses façons de décrire le sujet. Il est possible que l'information pertinente ne soit pas reconnue, et cette information peut ne satisfaire l'utilisateur que de façon partielle [23].
- *L'exploration* : le but n'est pas de répondre à une question en particulier, mais de parcourir l'ensemble des données pour découvrir quels types d'informations concernant un sujet ou un domaine sont présents.

Ces différentes stratégies de recherche, en particulier les trois premières, sont le sujet d'étude de domaines scientifiques distincts : la recherche d'un élément connu a nécessité la mise en place de langages de requête structurés tels que SQL [72] pour les bases de données ou XQuery [253] pour XML. La recherche d'une information spécifique est plutôt du domaine des systèmes de question-réponse [241]. Enfin le troisième type de besoin est l'apanage de la recherche d'information.

RI

La RI est donc le processus consistant à chercher une réponse appropriée au besoin de l'être humain dans la masse de données disponible. Pour satisfaire ce besoin, il ne suffit pas d'apporter à l'utilisateur un document traitant du sujet demandé ; il faut également que le type du document (en termes de longueur, de complexité, de niveau de langage, de portées géographique et temporelle [96]) soit approprié.

Nous allons bien sûr nous intéresser exclusivement à la recherche d'information assistée par ordinateur [199, 18], ce qui ajoute un acteur : le système de recherche d'information (SRI). Celui-ci fait l'interface entre les deux acteurs précédemment cités.

requête

L'utilisateur exprime son besoin à la machine au moyen d'une requête, et le système a pour mission de retrouver dans la masse d'information disponible les documents répondant à la requête. Etant donnée la quantité importante de données potentiellement retrouvée, les systèmes modernes effectuent un classement des documents de manière à placer en tête de liste ceux qui sont jugés les plus pertinents. La figure 2.1 illustre ce processus de façon très générale. Une phase préalable d'analyse des documents, indépendante du besoin, est nécessaire. Elle correspond en pratique au processus d'*indexation*, elle utilise un *modèle de documents* (cf. section 2.3). La formalisation de la requête, ainsi que la recherche et le classement des documents (à l'aide d'une *fonction d'appariement*) représentent la base de la phase d'*interrogation* (cf. section 2.4). Le résultat est dans la plupart des cas un ensemble ordonné de références à des documents de la collection (celui jugé le plus pertinent étant bien sûr placé en première position). La description théorique de toutes ces étapes est appelée *modèle de recherche d'information*.

modèle de RI

De nombreuses extensions, comme le retour (ou rétroaction) de pertinence (*relevance feedback*), la reformulation de la requête, l'évaluation, peuvent venir enrichir ce schéma universel [199, 22, 100, 177, 204].

and time-dependant components that are not present in the concept of data." [133, page 8]

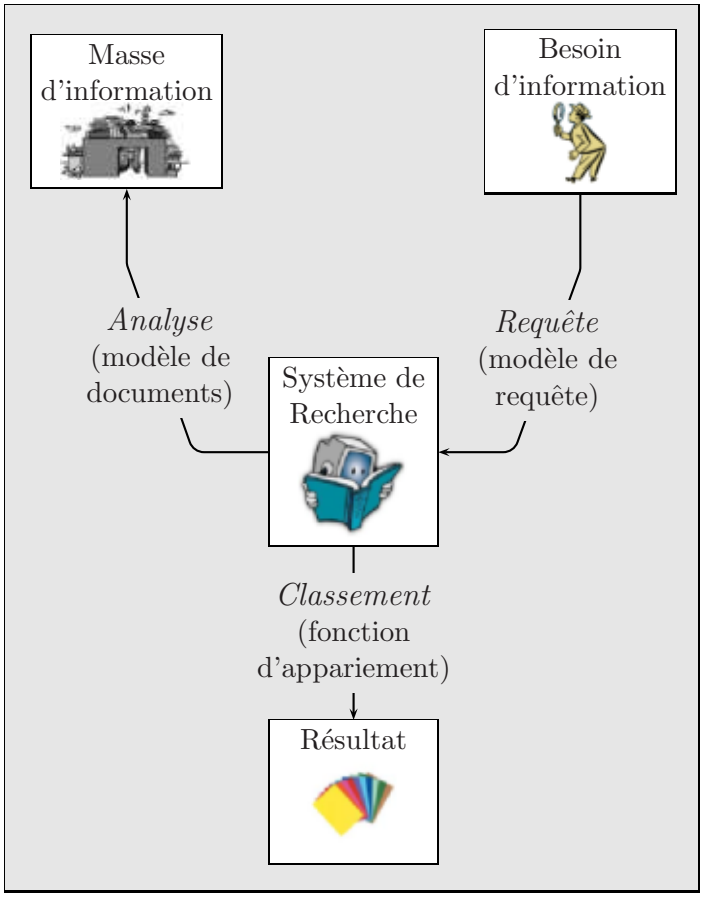


FIG. 2.1 – Principe général de la recherche d'information.

Il est important de noter d'ors et déjà que le besoin d'information de l'utilisateur est parfois vague et toujours subjectif. Deux personnes décriront un même besoin de deux façons différentes, et une même personne le décrira différemment selon son humeur. De plus une formulation peut être comprise de plusieurs façons par des personnes différentes. La prise en compte de ces aspects psychologiques conduit aux constats suivants :

- La perte d'information entre la réalité du besoin d'information et son expression peut être importante.
- La *pertinence* d'un document pour une requête est une notion variable et très complexe à définir.
- Il ne peut pas exister de système de recherche d'information parfait.
- L'évaluation d'un système dépasse les aspects habituels de performance informatique, nécessite la mise en place de métriques particulières et l'intervention de juges humains, eux-mêmes subjectifs et soumis à des variations entre les individus d'une part, et pour un même individu d'autre part.

Nous reviendrons sur ces différents points tout au long de ce chapitre.

Comme nous l'avons dit, chaque type de besoins d'information se concrétise par une discipline de recherche spécifique. Ainsi, ce que nous appelons ici "recherche d'information" se distingue d'autres domaines proches mais mettant en œuvre des techniques sensiblement différentes. Notamment :

- La recherche de données qui consiste, dans le cadre de la recherche dans un texte,

à déterminer quels documents contiennent les mots-clés demandés par l'utilisateur [18]. Dans le domaine des bases de données, elle permet de retrouver des données dans un espace structuré. Dans tous les cas l'évaluation de la réponse est binaire (la donnée correspond ou ne correspond pas à ce qui est cherché).

extraction d'information

– L'extraction d'information [57] qui isole des fragments d'information, les classe et les organise dans un format fixe. Elle peut servir au remplissage automatique de bases de données ou à un processus d'indexation, préalable à la recherche d'information.

question/réponse

– Les systèmes question/réponse (*Question Answering*) [86, 241] qui donnent une réponse aussi concise que possible à une question précise posée en langage naturel*. Par exemple, la réponse à la question *"Qui était président de la France en 1985 ?"* doit être *"François Mitterrand"*, et non des documents traitant ce sujet.

résumé

– Le résumé automatique [162] qui a pour but de prendre un ou plusieurs documents pré-sélectionnés en entrée et d'en composer un résumé.

2.2 Les spécificités de la recherche d'information semi-structurée

La recherche de l'information dans des corpus semi-structurés (écrits en XML dans notre cas) avec une approche orientée document* comporte des caractéristiques spécifiques qui la distinguent fortement de la recherche d'information classique. Par certains aspects, notamment la nécessité d'utiliser un langage de requêtes structuré, elle se rapproche des interrogations de bases de données. Pourtant les enjeux restent bel et bien ceux de la recherche d'information.

2.2.1 L'unité d'information pertinente

unité d'information

Dans la discipline de la recherche d'information, se pose le problème de l'*unité d'information* adéquate. En réponse à une requête, la majorité des systèmes retourne un ensemble de *documents*. Outre qu'il est parfois difficile d'établir ce qu'est un document numérique [182], le choix de cette unité n'est pas une garantie de la satisfaction de l'utilisateur. Il peut en effet sembler plus pertinent, dans certains cas, de ne retourner qu'une portion du document, voire un ensemble de textes issus de documents différents, pour éviter que l'information utile ne soit trop dispersée [165]. De plus, la division des documents en plus petites unités de sens peut constituer une aide au processus de recherche [104]. Ces considérations ne datent pas de l'apparition des langages semi-structurés, et le sujet préoccupe depuis longtemps les tenants de la *recherche de passages* (*passage retrieval* [196]).

doxel

Dans le cadre de la recherche d'information dans des documents XML, une nouvelle unité d'information potentielle se présente : l'élément XML, aussi appelé *doxel* [178] (pour *"document element"* en Anglais, et par analogie avec le pixel, l'unité graphique élémentaire), ou *nœud* (sous-arbre) dans la représentation arborescente du document. Il est également possible de s'intéresser à des ensembles de doxels, consécutifs ou pas dans les documents initiaux, ceux-ci composant eux-mêmes un document (semi-)structuré [177]. Les autres unités, comme le document entier, la phrase, le paragraphe ou des unités sémantiques particulières, sont toujours à prendre en considération.

Par ailleurs, la pertinence du choix d'une certaine unité d'information en réponse à une requête peut se mesurer selon deux dimensions [43, 136]. La première correspond à la notion de présence de l'information demandée dans l'information apportée ; la seconde détermine si l'unité d'information retournée a une taille appropriée par rapport

à la quantité et à la disposition des informations pertinentes qu'elle contient. Ces deux dimensions sont reprises dans les mesures de *degré de pertinence* et de *couverture*, puis d'*exhaustivité* et de *spécificité* mises en place pour l'évaluation de la recherche d'information dans les documents XML (cf. section 2.5).

auto-explicativité

On considère généralement que la partie de document renvoyée par un système se doit d'être *auto-explicative*, c'est-à-dire de ne pas dépendre du reste du document pour être comprise. Pourtant cette règle dictée par une vision de laboratoire de la recherche d'information [130] nous semble très discutable. Tout d'abord parce qu'aucun document complet n'est indépendant du reste de la connaissance (voir les quelque 250 références externes que comporte ce mémoire). Ensuite parce qu'un document est généralement écrit pour être lu en entier ; il est conçu comme un tout, l'humain et le langage étant ainsi faits qu'il semble difficilement concevable qu'un paragraphe bien écrit ne fasse en aucune manière référence au reste (introduction préalable du contexte, recours à des références pronominales, à des paraphrases, etc.). Une application stricte du principe d'auto-explicativité conduirait à une perte bien trop importante de *spécificité*. Par ailleurs l'exposition des résultats qui semble satisfaire le plus l'utilisateur est celle consistant à mettre en évidence l'élément retourné tout en le présentant dans son contexte (le document entier), avec notamment un accès à la table des matières de ce document [83, 139].

2.2.2 Recherche sur contenu et structure

L'utilisateur cherchant une information dans un corpus qu'il sait structuré peut formuler ses requêtes de trois façons différentes :

contenu seul

– Il peut choisir d'ignorer la structure et de ne formuler sa requête qu'en fonction du *contenu*, parce qu'il n'a pas d'idée précise de l'endroit où se trouve la réponse à sa question, ou parce que son besoin d'information est trop vague, ou encore parce qu'il ne connaît pas précisément la structure des documents.

contenu/ structure

– Il peut "aider" le moteur de recherche en précisant des contraintes structurelles dans sa requête.
– Il peut également ne se préoccuper que de la structure. Cet aspect correspond à une interrogation de type "base de données" (listings par exemple) dans laquelle la recherche d'information n'a pas sa place, et nous ne l'aborderons pas.

Le format habituel d'une requête, dans le cadre de la recherche d'information dans des documents plats*, est une simple liste de mots-clés, ordonnée ou non, avec éventuellement des opérateurs (booléens ou autres). Le besoin d'exprimer des contraintes structurelles impose de mettre en place des langages de requêtes possédant un formalisme plus complexe. Les possibilités des systèmes de recherche documentaire "classique" restent indispensables, mais des fonctionnalités propres aux manipulations de bases de données doivent être ajoutées. Ainsi *Sacks-Davis et al.* [192] distinguent huit classes de requêtes, de la recherche par mots-clés uniquement jusqu'aux accès de type bases de données, en passant par les requêtes dans des sous-parties de documents ou dans différents types de documents. *Bonifati et Ceri* [28] comparent des modèles de RI en utilisant dix groupes de caractéristiques, parmi lesquelles le modèle de données, les possibilités de sélection, de filtrage, de jointures, de restructuration, d'opérations sur les ensembles, de mise à jour des données, mais aussi l'intégration avec les spécificités de XML. *Baeza-Yates et Navarro* [17] ajoutent la possibilité de chevauchement structurel dans les réponses et la notion d'*expressivité*, et *Sauvagnat* [205, 204] celle d'*informativité*.

Un compromis est souvent nécessaire entre les fonctionnalités avancées, la facilité d'utilisation du langage et la complexité informatique des algorithmes mis en place.

2.2.3 Interprétation de la requête

Si l'utilisation d'un langage de requêtes structuré semble rapprocher la recherche d'information semi-structurée de l'interrogation des bases de données, il est important de souligner que les enjeux restent très différents.

Dans le document XML tel que nous le considérons, le contenu textuel est l'information prépondérante. La recherche d'information est un instrument appliqué à des documents dont la fonction première est d'être tout simplement lus par des humains (après une mise en forme appropriée). A l'opposé, une base de données n'existe que pour être interrogée par un utilisateur ou un logiciel (en SQL généralement). Cet utilisateur connaît dans la plupart des cas la structure de la base qu'il interroge et il doit impérativement fournir des indications structurelles dans sa requête. De plus son besoin d'information est précis, il recherche des données répondant à des contraintes de contenu et de structure *strictes*.

L'interprétation de la requête SQL est donc stricte. En conséquence, une requête correcte conduira inévitablement à la réponse attendue, et tout autre requête produira des résultats inutiles. Si la notion de requête "meilleure" qu'une autre existe, c'est en termes de *performances* (rapidité de traitement par tel ou tel système de gestion), mais en aucun cas au niveau de la *qualité* des enregistrements retournés.

En revanche, en recherche d'information (structurée comme traditionnelle), l'utilisateur n'a généralement pas une idée précise du type d'information qui le satisferait le plus (taille, forme et même contenu). Sa requête n'est donc plus le reflet fidèle du résultat souhaité ; elle est plutôt considérée comme une "aide" apportée au système pour lui permettre de répondre efficacement au besoin d'information. C'est pourquoi il n'existe pas de réponse "parfaite" à une requête, et il est toujours possible d'améliorer sa requête pour obtenir des résultats plus proches de son attente. De plus, des requêtes très différentes peuvent conduire à des résultats de pertinence comparable.

Ainsi un résultat qualitativement satisfaisant ne dépend, dans le cadre des bases de données, que de la composition de la requête ; en RI, en revanche, la clé d'un bon résultat est partagée entre la requête et le système qui la traite. Le moteur de recherche a donc la "liberté" d'interpréter la requête de façon beaucoup plus souple. Ce principe s'applique bien sûr au contenu textuel[1] mais aussi à la structure. Une contrainte structurelle peut n'être considérée que comme une indication non stricte. On peut par exemple imaginer qu'un utilisateur qui demande un paragraphe expliquant un concept donné puisse être satisfait par une figure décrivant ce concept. En revanche un long article sera moins pertinent, puisque la requête concernait un élément relativement court.

Ces subtilités spécifiques à la recherche d'information semi-structurée sont très importantes et justifient en grande partie le travail présenté par la suite.

2.2.4 Bilan

La tableau 2.1 récapitule les enjeux de la recherche d'information dans les documents semi-structurés. Il ne s'agit ici que des aspects concernant le principe même de la recherche, et non des techniques à mettre en œuvre pour gérer les problèmes engendrés par la présence d'une structure flexible (indexation, évaluation, etc.). Par ailleurs nous n'abordons que les documents dont le contenu est textuel. Pourtant, nous avons vu (section 1.1) que les documents XML pouvaient intégrer des références à des éléments multimédia, voire ces éléments eux-mêmes, qui peuvent bien entendu être pris en compte

[1] Les systèmes peuvent retourner des documents ne contenant pas tous les termes de la requête, mais aussi utiliser des techniques d'extension de requête, de racinisation, etc. (cf. section 3.5).

	Documents plats	*Documents semi-structurés*	*Documents structurés*
Contenu	texte seulement	texte + structure	structure + données
Besoin d'information	**général**		précis
Unité de recherche	document/segment	**élément (doxel)**	(tuple)
Demande	texte seulement	contenu et/ou structure	
Requête	mots-clés	**langages de requêtes structurés**	
Interprétation	**vague**		stricte

TAB. 2.1 – Récapitulatif de quelques caractéristiques des documents plats, semi-structurés et structurés dans le cadre de la recherche d'information.

en recherche d'information [234].

Des observations menées auprès d'utilisateurs "réels" (non experts) en 2005 [139] ont montré que ces fonctionnalités spécifiques, en particulier la variabilité de la taille de l'unité de recherche, apportaient une véritable satisfaction en comparaison des moteurs de recherche classiques connus (de type Google[1] ou Yahoo !?[2]). La condition de cette satisfaction est une présentation appropriée des résultats, tenant compte du chevauchement éventuel des réponses et du contexte des éléments retournés.

2.3 Indexation

Il n'est pas envisageable d'effectuer une recherche directement dans la collection de documents à chaque nouvelle requête. Notamment de nombreux traitements de cette collection, préalables à la recherche, peuvent être effectués une seule fois (ou au moins à chaque fois que la collection change) : détermination de la pondération des mots, élimination des mots "vides" (c'est-à-dire très fréquents et/ou sans portée sémantique, tels que les mots de liaison [73, 207]), lemmatisation*, étiquetages syntaxiques ou sémantiques, reconnaissance des termes complexes, etc. (voir chapitre 3[3]).

indexation

L'indexation consiste à *"identifier l'information contenue dans tout texte et [à] la représenter au moyen d'un ensemble d'entités appelé index pour faciliter la comparaison entre la représentation d'un document et d'une requête."* [189, p. 12]. Cette représentation est formalisée par un *modèle de documents*. Il s'agit principalement, dans le cadre de la recherche d'information traditionnelle (documents plats), de collecter des mots-clés permettant un ciblage sémantique du document. Ce processus peut être effectué de façon manuelle, automatique ou semi-automatique (un expert ajustant un index composé automatiquement). Nous abordons ici rapidement l'indexation automatique à travers la pondération des termes collectés et la construction de l'index[4].

modèle de document

[1] http://www.google.com.
[2] http://search.yahoo.com.
[3] Section 3.5, page 74.
[4] Pour plus de détails, *Roussey et al.* [190] proposent un tour d'horizon complet de cette problématique.

2.3.1 Pondération des termes

La pondération des termes permet d'une part de déterminer l'importance relative des termes de la requête (leur pouvoir discriminant[1]), et d'autre part de quantifier leur importance dans la description sémantique du contenu d'un document[2].

tf - idf

Les techniques courantes de pondération des termes sont basées sur les notions de fréquence des termes dans un document (*tf* pour *term frequency*, mesure représentant l'importance locale d'un terme – fréquence relative) et de fréquence de ces termes dans l'ensemble des documents de la collection étudiée (on utilise l'inverse de cette mesure, simulant l'importance globale d'un terme, soit *idf* pour *inverse document frequency* – fréquence absolue) [197], notions elles-mêmes fondées sur des constatations linguistiques de *Zipf* [264] et *Luhn* [146], qui établissent une relation entre la fréquence des termes et leur pouvoir discriminant, c'est-à-dire leur importance dans la participation au sens d'un texte.

La mesure de l'*idf* part du principe que *"le nombre de documents pertinents à une requête est généralement faible [en comparaison du nombre total de documents], et donc les termes apparaissant fréquemment doivent nécessairement apparaître dans beaucoup de documents non pertinents. En revanche, les termes peu fréquents ont une plus grande probabilité d'apparaître dans des documents pertinents [et donc sont plus importants]."* [3] [118]

Le produit de ces deux mesures [117], utilisées avec des formules diverses, sert donc à indiquer l'importance d'un terme dans un document. Un terme répété de nombreuses fois dans un petit nombre de documents est probablement très significatif quant au sujet de ces documents.

Notons que l'utilisation brute de ces métriques est une simplification forte mais courante consistant à considérer que le poids d'un terme dans un document est indépendant du poids des autres termes du document [18, p. 25].

Dans le cadre de la recherche d'information dans les documents XML, les unités d'information sont imbriquées les unes dans les autres. Les mesures aux niveaux local et global vues ci-dessus nécessitent d'être adaptées. L'importance d'un terme dans un élément dépend fortement de l'importance de ce terme dans les sous-éléments qui le composent [144], et même des éléments situés autour [129]. Ceci a provoqué de nombreuses propositions conservant la mesure *tf.idf* [44, 209, 134] ou l'abandonnant [12, 48, 56, 129],

ief

avec notamment la définition de l'*inverse element frequency (ief)* [258, 95] pour généraliser la notion d'*idf*, en considérant les nœuds* d'un certain type plutôt que les documents.

2.3.2 Indexer le contenu et la structure

Nous ne nous attarderons pas ici sur les diverses méthodes utilisées pour créer un index (fichiers inversés, tableaux ou arbres de suffixes, fichiers de signatures [18, chap. 8]). Dans tous les cas le principe consiste à faire correspondre aux termes de la collection (parfois modifiés d'une façon ou d'une autre) un certain nombre d'informations utiles

[1] Par exemple, dans une requête portant sur l'*indexation des documents*, le mot *"indexation"* a plus d'importance discriminative que le mot *"document"*.

[2] On peut imaginer notamment qu'un terme apparaissant à de nombreuses reprises dans un document le décrira de façon plus judicieuse qu'un terme apparaissant rarement.

[3] "The number of documents relevant to a query is generally small, and thus any frequently occurring terms must necessarily occur in many irrelevant documents; infrequently occurring query terms, conversely, have a greater probability of occurring in relevant documents and should thus be considered as being of greater potential importance when searching a database." [118, p. 307]

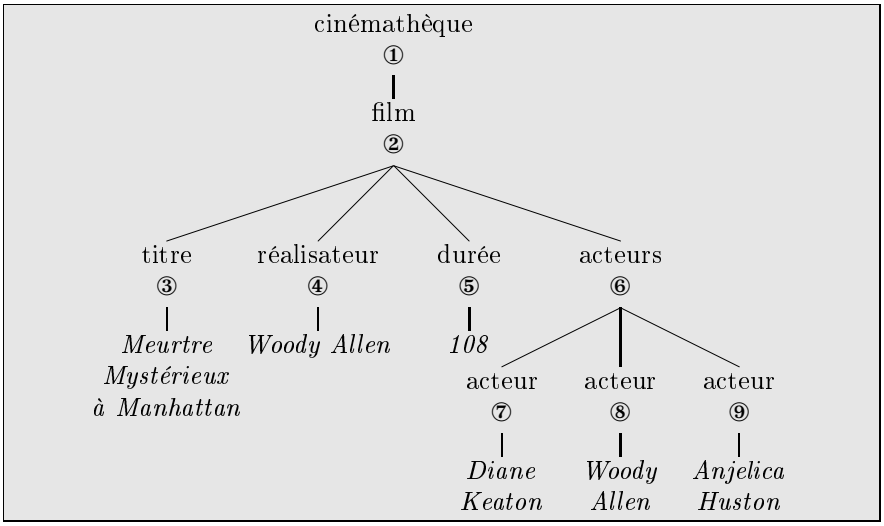


FIG. 2.2 – Exemple de document XML représenté sous forme arborescente, avec numérotation des nœuds.

au processus de recherche. Dans le cas d'une collection de multiples documents à indexer, l'information minimale à faire figurer est bien entendu l'identifiant des documents contenant le terme. Par la suite on peut ajouter les positions de ce terme dans chaque document, ceci permettant de mettre en place des algorithmes tenant compte de la proximité des termes concernés par la requête [263]. On peut également ajouter des informations d'ordre linguistique, comme la catégorie grammaticale du mot, jusqu'à aller vers des types d'indexation bien plus complexes comme l'utilisation de graphes conceptuels ou l'indexation sémantique [167].

En ce qui concerne les documents structurés (et en particulier XML), le schéma d'indexation dépend des capacités voulues pour le langage de requêtes qui fera l'interface entre l'utilisateur humain et le système. Dès lors que l'on souhaite effectuer une indexation structurée d'un document XML, la question des liens hiérarchiques entre les unités d'information se pose. A quel point l'information présente dans un élément est-elle répercutée dans ses ancêtres* ou ses descendants*? Ce problème se présentera à chaque étape de la recherche d'information. Au niveau de l'indexation, si beaucoup considèrent les éléments comme des *unités disjointes* sans connection, d'autres [56, 209, 214] propagent certains termes ou tous les termes d'un élément donné vers ses ancêtres (technique dite des *sous-arbres imbriqués*). Ainsi on peut considérer que dans l'exemple de la figure 2.2 l'élément `film` contient tous les mots de ses descendants. Nos exemples suivants tiennent compte de cette présence ou non de propagation.

unités disjointes

sous-arbres imbriqués

L'indexation de la structure passe par le choix du mode de représentation des éléments structurants. Parmi les possibilités présentées par *Luk et al.* [148], on trouve :

- *L'indexation par les champs*, où chaque terme est associé au nom de la balise dans laquelle il apparaît (voir table 2.2). Cette méthode permet de savoir dans quel contexte l'information textuelle est énoncée. Il est également possible d'indexer selon les *chemins*, c'est-à-dire d'indiquer le chemin d'accès logique plutôt que le nom de la balise [48] (`cinémathèque/film/titre` au lieu de `titre`).
- *L'indexation par les segments* qui divise les documents structurés en régions. Les modèles qui y sont associés [194, 45] sont très efficaces, mais antérieurs à l'apparition du format XML, et ils en supportent mal toutes les possibilités.
- *L'indexation par les arbres*, où un identifiant unique est attribué à chaque nœud

Terme	*Sans propagation*	*Avec propagation*
Meurtre	titre	cinémathèque, film, titre
Allen	réalisateur, acteur	cinémathèque, réalisateur, film, titre, acteurs, acteur
108	durée	cinémathèque, film, durée

TAB. 2.2 – Exemple d'indexation basée sur les noms des balises (associée au document de la figure 2.2).

Terme	*Sans propagation*	*Avec propagation*
Meurtre	③ (/cinémathèque[1]/film[1]/titre[1])	① (/cinémathèque[1]) ② (/cinémathèque[1]/film[1]) ③ (/cinémathèque[1]/film[1]/titre[1])
Allen	④ (/cinémathèque[1]/film[1]/réalisateur[1])	① (/cinémathèque[1]) ② (/cinémathèque[1]/film[1]) ④ (/cinémathèque[1]/film[1]/réalisateur[1]) ⑦ (/cinémathèque[1]/film[1]/acteurs[1]) ⑧ (/cinémathèque[1]/film[1]/acteurs[1]/acteur[2])
...	...	...

TAB. 2.3 – Exemple d'indexation basée sur les arbres (associée au document de la figure 2.2).

(élément) du graphe représentant le document XML [140]. Les termes sont donc associés à cet identifiant, ce qui permet de localiser de façon précise l'endroit où ces termes sont apparus et de retrouver les relations hiérarchiques entre les éléments [212]. L'identifiant unique peut également être, tout simplement, le chemin d'accès (XPath absolu, avec le numéro des éléments) de l'élément [259] (voir table 2.3).

Par ailleurs, au niveau de l'indexation purement structurelle (dans laquelle les termes n'ont plus leur place), et dans le cas où les nœuds sont identifiés de façon unique, des informations concernant les relations avec d'autres nœuds peuvent être ajoutées pour faciliter certaines opérations de navigation : une référence au nœud parent [258], au parent et aux enfants [227], au dernier (*rightmost*) descendant [209]. *Grust* [99] propose d'attribuer deux identifiants (numérotation par deux parcours en largeur d'abord et en profondeur d'abord) et de référencer le parent et le nom de la balise. *Hayashi et al.* [103] composent sept index "plats" représentant le titre, l'auteur, le sujet, les mots-clés, etc.

2.4 Interrogation

L'interrogation est, contrairement à l'indexation qui n'est effectuée qu'une seule fois, un processus qui se déclenche à chaque nouvelle requête. Elle consiste dans sa forme générale en trois phases qui sont la formulation de la demande de l'utilisateur, la recherche dans la collection de documents et la présentation des résultats. Comme nous l'avons vu dans l'introduction (voir figure 2.1), le cadre théorique de l'interrogation est essentiellement composé d'un *modèle de requête* et d'une *fonction d'appariement* (ou *fonction de correspondance*) :

modèle de requête

– Le *modèle de requête* décrit comment le besoin de l'utilisateur doit être retranscrit pour pouvoir être traité par le système, et comment ce système va procéder à

l'analyse de ce besoin.

– La *fonction d'appariement* détermine la pertinence d'un document par rapport à une requête, et permet éventuellement de classer les documents par ordre de pertinence supposée.

fonction d'appariement

Ces notions sont communes à la RI traditionnelle (*i.e.* concernant des documents plats*), et à la RI dans les documents semi-structurés. Ces principes de base doivent pourtant être adaptés aux caractéristiques propres des documents XML, notamment le changement de l'unité d'information considérée et la nécessité d'interroger la structure (voir section 2.2). Ceci impose de proposer aux utilisateurs de nouvelles manières de formuler leurs besoins d'information, et de traiter ces requêtes avec des techniques spécifiques.

2.4.1 Modèles traditionnels

Les modèles de recherche d'information dans les documents plats sont légion et les plus classiques ont été amplement décrits [133, chap. 4][18, chap. 2]. Nous donnons ici un aperçu rapide des trois grandes classes de modèles :

– Les modèles *ensemblistes*, en particulier le modèle *booléen*, dans lequel les documents sont vus comme un ensemble de termes associés à une variable booléenne, celle-ci étant positionnée à "vrai" pour un document si le terme y est présent. Les requêtes sont des suites de termes séparés par des opérateurs booléens, et les documents pour lesquels cette expression est vérifiée sont retournés à l'utilisateur (table 2.4). Ce modèle est dit *exact* car il ne renvoie aucun document répondant partiellement à la demande et ne permet pas de classer les résultats. Cependant il a été étendu [198] pour traiter les poids des termes et autoriser un classement.

modèle booléen

$t_1 \in \mathcal{D}$	$t_2 \in \mathcal{D}$	$t_3 \in \mathcal{D}$	**pertinence de $\mathcal{D}$**
0	x	x	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

TAB. 2.4 – Pertinence d'un document $\mathcal{D}$ pour la requête $q = t_1 \wedge (t_2 \vee t_3)$ avec le modèle booléen (où t_1, t_2 et t_3 sont des termes quelconques).

– Les modèles *algébriques*, notamment le modèle *vectoriel* [195], dans lequel documents et requêtes sont représentés par un vecteur dont les coordonnées sont calculées dans un espace euclidien à n dimensions (n étant le nombre de termes de l'index). La longueur de la projection d'un vecteur selon une dimension est proportionnelle au poids du terme qu'elle représente (figure 2.3). La pertinence d'un document par rapport à la requête est évaluée par le degré de similarité entre le vecteur du document et celui de la requête, cette similarité étant calculée par exemple par un produit scalaire, une mesure de distance métrique ou, plus couramment, la mesure du cosinus de l'angle entre les deux vecteurs. Ce modèle permet des requêtes moins expressives (sans opérateurs) que le modèle booléen, mais son efficacité est démontrée.

modèle vectoriel

– Les modèles *probabilistes* (initiés par *Maron et Kuhns* [153]) abordent la notion de probabilité de pertinence et de non-pertinence d'un document par rapport à une requête, à travers le "principe de classement probabiliste" (*Probability Ranking Principle – PRP*) [187].

modèle probabiliste

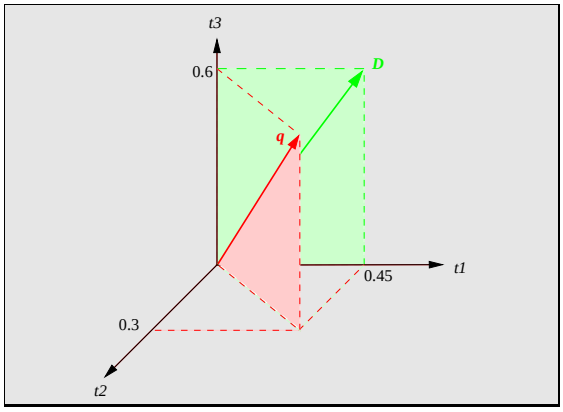


FIG. 2.3 – Représentation vectorielle d'une requête q contenant les trois termes t_1, t_2 et t_3, de poids repectifs 0.45, 0.3 et 0.6, et d'un document $\mathcal{D}$ ne contenant que les termes t_1 et t_3.

$$(\mathcal{I}(\text{"}\texttt{[play}\text{"}) \diamond \mathcal{I}(\text{"}\texttt{play]}\text{"})) \rhd (\mathcal{I}(\text{"}\texttt{birnam}\text{"}) \diamond \mathcal{I}(\text{"}\texttt{dunsinane}\text{"}))$$

FIG. 2.4 – Représentation avec le langage de *Clarke et al.* [45] de la requête : *Trouvez les pièces contenant "Birnam" suivi de "Dunsinane"*. Les crochets `'['` et `']'` représentent le début et la fin d'un élément structurel (ici la pièce de théâtre – *play*). L'opérateur $\diamond$ signifie "suivi de", tandis que $\rhd$ exprime l'inclusion.

Des modèles "alternatifs" de chacune de ces trois grandes classes sont décrits par *Beaza-Yates et Ribeiro-Neto* [18, chap. 2].

2.4.2 Les langages de requêtes

L'apparition des premiers langages de requêtes permettant de prendre en compte la structure des documents est antérieure à la création du langage XML. Certains ajoutaient des opérateurs logiques ou des prédicats à des requêtes "plates" pour spécifier des contraintes structurelles (inclusion, succession, opérations) [45, 170] (figure 2.4). Les plus nombreux, issus de la communauté des bases de données, ont répondu aux particularités de la RI structurée par des langages de requêtes s'approchant de SQL, par le format et les fonctionnalités (sélection d'éléments, mais aussi jointures, aggrégats, construction d'un résultat structuré, mise à jour des documents... – notamment UnQL [34] et Lorel [158])[1].

L'arrivée en 1997 du langage XML, très vite pressenti comme un standard important pour le stockage et l'échange de données, engendre un essor des recherches sur le sujet. Celles-ci sont à l'origine de nombreux nouveaux langages de requêtes, notamment à travers le *Query Language Workshop* dès 1998 [183], où sont proposés entre autres XML-QL [62] (figure 2.5), XQL [188] et XML-GL [39], et où de nombreux articles de positionnement face à la nouvelle problématique sont présentés [160] ; puis avec le premier atelier de la conférence ACM SIGIR consacré à XML en 2000 [16], où sont présentés par exemple XIRQL [76], une amélioration de XQL, et Tequyla [8] ainsi qu'un nouvel état de l'art [147]. Lorel est également adapté à XML [93], Quilt est créé pour interroger à la fois des documents XML et des bases de données [42] et les standards du

[1] Un tour d'horizon de l'état des recherches à cette époque a été proposé par *Abiteboul* [2] et *Florescu et al.* [71].

```
WHERE
    <film>
        <titre>$t</titre>
        <réalisateur>Woody Allen</réalisateur>
        <année>$a</année>
    </film>
IN "cinematheque.xml"
CONSTRUCT
    <woody>
        <film>$t</film><sortie>$a</sortie>
    </woody>
```

FIG. 2.5 – Représentation avec XML-QL [62] de la requête : *Trouvez les films de Woody Allen et leur date de sortie dans le document* ***cinematheque.xml*** (voir la structure utilisée dans la figure 1.2).

```
let $doc := (doc("cinematheque.xml"))
for $f in $doc/cinémathèque/film[./réalisateur = "Woody Allen"]
return
<woody>
     <film>$f/titre/text()</film><sortie>$f/année/text()</sortie>
</woody>
```

FIG. 2.6 – Représentation avec XQuery [253] de la requête : *Trouvez les films de Woody Allen et leur date de sortie dans le document* ***cinematheque.xml*** (voir la structure utilisée dans la figure 1.2).

World Wide Web Consortium (W3C [244]) se dessinent avec XPath [245], XSL [246] et XQuery [253] (figure 2.6 ; une adaptation de XQuery pour la recherche documentaire est également à l'étude [254]) [1].

Beaucoup de ces langages ne traitent que des requêtes exactes (présence/absence d'un élément structurel ou d'une chaîne de caractères), mais peuvent, pour certains, être étendus avec des prédicats plus "flous". De plus ils sont en général extrêmement complexes, ceci étant dû au nombre important de fonctionnalités possibles. Pour contrer cette difficulté, des aides graphiques à la construction d'une requête ont été proposées [40, 209], ainsi que des langages simplifiés, basés uniquement sur l'aspect recherche d'information (sans possibilité par exemple de construction de la sortie, d'agrégation, de mise à jour des données, mais avec de véritables fonctions de recherche textuelle) [36, 203]. C'est le cas de NEXI [230], qui est dérivé du langage XPath [245]. L'importance de ces deux langages pour le travail que nous décrirons par la suite nous amène à focaliser notre attention sur eux.

2.4.2.1 XPath

XPath [245] est un langage permettant de sélectionner un ensemble d'éléments XML (ou nœuds) répondant à certaines contraintes structurelles ou textuelles, ces contraintes étant représentées par des chemins (de même type que ceux qui permettent de se déplacer dans une arborescence de fichiers informatiques) et des filtres. Il est possible

[1] Des comparaisons de ces différents langages ont été effectuées par *Fernandez et al.* [70] et par *Bonifati et Ceri* [28].

```
//film[contains(./réalisateur/text() = "Woody Allen")]
```

FIG. 2.7 – Représentation avec XPath [253] de la requête : *Trouvez les films de Woody Allen.* Notons que la reconstruction de la sortie étant impossible, on ne peut pas recomposer un élément nouveau contenant deux nœuds de provenances diverses, comme dans les exemples précédents (figures 2.5 et 2.6).

notamment de caractériser les nœuds par leur position (absolue ou relative), leur type, leur contenu. XPath 1.0 distingue les nombres, les chaînes de caractères, les booléens.

Navigation. La navigation dans l'arborescence du document XML peut se faire selon différents axes. Parmi ceux-ci :

- *"child::"* représente les enfants* du nœud* contextuel (abréviation : chaîne vide) ;
- *"descendant-or-self::"* représente les descendants* du nœud contextuel ou ce nœud lui-même (abréviation : "/") ;
- *"parent::"* représente le parent* du nœud contextuel (abréviation : "..") ;
- *"ancestor::"* représente les ancêtres* du nœud contextuel ;
- *"following-sibling::"* représente les nœuds suivant le nœud contextuel et ayant le même parent que celui-ci ;
- *"attribute::"* représente les attributs* du nœud contextuel (abréviation : "@") ;
- *"self::"* représente le nœud contextuel lui-même (abréviation : ".") ;
- ...

Ainsi :

- `/child::para`, ou `/para` sélectionne tous les éléments `para` enfants de l'élément actif ;
- `/descendant-or-self::node()/child::para`, ou `//para` sélectionne tous les éléments `para` descendants de l'élément actif ;
- `parent::node()/child::para`, ou `../para` sélectionne tous les éléments `para` ayant le même parent que l'élément actif ;
- `.//liste/item` sélectionne tous les éléments `item` enfants d'un élément `liste`, lui-même descendant de l'élément actif ;
- `.//section/*` sélectionne tous les éléments, quels que soient leurs types, qui sont enfants d'un élément `section`, lui-même descendant de l'élément actif ;

Filtres. Il est possible de filtrer par :

- La position : `para[5]` sélectionne le cinquième enfant du nœud contextuel ayant le type `para` ;
- Le contenu structurel : `chapitre[titre]` sélectionne les éléments `chapitre` ayant au moins un enfant de type `titre` ;
- Le contenu structurel et textuel : `chapitre[titre="introduction"]` sélectionne les éléments `chapitre` dont un enfant de type `titre` a la valeur `"introduction"`.

Prédicats. Il est également possible d'utiliser des prédicats, dont certains sont prédéfinis dans le langage. Par exemple, le prédicat 'contains' :
`chapitre[contains(./titre, "Napoléon") and contains(./titre, "Wagram")]`
sélectionne les chapitres dont le titre contient les mots `"Napoléon"` et `"Wagram"`.

2.4.2.2 NEXI

La campagne d'évaluation INEX (cf. section 2.5.4) a choisi le langage XPath en 2003 pour permettre aux utilisateurs d'exprimer leurs besoins d'informations. Il s'est avéré que ce formalisme était trop complexe à utiliser, tandis que beaucoup de ses fonctionnalités étaient superflues. Ainsi 63 % des requêtes proposées étaient syntaxiquement ou sémantiquement incorrectes, et jusqu'à 12 tours de corrections ont été nécessaires pour parvenir à un ensemble satisfaisant de requêtes. En 2004, le langage NEXI (Narrowed Extended XPath I [230]) a donc été proposé en remplacement. Grâce à ce nouveau formalisme, le taux d'erreur est tombé à 12 %.

NEXI

NEXI est un sous-langage de XPath. Il le restreint en grande partie :

- Seul l'axe *descendant* est conservé ;
- Seules deux étapes de navigation sont permises (`//article[...]//section[...]` est valide, tandis que `//article[...]//section[...]//paragraph[...]` ne l'est pas).
- Le filtrage selon le contenu est reservé aux données numériques (ex. : `//article[.//année = 1984]`) ;
- Le filtrage selon la position ainsi que selon la structure seule sont impossibles ;
- Les prédicats prédéfinis sont interdits.

Un prédicat binaire vague `about(P, C)` est cependant ajouté. Il est considéré vrai si `P` *parle de* `C`. Dans la partie `C`, NEXI accepte une liste de mots-clés. Un mot-clé peut être :

- Un simple terme ;
- Un groupe de mots, entouré alors par des guillemets ;
- Un de ces deux éléments précédé par les opérateurs `'+'` (signifiant une importance particulière du mot-clé) ou `'-'` (signifiant que le mot ne doit pas apparaître).

NEXI est donc conçu exclusivement pour la tâche de recherche d'information "basique" qui consiste à retourner des éléments satisfaisant un besoin d'information, et ce sans construire de sortie nouvelle et sans opérer de jointure ni d'agrégation. Il est présenté comme le "langage de requête le plus simple qui puisse jamais fonctionner" (*"The Simplest Query Language That Could Possibly Work"* [172]).

Des exemples de requêtes sont donnés à la figure 2.8, ainsi que dans la section 2.5.4.2. Notons qu'une requête NEXI possède deux parties distinctes :

- la *cible*, l'élément qui doit être retourné à l'utilisateur, ainsi que les indications concernant son contenu ;
- le *support*, les autres contraintes structurelles et textuelles, permettant de mieux retrouver la cible.

cible

support

Ainsi, dans la requête *"Trouvez les films dont Woody Allen est à la fois le réalisateur et l'acteur principal"*, le film est la cible tandis que le réalisateur et l'acteur sont le support. Dans les requêtes de la figure 2.8, la cible a été soulignée. Que ce soit pour la cible ou le support, l'élément peut être de type `'*'`, c'est-à-dire de type indifférent.

Un des travaux que nous décrivons dans la seconde partie de ce mémoire a consisté en une extension de NEXI pour apporter de nouvelles fonctionnalités, utiles notamment dans le cadre de la mise en œuvre de techniques de traitement automatique du langage naturel (voir chapitre 6).

12. Les sections (`sec`) qui parlent de moteurs de recherche sur Internet (*"Internet search engine"*) dans des articles publiés en 2001 ou 2002 :

```
//article[.//pdt//yr = 2001 or .//pdt//yr = 2002]
  //bdy//sec[about(., internet search engine)]
```

70. Les résumés (`abs`) traitant de recherche d'information (*"information retrieval"*) et de bibliothèques numériques (*"digital libraries"*) :

```
//article[about(.//fm//abs, +"information retrieval"
                           "digital libraries")]
```

72. Les éléments de corps d'article (`bdy`) à propos des systèmes de prédictions météorologiques, et écrits par des auteurs affiliés aux Etats-Unis :

```
//article[about(.//fm//au//aff, United States of
America)]//bdy//*[about(., weather forecasting systems)]
```

FIG. 2.8 – Exemples de requêtes NEXI, extraites des ensembles de requêtes utilisés à INEX (requêtes 12, 70 et 72). Dans chaque cas la partie cible a été soulignée.

2.4.3 Le traitement des requêtes

La majorité des approches orientées RI présentées dans la littérature sont des adaptations des modèles traditionnels. Ces adaptations visent essentiellement à tenir compte de certaines spécificités des documents XML, à savoir les tailles très différentes que peuvent avoir les éléments, le fait que ces éléments peuvent être imbriqués les uns dans les uns (et donc non indépendants), ainsi que, plus rarement, les informations données par la structure elle-même.

Nous n'allons pas détailler les différentes propositions faites pour traiter les requêtes, mais plutôt donner un rapide tour d'horizon des caractéristiques mises en jeu. Nous présenterons ensuite le système GPX, du Queensland University of Technology, non parce que nous le jugeons représentatif, mais parce que nous avons nous-même utilisé ce système dans une partie de nos recherches.

2.4.3.1 Extension des modèles traditionnels

Les différents types de modèles (vectoriel, probabiliste, de langage, etc.) sont étendus de diverses façons pour tenir compte de la structure ou, le plus souvent, des impacts de la présence d'une structure sur les caractéristiques des unités d'information. On ajoute ainsi des paramètres supplémentaires pour ajuster les formules classiques : le nombre d'enfants d'un élément est important [83], ainsi que son type (à travers des informations sur son importance sémantique [83, 64], la fréquence de ce type d'élément dans la collection [209, 95], l'importance d'un terme dans les autres éléments du même type [95] ou tout simplement des index séparés par types d'éléments [156]).

Lorsque les éléments sont considérés comme des unités disjointes (cf. section 2.3), les scores de pertinence sont *propagés* des nœuds feuilles vers leurs ancêtres en diminuant progressivement l'influence des termes pour tenir compte du besoin de spécificité[1]. Les

[1] Cette propagation permet de rendre compte du fait que du texte contenu par un nœud feuille est également contenu par ses ancêtres. Le score d'un élément doit donc participer au score de son parent

scores sont calculés par diverses formules (maximum des poids [12], mesure d'entropie [56], somme pondérée [88, 110], facteur d'augmentation [77]). *Lalmas* [136] utilise la théorie de Dempster-Shafer qui permet de combiner les *croyances* de plusieurs doxels par rapport à une requête.

L'information structurelle en elle-même subit rarement un traitement spécifique. Souvent les éléments retrouvés par la recherche textuelle sont simplement filtrés [214, 227] (ou pondérés) en fonction des contraintes structurelles de la requête. Pourtant certains utilisent des représentations structurées de la requête et des documents et tentent de calculer une similarité entre ces deux "arbres". Ainsi *Fuzzy XPath* [32] adapte XPath pour permettre d'ordonner un ensemble de chemins. D'autres adaptent les techniques de *"tree matching"* pour comparer les nœuds [209] ou définissent une distance entre elements [262, 180].

De nombreux modèles récents sont décrits dans les actes résultant des campagnes INEX [78, 79, 81, 80] ainsi que ceux des ateliers consacrés au sujet à la conférence SIGIR [15].

2.4.3.2 Le système GPX

Le système GPX de *Shlomo Geva* (Queensland University of Technology) [89, 87, 88] est basé sur une indexation par fichiers inversés héritée de la RI plate et sur deux formules heuristiques de propagation des poids des termes. Le langage de requêtes qui lui est associé est NEXI (cf. section 2.4.2.2).

Indexation. Le schéma d'indexation considère les éléments comme des unités disjointes et conserve le chemin absolu des doxels contenant les termes (indexation par les arbres, cf. section 2.3). Chaque terme présent dans un document est associé à l'identifiant de ce document, à son chemin absolu (XPath) et à sa position dans le contenu textuel de l'élément (tableau 2.5).

Tab. 2.5 – Fichier inversé du système GPX.

Document	XPath	Position
e1303.xml	article[1]/bdy[1]/sec[6]/p[6]	23
e1303.xml	article[1]/bdy[1]/sec[7]/p[1]	12
e2404.xml	article[1]/bdy[1]/sec[2]/p[1]/ref[1]	1
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]	3
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[14]/pp[1]	2
g5742.xml	article[1]/fm[1]/abs[1]	7

En pratique, pour éviter les redondances (des termes et des chemins), l'index est stocké dans une base de données relationnelle comprenant quatre tables (trois tables *termes*, *chemins* et *documents* reliées ensemble par une table *liste*).

Traitement des requêtes. Le système considère que les nœuds feuilles* contiennent l'essentiel de l'information textuelle. Un score est calculé pour ces nœuds, puis les valeurs sont propagées vers le haut jusqu'à la racine*.

Le score de pertinence des feuilles est calculé avec la formule (2.1), que ce soit pour

(voir l'explication du système GPX à la section suivante).

les éléments concernant le support* ou la cible* de la requête.

$$L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \tag{2.1}$$

Ici n est le nombre de termes de la requête présents dans l'élément, K est un petit entier ($K = 5$ dans le système) servant à favoriser les éléments contenant plusieurs termes de la requête. La somme est calculée sur l'ensemble des termes, où t_i est la fréquence du $i^{ème}$ terme de la requête dans la feuille et f_i la fréquence de ce même terme dans l'ensemble de la collection. Ainsi les termes rares contribuent plus au score que les termes communs.

Le score des éléments feuilles est alors utilisé pour calculer celui de leurs ancêtres*. Il s'agit de propager la pertinence des éléments (l'*exhaustivité*) tout en favorisant la *spécificité*. Ceci est assuré par la formule (2.2).

$$R = D(n) \sum_{i=1}^{n} L_i \tag{2.2}$$

Avec :

- n = le nombre d'enfants* jugés pertinents de l'élément (avec $L > 0$)
- $D(n) = \begin{cases} 0.49 & \text{si } n = 1 \\ 0.99 & \text{sinon} \end{cases}$
- L_i = le score du $i^{ème}$ enfant

Ainsi la fonction $D(n)$ assure que le score propagé décline fortement lorsqu'un élément n'a qu'un seul enfant pertinent (en effet dans ce cas il vaut mieux sélectionner cet enfant directement). En revanche un nœud comprenant plusieurs enfants ayant reçu un score non nul sera mieux classé que ses enfants (voir figure 2.9).

Les scores des parties support et ceux de la partie cible sont ensuite additionnés. Par exemple, pour la requête :

```
//A[about(.//B,C)]//X[about(.//Y,Z)]
```

le score du support `//A//B` (avec la condition `C`) sera ajouté à celui de la cible `//A//X//Y` (condition `Z`).

Les contraintes structurelles font l'objet d'un simple filtrage (vérification du type de la balise) si l'on souhaite les appliquer de façon stricte.

Le système GPX a participé aux différentes campagnes d'INEX avec de très bons résultats[87, 88].

2.5 Evaluation

Pour évaluer et comparer les systèmes de recherche d'information, les critères de performance (ou d'efficience, selon un anglicisme usité) habituels en informatique, c'est-à-dire les mesures du temps de réponse et de l'espace disque occupé, ne sont plus suffisants (même si les applications des modèles se doivent de considérer ces aspects). En plus de la performance, d'autres indicateurs sont l'*environnement* (notamment l'ergonomie du système en termes de facilité d'utilisation et de présentation des résultats) mais surtout les *résultats* (*outcome*) [159, chap. 16]. En effet, il n'est pas possible de formuler un jugement qualitatif binaire sur la sortie du système (sortie correcte ou sortie incorrecte). Dans ce qui suit, nous considérons que le critère unique d'évaluation

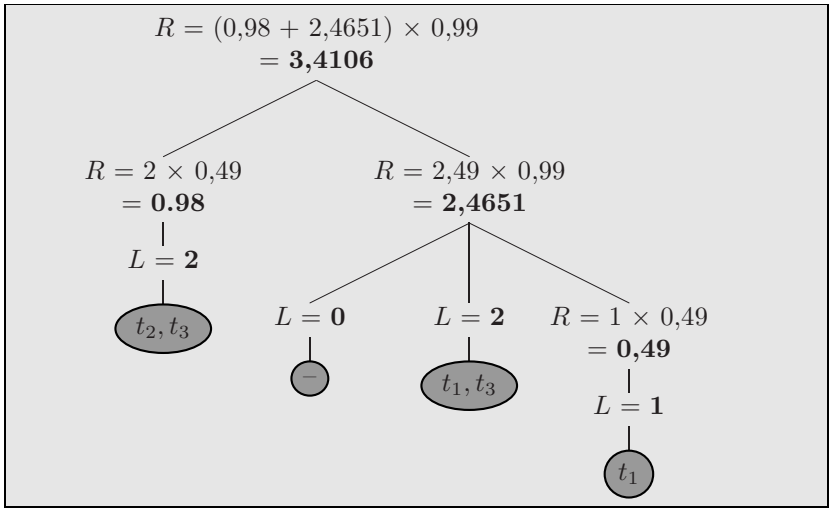


FIG. 2.9 – Système GPX. Propagation des scores des nœuds feuilles vers les nœuds supérieurs, pour la requête concernant les termes t_1, t_2 et t_3. On considère pour simplifier que chacun de ces termes a un poids de 1.

des systèmes de recherche d'information est leur *efficacité*, c'est-à-dire leur capacité à trouver des documents répondant au besoin de l'utilisateur [120].

2.5.1 L'évaluation ou la dictature de l'humain

Pertinent *: Qui se rapporte exactement à ce dont il est question ; approprié*
Dictionnaire Larousse.

Comme nous l'avons déjà noté, la plupart des paramètres en jeu dans le processus de recherche d'information sont des paramètres humains. Notamment c'est un humain qui pose une question et reçoit (et juge) la réponse. Un être humain est, en l'occurrence, caractérisé par :

- sa subjectivité : pour un même besoin d'information, deux humains formuleront des requêtes (très) dissemblables ; ils estimeront la qualité des réponses retournées de deux façons tout aussi différentes. Les variables entrant en jeu sont notamment le passé des personnes, leur caractère, leur connaissance initiale du sujet, leur niveau de langue, leur niveau d'exigence.
- sa versatilité : pour un même besoin d'information, un seul humain formulera des requêtes (très) dissemblables à divers moments de sa journée ; il estimera la qualité des réponses retournées de plusieurs façons tout aussi différentes. Il s'agit là de variations d'humeur, d'éveil, de pensées parasites ou même d'évolution de la connaissance du sujet entre deux instants.

Déjà ardue, la tâche apparaît donc insurmontable quand on se souvient que les documents dans lesquels l'information est recherchée sont eux-mêmes le fruit d'humains subjectifs et versatiles, et qu'à toutes ces notions s'ajoutent les phénomènes linguistiques de variations morphologiques, lexicales, sémantiques, syntaxiques (voir chapitre 3).

Pourtant seuls des humains peuvent formuler des requêtes (c'est la finalité de la discipline), seuls des humains peuvent écrire les documents et seuls des humains peuvent juger les réponses. Si ce n'était pas le cas les systèmes de recherche d'information n'auraient plus besoin d'être perfectionnés.

Toutes ces réflexions mènent à la conclusion qu'il n'existe pas un seul ensemble possible de réponses à la question posée, et même qu'il n'existe pas de bonne réponse dans l'absolu.

pertinence

En outre, la notion de *pertinence* d'un document ou d'une portion de document, qui est au cœur du principe de l'évaluation des systèmes de RI, est relativement floue. Souvent décrite comme un synonyme d'*utilité* ou de *satisfaction* de l'utilisateur [113], elle est elle-même un phénomène social et cognitif complexe [52] et n'a pas de définition consensuelle. Des formalisations ont été proposées [52, 257], ainsi que des études sur les variables affectant les jugements humains sur la pertinence [184], ou les diverses facettes qu'elle peut prendre [201, 202, 163, 164], mais le fait est que la pertinence est un concept insaisissable car, comme son inventeur, subjectif et versatile.

En l'absence de possibilité d'attribution d'une valeur binaire par une seule personne à un moment donné, elle serait dans l'idéal le résultat d'une *"tendance consensuelle, centrale"* de plusieurs juges humains [22, chap. 4], ce qui est complexe à obtenir en pratique.

A ces nombreux problèmes vient s'ajouter celui de la taille constamment croissante de la masse de documents mise en jeu. La collection d'INEX 2005 compte 760 méga-octets[1], celle d'INEX 2006 environ 10 giga-octets (100 avec les images), et certaines collections utilisées par TREC [239] dépassent la dizaine de giga-octets, soit l'équivalent du texte de plusieurs encyclopédies. Il est donc depuis longtemps hors de question qu'un expert humain cherche dans l'ensemble de la collection les réponses pertinentes à une requête donnée. Pourtant les "bonnes" réponses doivent être connues si l'on veut

pooling

évaluer correctement les systèmes. Pour pallier (de façon très imparfaite) ce problème insoluble la méthode du *pooling* a été mise en place [219, 240], qui consiste en une fusion "intelligente" des résultats des participants en un ensemble de documents qui devra être jugé. Les documents non jugés sont considérés non pertinents. Pourtant *Zobel* [265] a estimé qu'au mieux 50 à 70 % des documents pertinents étaient détectés par le pooling.

Autant dire qu'une évaluation objective et impartiale des systèmes de RI est une tâche perdue d'avance, et que toutes les méthodologies, toutes les procédures, toutes les métriques mises en place ne seront jamais que des pis-aller[2].

2.5.2 Mesures d'évaluation des systèmes classiques

Une mesure d'évaluation appropriée doit estimer la faculté des systèmes à trouver des documents pertinents, sans pour autant favoriser ceux qui retournent plus de documents (et donc potentiellement plus de documents pertinents) [224]. Il est donc nécessaire de considérer aussi les documents non pertinents dans une métrique.

Les mesures combinées de *rappel* et de *précision* [131] sont les mesures les plus simples et les plus utilisées qui tiennent compte de ces deux impératifs (retrouver des documents pertinents, ne pas retrouver de documents non pertinents).

Soient, pour une requête donnée et un système donné (voir figure 2.10) :

- P l'ensemble des documents de la collection qui sont pertinents par rapport à cette requête.
- R l'ensemble des documents retrouvés par le système pour cette requête.
- $P_R = P \cap R$, l'ensemble des documents pertinents retrouvés par le système.

[1] Voir la section 2.5.4.

[2] *Swanson* [223] dresse une liste de neuf "postulats d'impuissance" (*"Postulates of Impotence"*) dans un article tristement intitulé : "La recherche d'information et le futur d'une illusion" (*"Historical Note : Information Retrieval and the Future of an Illusion"*). Seuls trois "postulats de fertilité" (*"Postulates of Fertility"*) viennent contrebalancer ce noir tableau.

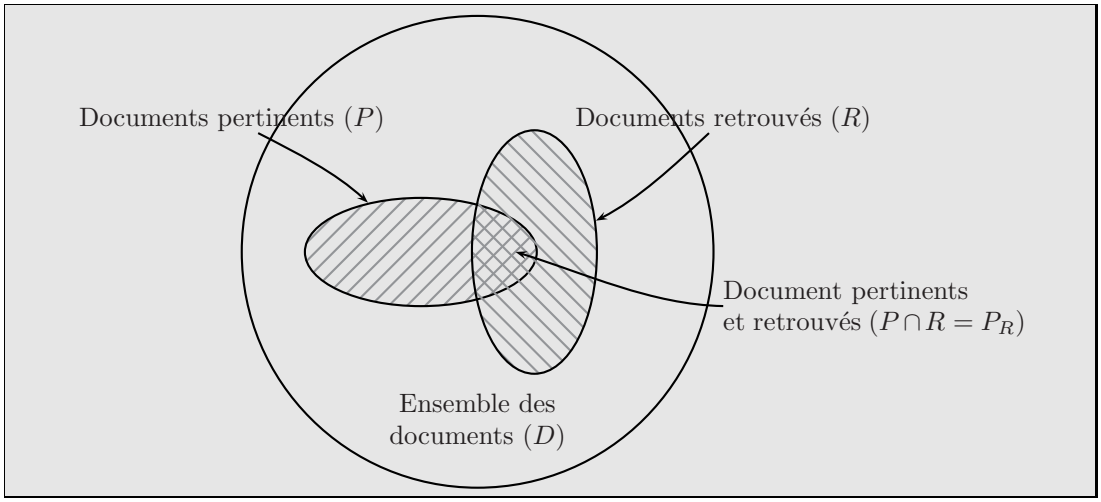


FIG. 2.10 – Répartition des documents face à une requête.

précision

La précision mesure le taux de documents pertinents parmi tous les documents retrouvés par le système :

$$précision = \frac{|P_R|}{|R|}$$

rappel

Le rappel mesure le taux de documents pertinents retrouvés par le système parmi l'ensemble des documents pertinents de la collection :

$$rappel = \frac{|P_R|}{|P|}$$

Ces deux valeurs sont indissociables et concurrentes, puisque souvent l'amélioration de l'une d'elle se fait au détriment de l'autre. Une caractéristique intéressante est d'ailleurs l'évolution de la précision en fonction du rappel (précision quand 10 % des documents pertinents ont été retrouvés, puis 20 %, etc.). Ces informations sont regroupées dans la *courbe de rappel-précision*.

Universellement utilisés, précision et rappel posent cependant quelques problèmes. Parmi eux :

- La mesure du rappel nécessite de connaître en théorie l'ensemble des documents pertinents par rapport à une requête dans toute la collection. Nous avons vu que cela était impossible.
- Le rappel n'est pas une mesure qui reflète la satisfaction de l'utilisateur moyen. Celui-ci est rarement intéressé par la connaissance de l'ensemble des documents pertinents, mais souhaite plutôt trouver "vite" (donc bien classés) quelques documents pertinents (voire un seul).

Pour pallier ces inconvénients, de nombreuses métriques héritées de la précision et du rappel ont été proposées. Ces mesures sont décrites dans divers supports [18, chap. 3][133, chap. 8].

Nous allons nous intéresser plus précisément aux mesures d'évaluation appliquées pour les systèmes de RI dans les documents XML.

2.5.3 Mesures d'évaluation des systèmes de RI structurée

L'évaluation des systèmes de recherche d'information structurée pose de nouvelles difficultés en termes de métriques et de comportements humains :

- On considère que l'unité d'information est l'élément XML, ou *doxel*. L'évaluation doit donc porter sur chaque élément. Dans la collection INEX 2005, à laquelle nous ferons souvent référence (cf. section 2.5.4), chaque document contient en moyenne 1500 doxels. Même si le volume de données reste inchangé entre un contenu plat et le même contenu structuré, le nombre d'éléments qu'il est potentiellement nécessaire de considérer augmente de façon non négligeable la charge de travail des juges humains.
- Comme nous l'avons déjà remarqué, deux dimensions sont nécessaires à l'estimation de la pertinence d'un élément : la première dimension permet de déterminer si (ou *à quel point*) le doxel contient des informations intéressantes au regard de la requête, la seconde si (ou *à quel point*) sa taille est appropriée, c'est-à-dire s'il aurait pu être ou non possible, en choisissant un élément descendant ou ancêtre, de concentrer de manière plus précise l'information pertinente.

chevauchement

- Il est normal de considérer que si un élément de petite taille est pertinent, alors ses ancêtres le sont aussi (même s'ils sont moins *spécifiques*). Pourtant le retour par un système de plusieurs éléments se chevauchant (c'est-à-dire, les uns étant ancêtres des autres) pose un problème. Si l'on intègre de nombreux éléments imbriqués dans la *base de rappel* (la liste de tous les doxels pertinents), cela encourage les concepteurs des systèmes à renvoyer des informations très redondantes. Ceci est contre-productif vis-à-vis de l'utilisateur (qui ne sera pas satisfait de visualiser la même information plusieurs fois en parcourant la liste des résultats[1]), mais aussi pour la comparaison entre les systèmes [127].
- Enfin, nous avons déjà abordé le problème du caractère auto-explicatif des éléments retournés (2.2.1). Il semble impossible de s'assurer qu'un élément permet de répondre à la requête de façon indépendante du reste du document. L'entorse à cette règle doit tout de même être limitée, pour ne pas nuire à la lisibilité des résultats.

Les métriques d'évaluation sont loin d'être stabilisées et consensuelles pour la RI structurée. Leur évolution est indissociable de la campagne INEX. Toutes les propositions ont été faites dans ce cadre, et les métriques retenues chaque année pour l'évaluation "officielle" ont eu de fait un succès supérieur. De plus, beaucoup des formules proposées dépendent fortement des tâches spécifiques à INEX d'une part et du mode de jugement de l'exhaustivité et de la spécificité d'autre part. C'est pourquoi, après cette introduction de la problématique générale, nous abordons les métriques au sein de la section consacrée à INEX (2.5.4.4).

2.5.4 La campagne d'évaluation INEX

INEX

INEX (INitiative for the Evaluation of XML retrieval [149] [2]) est une campagne d'évaluation visant, à la manière de TREC (Text REtrieval Conference [239] [3]) ou CLEF (Cross Language Evaluation Forum [174] [4]) pour les documents plats, à permettre une

[1] Cependant cette difficulté n'est pas nouvelle, puisque des informations rigoureusement identiques (des citations par exemple) peuvent apparaître dans des documents différents. Elle a toujours été contournée (y compris à INEX) en demandant aux évaluateurs de considérer les documents indépendamment les uns des autres.

[2] http://inex.is.informatik.uni-duisburg.de/2005/.

[3] http://trec.nist.gov.

[4] http://www.clef-campaign.org.

évaluation et une comparaison aussi rigoureuses que possible des systèmes de recherche d'information dans les collections XML orientées documents.

Ainsi INEX fournit un ensemble de documents, un ensemble de requêtes et des *jugements de pertinence*, c'est-à-dire les estimations humaines des éléments pertinents concernant chaque requête. Nous allons consacrer quelques pages à décrire le fonctionnement global d'INEX, qui est devenu l'organe central et le promoteur majeur des progrès en recherche d'information XML.

2.5.4.1 Les documents utilisés par INEX en 2005

La collection regroupait lors de la campagne de 2005 (quatrième édition) des articles scientifiques issus de journaux fournis par l'IEEE Computer Society, écrits en XML avec une DTD unique. 16819 articles balaient ainsi de nombreux domaines de l'informatique entre 1995 et 2004. Des extraits d'un article sont fournis à la figure 2.11.

Dans un article type, des méta-données concernant la publication (auteur, titre, journal mais aussi numéro ISBN, numéros de pages, prix, etc.) sont d'abord fournies, puis vient le texte lui-même, séparé en divers types de sections (`'sec'`), sous-sections (`'ss1'`, `'ss2'`), paragraphes (`'p'`), figures (`'fig'`), avec des balises distinguant notamment les références à des figures, à la bibliographie, à des notes de bas de page (`'ref'`), ainsi que des indications de forme, comme la mise en caractères italiques (`'it'`) ou gras (`'b'`). Puis les annexes sont essentiellement composées des éléments bibliographiques cités dans le corps du document et des indications concernant les auteurs.

La DTD comprend 192 balises différentes et un article est composé en moyenne d'environ 1500 éléments.

2.5.4.2 Les requêtes

Une requête est la représentation la plus fidèle possible d'un besoin d'information. Dans INEX ces requêtes (appelées aussi *topics*) sont proposées par les participants, donc experts du domaine de la recherche d'information et possédant des connaissances diverses en informatique (domaine de la collection), ce qui présente un biais non négligeable (d'autant plus que les participants sont également les juges – cf. section 2.5.4.3). Cependant il est demandé que ces requêtes *"reflètent les besoins réels de systèmes opérationnels"*[1].

topic

De façon générale, il est possible de distinguer deux types de requêtes :

- Celles concernant le contenu seul, pour les utilisateurs qui ignorent (ou qui souhaitent ignorer) la structure des documents.
  Exemple : *Les algorithmes de codage pour la compression de texte ou d'index.*[2]
- Celles concernant le contenu et la structure, incorporant donc des contraintes structurelles.
  Exemple : *Nous recherchons des résumés de documents concernant le data mining écrits par Jiawei Han.*[3]

contenu seul

contenu/ structure

Ceci a conduit à un format de requêtes dont des exemples sont donnés dans les figures 2.12, 2.13 et 2.14, et dont les champs importants sont [215] :

- *description*, une courte description du besoin d'information, rédigée en Anglais.

[1] "Topics should reflect the real need of operational systems." [215]
[2] "Any type of coding algorithm for text and index compression." (Topic 162, INEX 2004).
[3] "We are looking for the abstracts of the documents about data mining and written by Jiawei Han." (Topic 132, INEX 2004).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE article SYSTEM "/usr/projects/inex/2005/inex/dtd/xmlarticle.dtd">
<article>
 <fno>A1067</fno>
 <doi>10.1041/A1067s-1998</doi>
 <fm>
  <hdr>
   <hdr1>
    <ti>IEEE ANNALS OF THE HISTORY OF COMPUTING</ti>
    <crt><issn>1058-6180</issn>/98/$10.00
          <cci><onm>&copy; 1998 IEEE</onm></cci>
    </crt>
   </hdr1>
   <hdr2>
    <obi><volno>Vol. 20</volno>,<issno>No. 1</issno></obi>
    <pdt><mo>JANUARY-MARCH</mo>
      <yr>1998</yr></pdt>
    <pp>pp. 67-76</pp>
   </hdr2>
  </hdr>
  <tig>
   <atl>Calculators</atl>
   <pn>pp. 67-76</pn>
  </tig>
 </fm>
 <bdy>
  <sec>
   <st>THE CALCUMETER</st>
   <ip1><b>by Robert Otnes</b></ip1>
   <ip1>
      The Calcumeter adder appeared shortly after 1900, prospered for a while,
      and then apparently stopped production by 1920. Before its demise, as many
      as 100,000 adders may have been made. The Calcumeter is an interesting,
      well-made, small adding machine that is sought by collectors.
   </ip1>
   <lc>
    <li>
     <p>
       Walsh was the inventor. He held the two patents on the machine in his name.
       Early machines are labeled Morse and Walsh, whereas later ones are labeled
       only Morse, perhaps indicating that Walsh dropped out of the business. His
       name is mentioned in the 1905 <it>Trenton City Directory</it> (TCD)
       advertisement for the Calcumeter, but it was omitted in 1906.
     </p>
    </li>
    <li>
     <p>
       Morse apparently managed the business. At least one of the Calcumeter ads
       has his picture. The city directories mention him as being business manager
       for the State Department of Public Instruction in 1919; this is at the end
       of the era of the Calcumeter.
     </p>
    </li>
```

FIG. 2.11 – Exemple de document XML de la collection INEX (extraits, 1/2)

```
    </lc>
    <ss1>
      <st>The Machine</st>
      <ip1>
        Calcumeters are shown in <ref rid="a10671" type="fig">Figs. 1</ref> and
        <ref rid="a10672" type="fig">2</ref>.<ref rid="a10671" type="fig">Fig. 1
        </ref> shows a relatively early machine without the reset mechanism. This
        machine is marked <scp>THE CALCUMETER</scp>.
        <ref rid="a10672" type="fig">Fig. 2</ref> shows a later example that
        includes a reset mechanism. Note that this machine is marked <scp>THE
        STANDARD DESK CALCUMETER</scp> rather than <scp>THE
        CALCUMETER</scp>. Both machines are about the same size.
      </ip1>
    </ss1>
  </sec>
</bdy>
<bm>
  <ack>
    <h>Acknowledgment</h>
    <ip1>
      I would like to thank Rolf Ziegler, my colleague in the Section on IBM
      History of the IBM Club B&ouml;blingen.
    </ip1>
  </ack>
  <bib>
    <bibl>
      <h>References</h>
      <bb id="biba10671">
        <au>
          <fnm>M.T.</fnm><snm>Olivier</snm>
        </au>
        <atl>"Machines a Calculer du Docteur Roth,"</atl>
        <ti>Bulletin De La Societe D'Encouragement Pour L'Industrie Nationale,</ti>
        <pdt><mo>Sept.</mo><yr>1843,</yr></pdt>
        <loc><cty>Paris.</cty></loc>
      </bb>
      <bb id="biba10672">
        <au>
          <fnm>E.</fnm><snm>Martin</snm>
        </au>
        <ti>Die Rechenmaschinen und ihre Entwicklungsgeschichte,</ti>
        <obi>1. Band, 1924,</obi><pp>p. 63</pp>
        <pdt><yr>1992.</yr></pdt>
      </bb>
    </bibl>
  </bib>
</bm>
</article>
```

Fig. 2.11 – Exemple de document XML de la collection INEX (extraits, 2/2)

```
<inex_topic topic_id="205" query_type="CO+S" ct_no="12">
     <InitialTopicStatement>McLuhan</InitialTopicStatement>
     <title>marshall mcluhan</title>
     <castitle>//bdy//*[about(., "Marshall McLuhan")]</castitle>
     <description>
          Find information about the relevance of Marshall McLuhan's ideas for current
          digital technologies.
     </description>
     <narrative>
          I am writing an essay on the influence of new media icon Marshall McLuhan
          on digital technologies. I'm seeking information describing how McLuhan's views
          have influenced current digital technologies. To be relevant, a retrieved item should
          discuss some aspect of Marshall McLuhan's visionary ideas or famous one-liners
          in the context of current digital technologies. Retrieved elements that merely cite
          some of McLuhan's work are non-relevant, as are elements that discuss ideas not
          originating from McLuhan.
     </narrative>
</inex_topic>
```

FIG. 2.12 – Exemple de requête CO+S (Topic 205 d'INEX 2005).

- *narrative*, une explication détaillée du besoin d'information, rédigée en Anglais, expliquant les raisons de la requête, ce qui sera jugé pertinent et éventuellement ce qui ne le sera pas.
- *title*, une liste de mots-clés concernant le besoin d'information.
- *castitle*, un titre écrit en NEXI (cf. section 2.4.2.2).

Il est important de préciser que la *description* doit contenir *"les mêmes termes et les mêmes conditions structurelles que ceux apparaissant dans les parties* title *et* castitle*"*[1].

En 2005, les organisateurs ont souhaité étudier l'impact des contraintes structurelles sur l'efficacité de la recherche. Ils ont donc ajouté la possibilité de donner des indices concernant la structure des résultats souhaités, y compris dans les requêtes sur le contenu seulement. Cette réflexion a mené aux deux types de requêtes suivants :

CO + S

- *Contenu seulement et structure* (*"Content-Only + Structure"* – *CO+S*), dont le nom étrange est dû à l'évolution historique de la tâche. Il qualifie les requêtes se rapportant au contenu textuel seul (*title* et *description*), mais pour lesquels l'utilisateur peut malgré tout "aider" le système par des indications structurelles (dans le *castitle* – voir un exemple figure 2.12).

CAS

- *Contenu et Structure* (*"Content and Structure"* – *CAS*) permet aux utilisateurs connaissant la structure des documents de formuler des contraintes concernant les types d'éléments souhaités (voir un exemple figure 2.13).

De plus, dans le but d'évaluer de façon précise l'action de chaque contrainte structurelle, chaque requête comportant plusieurs références à la structure (résumé + auteur dans le topic 132 sur le data mining ci-dessus, deux paragraphes dans l'exemple de la figure 2.13) doit être accompagnée d'une sous-requête par contrainte (une des sous-requêtes correspondant au topic 256 de la figure 2.13 est reproduite à la figure 2.14).

[1] "The description [...] should contain the same terms and the same structural requirements that appear in the <title> and in the <castitle>". [215]

```
<inex_topic topic_id="256" query_type="CAS" ct_no="61">
     <InitialTopicStatement>To Find Information regarding data embedding using
     watermarking.</InitialTopicStatement>
     <title></title>
     <castitle>
          //article[about(.//p,"data embedding")] //p[about(.,watermarking)]
     </castitle>
     <description>
          We are looking for paragraphs describing watermarking in articles which describe
          data embedding.
     </description>
     <narrative>
          In today's world the issue of data security is highly significant. One such tech-
          nique to ensure data security is steganography where data is embedded in various
          media files like images, sound files, video files and so on. A commonly used data
          embedding technique is Watermarking where data can be efficiently hidden in
          a file without the changes being visible to a common person. Therefore I am
          interested in retrieving information regarding Watermarking technique for data
          embedding.Any information regarding data embedding that does not expalin wa-
          termarking is irrelevant. The purpose of this retrieval is to make a presentaion on
          watermarking.
     </narrative>
</inex_topic>
```

FIG. 2.13 – Exemple de requête CAS (Topic 256 d'INEX 2005).

```
<inex_topic topic_id="256" query_type="CAS" ct_no="61">
     <InitialTopicStatement>To Find Information regarding data
     embedding.</InitialTopicStatement>
     <title>embedding data</title>
     <castitle>//article//p[about(.,embedding data)]</castitle>
     <parent>
          //article[about(.//p,embedding data)]//p[about(.,watermarking)]
     </parent>
     <description>
          We are looking for paragraphs in articles which contain embedding data
     </description>
     <narrative>
          I am interested in retrieving information regarding techniques for embedding data.
          Any information that does not explain data embedding is irrelevant
     </narrative>
</inex_topic>
```

FIG. 2.14 – Exemple de requête CAS ayant pour parent une autre requête (Topic 272 d'INEX 2005, dépendant du Topic 256 de la figure 2.13).

2.5.4.3 Les jugements de pertinence

Les jugements concernant la pertinence des éléments renvoyés par les systèmes sont effectués par les participants eux-mêmes, à l'aide d'une interface mise à disposition en ligne [179, 137]. Les différents niveaux et degrés de pertinence, ainsi que la forme prise par l'interface d'évaluation ont changé chaque année depuis les débuts d'INEX [101, 128, 150]. Nous décrivons ici la méthodologie mise en œuvre en 2005. Deux dimensions ont été utilisées [82] :

exhaustivité

spécificité

– l'*exhaustivité**, qui décrit à quel point l'élément traite du sujet de la requête ;
– la *spécificité**, qui décrit à quel point l'élément se focalise sur le sujet de la requête.

Ces dimensions sont presque indépendantes l'une de l'autre. Par exemple, une question peut être traitée de façon satisfaisante dans un élément (exhaustivité forte) mais rester malgré tout un sujet marginal de cet élément (spécificité basse).

L'*exhaustivité* est jugée par l'humain sur une échelle à quatre niveaux [137] :

– Pas exhaustif (*not exhaustive*) : aucune correspondance entre la requête et l'élément ;
– Partiellement exhaustif (*partially exhaustive*) : l'élément ne traite que certains aspects de la requête ;
– Très exhautif (*highly exhaustive*) : l'élément traite tous ou presque tous les aspects de la requête ;
– Trop petit (*too small*) : l'élément peut être considéré comme pertinent à certains égards, mais la partie pertinente est trop petite pour être considérée.

La *spécificité* est depuis 2005 considérée comme une dimension continue représentée par une valeur réelle comprise entre 0 et 1.

En pratique, la tâche du juge humain consiste, pour une requête donnée et un document proposé, à surligner le contenu qu'il juge pertinent, puis à attribuer un niveau d'exhaustivité aux doxels contenant du texte surligné. La spécificité est alors automatiquement calculée comme un ratio entre la longueur de texte surligné dans chaque élément et la longueur totale du texte [126]. De plus certaines règles automatiques spécifient des contraintes intuitives concernant l'exhaustivité. En particulier les ancêtres d'un élément doivent avoir une exhaustivité égale ou supérieure à cet élément.

2.5.4.4 Les métriques d'évaluation

Comme nous l'avons déjà expliqué (section 2.5.3), un axe important des recherches autour d'INEX consiste en la définition de métriques d'évaluation adaptées aux spécificités des documents XML.

Valeur de la pertinence. L'ensemble des mesures proposées utilise une valeur unique combinant exhausitivité et spécificité. Cette valeur unique est calculée par des fonctions différentes selon la métrique.

Parmi ces fonctions, l'agrégation *stricte* est la plus stable :

$$f_{strict}(s,e) = \begin{cases} 1 & \text{si } e = e_{max} \text{ et } s = s_{max} \\ 0 & \text{sinon} \end{cases} \tag{2.3}$$

s et e sont les valeurs de spécificité et d'exhaustivité pour un élément, et s_{max} et e_{max} sont les valeurs attribuées à la spécificité et l'exhaustivité maximale (3 en 2004, respectivement 1 et 2 en 2005). La pertinence est ainsi convertie en une valeur binaire.

L'agrégation *généralisée* propose une échelle de valeurs entre 0 et 1. Par exemple, en 2005, où s était un réel entre 0 et 1 et e un entier égal à 0, 1 ou 2 (2 correspond à "très exhaustif"), l'agrégation généralisée était calculée par le simple produit des deux valeurs :

$$f_{gen}(s,e) = s \times e \tag{2.4}$$

Les années précédentes cette fonction était une adaptation de celle-ci à des valeurs

discontinues de s. D'autres fonctions binaires autorisant différents degrés pour e ou s ont également abouti à autant de mesures d'évaluation en 2004 [125]. Par ailleurs, *Kazai et al.* [127] proposent une fonction donnant plus de poids à la spécificité, et *Hiemstra et Mihajlović* [107] une fonction binaire $f_{liberal}$ bien plus "généreuse" que f_{strict}.

N.B. : Désormais nous appellerons un élément *pertinent* un élément qui a un score non nul avec la fonction d'agrégation utilisée.

Métriques. Nous ne nous étendons pas sur l'historique des métriques utilisées au cours des premières années d'INEX, et nous nous focalisons sur celles utilisées en 2005, à savoir le *gain cumulé étendu normalisé* (*nxCG*) et la courbe d'*effort-précision* par rapport au *gain-rappel*.

La mesure du gain cumulé étendu (*xCG*) mesure le gain, c'est-à-dire le score de pertinence, calculé par une des fonctions présentées ci-dessus, et accumulé au fur à et mesure de la progression dans la liste ordonnée d'éléments retournés.

$$xCG(i) = \sum_{j=1}^{i} xG(j) \tag{2.5}$$

où $xG(j)$ est le score obtenu pour l'élément classé en $j^{ème}$ position par le système évalué.

On compare ce gain cumulé avec celui qu'aurait dû atteindre le système s'il avait produit une liste triée optimale[1], pour obtenir le gain cumulé étendu normalisé : *nxCG*

$$nxCG(i) = \frac{xCG(i)}{xCI(i)} \tag{2.6}$$

où $xCI(i)$ est le gain cumulé idéal. On peut comparer cette mesure au rappel* pour les documents plats.

La métrique complémentaire, l'*effort-précision*, représente l'effort (en nombre de liens à visiter) qu'un utilisateur doit fournir pour parvenir à un gain donné r :

$$ep(r) = \frac{e_{ideal}}{e_{run}} \tag{2.7}$$

où e_{run} (resp. e_{ideal}) est le rang auquel le gain r est atteint par le système (resp. par la liste optimale).

Parallèlement, le gain-rappel est le rapport entre le gain obtenu en une position i de la liste et le gain total auquel on peut parvenir :

$$gr(i) = \frac{xCG(i)}{xCI(n)} \tag{2.8}$$

où n est le nombre total d'éléments pertinents. Ainsi on obtient la courbe *ep/gr* en calculant l'effort-précision à différents points de *gain-rappel* (comme pour la courbe rappel/précision, à 10 %, puis 20 %, etc.). *ep/gr*

Enfin, la mesure MAep (Mean Average Effort Precision) est la moyenne des valeurs d'effort-précision pour chaque rang auquel un nouvel élément pertinent est renvoyé par le système évalué. *MAep*

[1] C'est-à-dire une liste contenant tous les éléments pertinents au début, par ordre décroissant de pertinence. Le problème du chevauchement a conduit à d'autres définitions de cette liste optimale [127].

2.5.4.5 Les différentes tâches

La tâche *ad-hoc*. La tâche principale de la campagne INEX est la tâche *ad-hoc*. Il s'agit de trouver les éléments XML contenant les informations pertinentes quant aux besoins d'information exprimés par les titres des requêtes, et d'ordonner ces éléments dans l'ordre supposé de leur pertinence. En 2005, plusieurs aspects de cette tâche ont été explorés, dans la but d'étudier l'influence de l'utilisation de la structure (dans les requêtes) sur l'efficacité de la recherche.

En effet, outre la distinction CO+S / CAS déjà exprimée, les organisateurs ont séparé la partie CAS en quatre sous-parties, qui distinguent le caractère strict ou souple de l'interprétation des contraintes structurelles dans la cible* et le support* : SSCAS correspond à une interprétation stricte des deux éléments, SVCAS à une interprétation stricte de la cible et souple du support, et VSCAS et VVCAS représentent les deux autres combinaisons.

La tâche CO, elle, propose trois variantes : *Focussed*, dans laquelle le système doit trouver le *meilleur* élément dans un chemin donné (le chevauchement est donc interdit) ; *Thorough* autorise les chevauchements, mais bien entendu les meilleurs éléments doivent être mieux classés ; enfin, la stratégie *Fetch and Browse* consiste à renvoyer un article (un document) entier et à lister à la suite les éléments les plus pertinents de cet article.

Autres tâches. Des tâches annexes ont été ajoutées au fil des ans à la tâche *ad-hoc* pour arriver au nombre de six en 2005 :

- *Relevance Feedback*, ou comment utiliser les résultats d'un moteur pour améliorer la requête.
- *Heterogenous Collections*, ou comment permettre aux utilisateurs de faire des recherches sur plusieurs collections différentes (aux DTD différentes). Pour cela d'autres ensembles de documents ont été mis à la disposition des participants.
- *Interactive*, ou comment rendre la recherche dans les collections XML accessibles à tous les types d'utilisateurs, en termes d'interface de requête et de présentation des résultats.
- *Document Mining*, ou comment appliquer les techniques de classification et de clusterisation aux collections XML.
- *Multimedia*, ou comment utiliser les contenus non textuels pour la recherche d'information XML.
- *Natural Language Processing*, que nous détaillerons dans la section 3.7.3 (page 87).

Chapitre 3

Traitement automatique de la langue

Comme nous l'avons remarqué à de nombreuses reprises au cours de ce rapport, l'être humain est au centre du processus de recherche d'information. Parmi tous ses charmes, nous avons déjà détaillé sa versatilité et sa subjectivité, qui ruinent les espoirs de parvenir à la conception d'un système de RI "idéal"[1], et nous avons tenté de montrer qu'un système, idéal à un moment donné pour un utilisateur donné, ne le serait plus le lendemain ou pour une autre personne.

langage naturel

Une autre caractéristique embarrassante de l'humain est qu'il s'exprime, à l'écrit comme à l'oral, dans un langage dit *naturel*. On appelle ainsi le langage courant, vecteur de la communication quotidienne et notamment du présent rapport, par opposition aux langages des machines (langages de programmation, protocoles de communication, etc.). Ainsi, qu'il soit dans une disposition plate ou semi-structurée (au moins dans l'approche orientée document*), et même souvent dans des bases de données, le contenu textuel des documents est composé d'énoncés en langage naturel. Ces textes sont écrits par des humains, pour des humains, et bien entendu aucun effort particulier n'est fait pour faciliter l'analyse linguistique du texte par une machine[2].

Par ailleurs, lorsqu'un être humain cherche une information, il se pose d'abord une question en langage naturel, avec toutes les imprécisions et les subtilités que cela implique, même s'il sait qu'il devra par la suite la transcrire au moyen d'un choix de mots-clés ou d'un langage de requêtes formel et spécifique[3].

On voit donc que le domaine du langage naturel et de son traitement automatique se trouve au cœur de la problématique de la recherche d'information. Il semble évident que les progrès des futurs moteurs de recherche passeront par une meilleure "compréhension" de la langue, aussi bien en ce qui concerne les documents que la question posée.

L'état actuel de la recherche est loin de cette compréhension, et de nombreuses difficultés se présentent à tous les niveaux de l'analyse de l'écrit. Les problèmes peuvent ainsi être d'ordres morphologique*, syntaxique*, sémantique* ou pragmatique*.

Dans un premier temps, de la section 3.1 à la section 3.4, ce chapitre décrira ces différents niveaux d'étude de la langue, et abordera quelques-uns des écueils auxquels

[1] Voir la section 2.5, page 34.

[2] Une indexation manuelle du document est parfois fournie, une structure (tables ou balisages) peut être mise en place, mais il ne s'agit pas là d'une aide à la "compréhension" sémantique ou syntaxique du texte.

[3] Nous ne nous attarderons pas sur deux autres points essentiels, qui sont d'une part que les humains ne s'expriment pas tous dans la même langue, et d'autre part que la majorité d'entre eux font des fautes d'orthographe et de grammaire en écrivant.

se heurte particulièrement toute tentative d'analyse automatique de textes.

Nous développerons par la suite les aspects du traitement automatique de la langue qui ont été explorés dans les systèmes de recherche d'information (section 3.5), puis nous aborderons le domaine des interfaces en langage naturel pour les bases de données (section 3.6). Enfin, en guise de transition vers la seconde partie, nous donnerons un aperçu des enjeux et des premières approches concernant les interfaces en langage naturel pour la recherche d'information semi-structurée (section 3.7).

3.1 Les différents niveaux du langage

L'analyse du langage nécessite une connaissance de sa structure sur de nombreux niveaux : que sont les mots ? Que signifient-ils ? Comment se combinent-ils pour former la phrase ? Comment contribuent-ils au sens de la phrase ? Et, par ailleurs, comment fonctionnent le monde et le raisonnement de l'humain dans le monde ?

De la réception des sons (ou leur prononciation) jusqu'à la compréhension approfondie des mots prononcés dans l'environnement où ils sont prononcés, les linguistes distinguent plusieurs palliers permettant l'analyse ou la génération d'un énoncé en langage naturel. Ces niveaux de connaissance restent bien entendu toujours valables lorsque l'on aborde l'analyse *automatique* de la langue. Les grandes spécialités sont :

- la **phonétique** et la **phonologie**, ou comment les mots et les phrases sont liés aux sons qui les réalisent à l'oral [38, 123]. Ne traitant que l'écrit, nous ne reviendrons pas sur ce domaine.

morphologie

- la **morphologie**, ou comment les mots sont construits et quels sont leurs rôles dans la phrase [157].

syntaxe

- la **syntaxe***, ou comment les mots se combinent pour former des *syntagmes**, puis des *propositions* et enfin des *phrases* correctes.

sémantique

- la **sémantique***, ou comment les mots font du sens lorsqu'ils sont insérés dans une phrase (indépendamment du contexte) [111].

pragmatique

- la **pragmatique***, ou comment les phrases peuvent être interprétées selon leur contexte d'énonciation (interlocuteurs, phrases précédentes, connaissance commune du monde, ...) [142, 58].

Allen [9] donne les exemples suivants pour faire la distinction entre syntaxe, sémantique et pragmatique. Considérons que les phrases suivantes sont candidates pour figurer en tête du présent mémoire, c'est-à-dire qu'elles sont énoncées en l'absence totale de contexte :

(3.1) La recherche d'information automatisée permet à un utilisateur de formuler un besoin d'information.

(3.2) Les grenouilles vertes ont des gros nez.

(3.3) Les idées vertes ont des gros nez.

(3.4) Vertes des ont les idées nez gros.

La première phrase semble être un début raisonnable pour un tel rapport. Elle correspond à tout ce qui est connu en matière de syntaxe, de sémantique et de pragmatique. La phrase 3.2 est bien formée sur les plans syntaxique et sémantique, mais pas pragmatique. En effet, elle conviendrait mal comme première phrase d'un mémoire de thèse, et le lecteur ne verrait aucune raison valable de la voir utilisée.

Mais l'exemple 3.3 serait pire encore : il est à la fois pragmatiquement et sémantiquement mal formé. On remarque en effet qu'il est possible d'affirmer que la phrase 3.2

est vraie ou fausse, tandis que c'est impossible pour 3.3 dans une conversation cohérente. La structure en est pourtant correcte, mais des idées ne peuvent pas être vertes et, même si elles le peuvent dans certains contextes, elles n'ont certainement pas de nez, ni gros ni maigre[1].

Enfin, la phrase 3.4 est tout simplement inintelligible : elle contient pourtant les mêmes mots que la précédente, mais ne respecte aucune des structures grammaticales (syntaxiques) admises en Français.

Dans le cadre de notre travail, nous nous basons uniquement sur la langue écrite[2], ce qui implique que les entités les plus petites que nous allons étudier (les *unités grammaticales*) sont les mots. Cela ne signifie pourtant pas que la structure interne du mot, en particulier les différentes flexions*, n'est pas prise en compte. Mais le mot comporte l'avantage d'être une unité relativement facile à distinguer dans un texte (en tout cas dans les langues française et anglaise, qui vont nous intéresser plus particulièrement), ce qui n'est pas le cas pour la langue orale.

3.2 Morphologie

> Mot *: Son ou groupe de sons articulés ou figurés graphiquement, constituant une unité porteuse de signification à laquelle est liée, dans une langue donnée, une représentation d'un être, d'un objet, d'un concept, etc.*
> Trésor de la Langue Française.

La morphologie [155] est l'étude de la forme des mots (de leur flexion – indications de cas, genre, nombre, mode, temps, etc. – de leur dérivation – préfixes, suffixes, infixes – et de leur composition – mots composés). Sous l'appellation de morphosyntaxe, elle représente également l'étude des règles de combinaison des morphèmes (unités minimales de sens) selon la configuration syntaxique de l'énoncé [171].

En pratique, dans le cadre du traitement automatique de la langue, l'analyse morphologique consiste à segmenter le texte en unités élémentaires (les mots, c'est la *tokenisation*) et à déterminer les différentes caractéristiques de ces unités, et en particulier les catégories grammaticales.

3.2.1 Catégories grammaticales

catégorie grammaticale

L'information morphosyntaxique la plus importante est la catégorie grammaticale, qui regroupe sous une étiquette commune les mots partageant le même comportement syntaxique dans un énoncé. Le *nom*, le *verbe*, l'*adjectif* sont des catégories grammaticales. Le nombre de catégories n'est pas fixé, il est défini dans chaque application par un compromis entre complexité et spécificité[3]. La liste de catégories la plus utilisée (en Anglais) est la *Penn Treebank* [152] et compte 45 classes[4]. Celle du *Brown Corpus* [74] en répertorie 87, d'autres dépassent la centaine. Le lecteur qui aurait oublié le rôle des prépositions, adverbes, articles et autres conjonctions peut consulter l'annexe B.1.

[1] Cet exemple est une référence à la célèbre phrase *"Colorless green ideas sleep furiously"* de Noam Chomsky, et donne une bonne idée de la distinction entre syntaxe et sémantique.

[2] L'analyse d'une question formulée à l'oral est pourtant aussi un aspect intéressant, notamment pour des raisons d'accessibilité.

[3] Des catégories très spécifiques permettent une analyse syntaxique plus précise, mais sont plus difficiles à attribuer sans ambiguïté. Ainsi il serait intéressant de savoir si un verbe est transitif* ou non, mais cette information nécessite des connaissances lexicales importantes.

[4] La liste complète est disponible en annexe B.1.2.

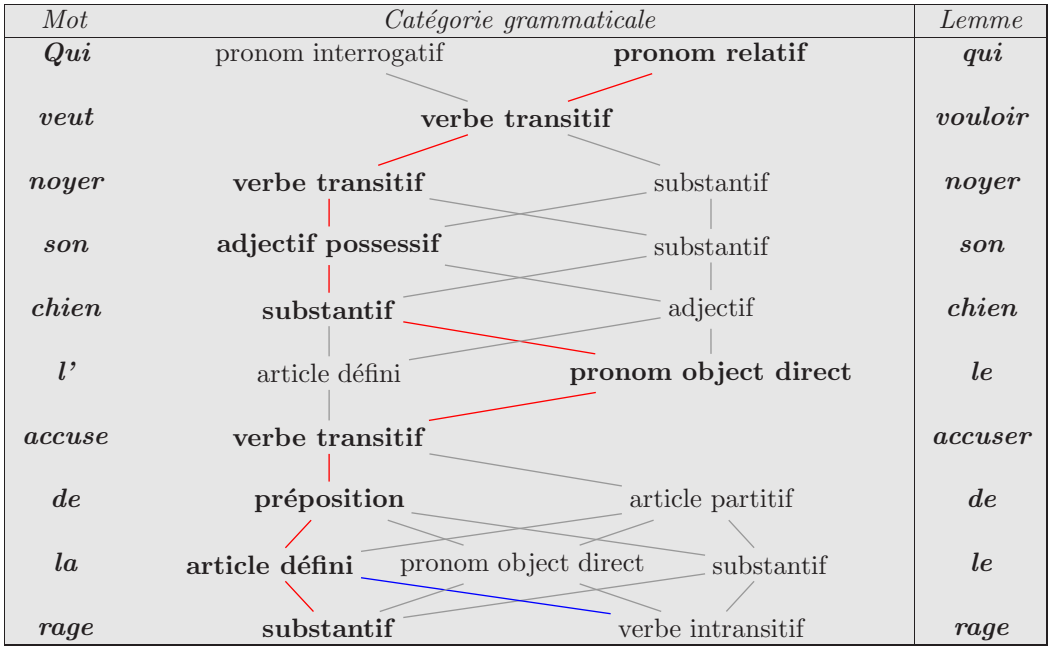


FIG. 3.1 – Catégories grammaticales possibles pour chacun des mots de la phrase *"Qui veut noyer son chien l'accuse de la rage"*. Les catégories correctes sont indiquées en gras. Les flexions* (voir plus bas) ne sont pas prises en compte (par exemple, *"accuse"* peut être au subjonctif ou à l'indicatif). Les diverses possibilités et le nom des catégories sont issus du *Trésor de la Langue Française*. A droite se trouvent les lemmes* dont dérivent les formes de la phrase.

Le problème qui se pose est que plusieurs catégories peuvent très souvent convenir à une même forme. En Français, les formes ambiguës sont estimées à environ 25 % du lexique, voire plus pour les mots les plus courants [63]. Nous illustrons ce problème à la figure 3.1 en montrant toutes les combinaisons de catégories possibles pour une simple phrase.

étiquetage morpho-syntaxique

L'étiquetage automatique des mots par leurs catégories est un problème bien connu et assez bien maîtrisé pour l'écrit des langues les plus étudiées. Il consiste à gérer les ambiguïtés décrites ci-dessus et à attribuer (au besoin avec un taux de probabilité) une seule catégorie à chaque mot du texte. Les techniques employées pour parvenir à ce but sont diverses, mais ont toutes pour principe commun l'utilisation du contexte de chaque mot, c'est-à-dire l'élimination des combinaisons impossibles ou improbables. Ainsi, pour les trois derniers mots de l'exemple de la figure 3.1, il est hautement improbable que la combinaison "préposition + article + verbe" soit la bonne, car elle ne correspond à aucun schéma de la langue.

Les méthodes récentes annoncent des précisions supérieures à 95 % pour les langues européennes. Des détails sur les différents algorithmes peuvent être trouvés dans différents supports [175, chap. 5][119, chap. 8][151, chap. 10].

3.2.2 Les autres informations morphosyntaxiques

La plupart des étiqueteurs automatiques fournissent comme seul renseignement la catégorie grammaticale. Il existe pourtant d'autres informations relatives à la morphologie.

lemme

Lemme. Le lemme est la racine d'un mot, dépouillée des marques d'accord, de conjugaison, de cas. C'est la forme qui se trouve en général en entrée du dictionnaire : en Français, le verbe à l'infinitif, le substantif au masculin singulier. Toute analyse de type sémantique nécessite la connaissance de ce lemme.

flexion

lemmatisation

Flexions. Les flexions sont les modifications opérées sur le lemme pour distinguer les formes de conjugaison (personne, temps, mode, voix – flexion verbale) ou le genre, le nombre et le cas (flexion nominale)[1]. Ces aspects sont illustrés à la figure 3.2. L'opération qui consiste à retrouver ces informations est la *lemmatisation*[2]. En Anglais, les flexions sont peu nombreuses, et il est aisé de les retrouver à partir du lemme et de la catégorie grammaticale. En Français, la tâche est plus ardue. Le logiciel *Flemm* [169] permet de retrouver les flexions d'un texte écrit en Français à partir du résultat des analyseurs morphosyntaxiques Brill ou TreeTagger.

veut	*rage*
cat : verbe	cat : nom
type : transitif	lemme : rage
lemme : vouloir	genre : féminin
mode : indicatif	nombre : singulier
temps : présent	
personne : 3s	
voix : active	

FIG. 3.2 – Description morphosyntaxique d'un verbe et d'un nom (voir figure 3.1).

3.3 Syntaxe

Un enfant parle très bien sa langue maternelle et pourtant il ne saurait en écrire la grammaire. [...] Le grammairien est celui qui sait pourquoi et comment l'enfant connaît la langue.
Umberto Eco, *Apostille au Nom de la Rose.*

A l'issue d'une analyse morphosyntaxique, les formes initiales présentes dans un énoncé sont remplacées par une liste ordonnée d'éléments contenant un certain nombre d'informations, parmi lesquelles la catégorie grammaticale et, éventuellement, le lemme, les flexions ou d'autres connaissances dont la présence dépend de l'application souhaitée.

syntaxe

La syntaxe décrit comment ces éléments s'ordonnent pour créer des *constituants*, composant aux-mêmes des *phrases*. De ce fait, on représente souvent le résultat d'une analyse syntaxique de façon hiérarchique.

3.3.1 Du mot à la phrase

3.3.1.1 Constituants

syntagme

tête

En linguistique structurale, si l'on examine la construction d'une phrase du bas (les mots) vers le haut (la phrase), les étapes intermédiaires constituent la formation de *constituants*, ou *syntagmes*. Ces syntagmes sont qualifiés par le type de l'élément principal (la *tête*) : syntagmes nominaux, verbaux, adjectivaux, prépositionnels, ad-

[1] Les langues utilisant les flexions sont dites *flexionnelles*.

[2] Ce terme de lemmatisation est souvent employé pour signifier la racinisation (ou *stemming*) que nous décrivons à la section 3.5.1. Il est pourtant important de distinguer les deux. En particulier, la

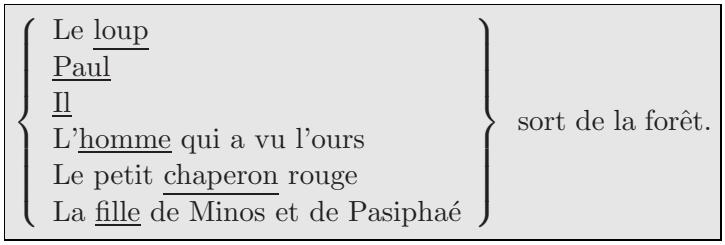


FIG. 3.3 – Substituabilité du syntagme (ici, un syntagme nominal, avec à sa tête l'élément souligné – un nom ou un pronom qui remplace lui-même un nom). Notons que cette caractéristique est vraie pour des genres, nombres et cas identiques. Le sujet de la phrase ne pourrait pas être remplacé par *"les loups"* ou *"lui"*, qui sont pourtant également des syntagmes nominaux.

FIG. 3.4 – Imbrication de syntagmes, avec *NP* = Noun Phrase (syntagme nominal) ; *PP* = Prepositional Phrase (groupe prépositionel).

verbiaux. Les éléments autres que la tête sont des *spécifieurs* (déterminants...), des *qualificateurs* (adjectifs, adverbes...), des *compléments* (compléments du nom, propositions relatives...).

Il est important de remarquer que ce qui forme un constituant est dépendant du contexte [193], comme l'illustre l'exemple suivant, dans lequel *"François et Ségolène"* forme tantôt un syntagme à part entière, tantôt des éléments séparés :

(3.5) [*François et Ségolène* forment un ménage moderne].

(3.6) [Les enfants sont gardés par *François*] *et* [*Ségolène* part en voyage].

Ainsi rechercher des types de syntagmes à un niveau purement local sans se préoccuper des autres types de construction possibles peut conduire à des erreurs.

Par ailleurs, les syntagmes de même type (nominal, verbal, etc.) ont la même fonction dans la phrase que leur tête, et ils sont *syntaxiquement* substituables entre eux (comme le montre la figure 3.3). Les syntagmes peuvent s'imbriquer les uns dans les autres (figure 3.4[1]), jusqu'à former des *propositions*, unité supérieure constituée d'un sujet, d'un verbe et d'éventuels compléments.

3.3.1.2 Phrases

Au niveau syntaxique, la phrase est l'unité prépondérante de l'analyse. Elle peut être constituée d'une seule proposition mais admet également la coordination (3.8) et la subordination (3.9) des propositions :

racinisation n'aboutit pas obligatoirement à des formes existant dans le lexique.

[1]Les abréviations des constituants que nous donnons ici et que nous utiliserons tout au long du mémoire sont les abréviations anglaises, universellement utilisées. Par exemple : *VP* pour *Verbal Phrase* (syntagme verbal), *S* pour *Sentence* (phrase), etc. (voir l'annexe B.2.5).

(3.7) [Deux pigeons s'aimaient d'amour tendre].[1]

(3.8) [Deux sûretés valent mieux qu'une], et [le trop en cela ne fut jamais perdu].[1]

(3.9) [Vous savez que [nul n'est prophète en son pays]].[1]

Une succession de phrases forment le *discours*.

3.3.1.3 Les fonctions grammaticales

fonction grammaticale

La fonction grammaticale est le rôle *syntaxique* qu'occupe un composant par rapport à un autre constituant. Elle dépend surtout de la position physique des constituants. Par exemple, sauf exception (voix passive, interrogation, etc.), le *sujet* d'un verbe est, en Français comme en Anglais, placé avant lui. Les autres fonctions essentielles sont l'objet et l'objet indirect, le complément du nom... Les relations basées sur une préposition (*"dans"*, phrase 3.13) peuvent prendre le nom de celle-ci (en l'absence d'apport sémantique). Notons également que pour la phrase 3.12, le sujet réel a pris la place du sujet grammatical (*"le chaperon"*) pour tenir compte de la voie passive.

(3.10) [Le loup]$_{sujet}$ [lorgne] [le petit chaperon rouge]$_{objet}$.

(3.11) [Le petit chaperon rouge]$_{sujet}$ [dit] [à sa grand-mère] [qu'elle a de grandes mains]$_{objet}$.
obj. indirect

(3.12) [Le chaperon]$_{objet}$ [n'est pas alerté]$_{sujet}$ [par [les grandes dents] [de sa grand-mère]]$_{complément\ du\ nom}$.

(3.13) [Le chaperon]$_{sujet}$ [finira] [***dans*** le ventre du loup]$_{dans}$!

Il ne faut pas confondre les relations grammaticales avec les relations sémantiques (*agent*, *bénéficiaire*, *lieu*) qui seront abordées plus loin[2]. Ces deux types de relations sont liés, mais il n'existe pas de moyen unique de passer de l'une à l'autre.

3.3.2 Les ambiguïtés syntaxiques

One morning I shot an elephant in my pajamas. How he got into my pajamas I don't know.
Groucho Marx, *Animal Crackers*.

"Parfaitement, dit le Major. Ma grand-mère, qui est morte, y avait un appartement et mon père l'a conservé."
Le Bison n'entendant pas d'e muet à la fin comprit qu'il s'agissait de l'appartement et non de la grand-mère.
Boris Vian, *Les Remparts du Sud*.

[1] *Jean de la Fontaine*.
[2] Section 3.4.2.2 page 66.

Les ambiguïtés concernant la syntaxe d'un énoncé peuvent avoir de nombreuses causes. Celles-ci ont toutes pour résultat une déduction erronée de l'agencement des mots dans la phrase. On peut classer les ambiguïtés syntaxiques selon les connaissances nécessaires pour les résoudre :

Connaissances pragmatiques du monde qui nous entoure ou, dans les exemples suivants, du contexte (nous avons besoin de connaître les occupations récentes de Jean, qui peut venir du Marché aux Puces ou de Pekin) :

(3.14) a. Jean [a rapporté^{Verbe}] [un vase^{NP}] [de Chine^{PP}]

$NP \rightarrow NP\ PP$

b. Jean [a rapporté^{Verbe}] [un vase^{NP}] [de Chine^{PP}]

$VP \rightarrow VP\ NP$

Connaissances sémantiques ajoutées par exemple au lexique (ici, une tarte a des ingrédients et les pommes sont comestibles, ce qui n'est pas le cas des clients, auxquels on vend des choses) :

(3.15) a. Jean [vend^{Verbe}] [une tarte^{NP}] [aux pommes^{PP}]

$NP \rightarrow NP\ PP$

b. Jean [vend^{Verbe}] [une tarte^{NP}] [aux clients^{PP}]

$VP \rightarrow VP\ PP$

Connaissances syntaxiques comme par exemple les accords en genre et en nombre :

(3.16) a. un [jus^{Nom}] d'[oranges^{Nom}] [fraîches^{Adj}]

Nom + Adj

b. un [jus^{Nom}] d'[oranges^{Nom}] [frais^{Adj}]

Nom + Adj

(3.17) a. des [tags^{Nom}] dans les [banlieues^{Nom}] [karchérisés^{Adj}]

Nom + Adj

b. des [tags^{Nom}] dans les [banlieues^{Nom}] [karchérisées^{Adj}]

Nom + Adj

L'exemple suivant rend peut-être encore mieux compte de l'ampleur du problème : le groupe *"à la rhubarbe"*, qui s'attache bien entendu à *"la tarte"*, peut être lié au verbe *"manger"* pour deux raisons différentes : en tant que complément d'object indirect (comme dans *"il a donné la tarte à sa belle-mère"*), si l'on ignore que *"manger"* n'a pas d'emploi transitif indirect, ou en tant qu'adverbial (comme dans *"il a mangé la tarte à la maison"*).

(3.18) Il [a mangé] [la tarte] [à la rhubarbe]

complément objet indirect

adverbial

En l'absence des connaissances pragmatiques, sémantiques et/ou syntaxiques nécessaires (et aucun système ne les possède toutes), une analyse pourra produire plusieurs résultats pour un ensemble donné de règles.

Les cas les plus fréquents d'ambiguïtés syntaxiques sont liés au rattachement des groupes prépositionnels et des propositions relatives. Ils peuvent très vite mener à des centaines de constructions possibles, même avec des phrases relativement courtes.

A titre d'expérience, nous avons soumis la phrase suivante : *"We are searching sections dealing with version management in articles containing a section about object databases."*[1] à deux analyseurs gratuits disponibles en ligne, LinGO ERG [53][2] et Link Grammar [216][3], parvenant à respectivement 159 et 543 analyses possibles.

3.3.3 L'analyse syntaxique

De nombreux formalismes ont été proposés pour l'analyse syntaxique ; ceux que nous présentons ici sont ceux qui ont une importance dans le travail décrit dans la seconde partie du mémoire.

3.3.3.1 L'analyse superficielle

analyse superficielle

L'analyse superficielle (ou partielle) [4] a pour but de reconnaître les syntagmes simples, non récursifs, d'un énoncé, sans les lier les uns aux autres. Ainsi, les principaux attachements responsables des ambiguïtés citées ci-dessus, comme les attachements prépositionnels, ne sont pas traités (du moins dans un premier temps). L'idée est d'obtenir des résultats certes moins riches, mais plus rapides et plus sûrs. Elle est appropriée pour un certain nombre d'applications qui ne cherchent pas à définir des dépendances précises, comme par exemple la reconnaissance de syntagmes nominaux [30]. Cette approche s'oppose à l'analyse profonde (ou complète), qui cherche à regrouper chaque phrase dans une unique représentation.

analyse profonde

Voici un exemple d'analyse partielle, dans laquelle on regroupe les syntagmes nominaux, verbaux et prépositionnels non récursifs. On peut également reconnaître une clause C, non ambiguë, mais dans laquelle le PP n'est pas inclu.

(3.19) [[Les gendarmes ^{NP}] [interpellent ^{VP}] [un conducteur ^{NP}] ^{C}] [en état d'ivresse ^{PP}]

Cass

Un exemple d'analyseur superficiel efficace est Cass [5], qui consiste en une cascade d'automates à états finis. Chacun de ces automates représente un niveau de reconnaissance. Au niveau le plus bas, le niveau 0, l'entrée est une suite de mots avec leur catégorie grammaticale (sortie d'un analyseur morphosyntaxique comme TreeTagger). L'automate de niveau 1 trouve toutes les séquences du niveau 0 correspondant à un certain motif – par exemple, des syntagmes nominaux – et les réduit à des éléments simples portant un nouveau nom – par exemple, `np`. La sortie de cet automate devient l'entrée de l'automate suivant, et ainsi de suite.

[1] "Nous cherchons des sections traitant de la gestion des versions dans un article contenant une section sur les bases de données objet." Cette phrase est inspirée d'une requête d'INEX 2004 (Topic. 130), avec uniquement des modifications de vocabulaire, car certains mots n'étaient pas reconnus par les analyseurs utilisés.

[2] http://www.delph-in.net/erg/

[3] http://hyper.link.cs.cmu.edu/link/index.html

```
:niveau0
    verbe      →  aux past_part
    np         →  art nom
:niveau1
    pp         →  prep np
    vp         →  verbe np
:niveau2
    prop_rel   →  pro_rel vp
```

FIG. 3.5 – Cascade de règles d'automates à états finis.

niveau 0

```
[np
 ├[art le]
 └[nom dessin]]
[d' prep]
[np
 ├[art un]
 └[nom boa]]
[qui pro_rel]
[verbe
 ├[aux a]
 └[part_passé avalé]]
[np
 ├[art un]
 └[nom éléphant]]
```

niveau 1

```
[np
 ├[art le]
 └[nom dessin]]
[pp
 └[d' prep]
   [np
    ├[art un]
    └[nom boa]]]
[qui pro_rel]
[vp
 └[verbe
    ├[aux a]
    └[part_passé avalé]]
   [np
    ├[art un]
    └[nom éléphant]]]
```

niveau 2

```
[np
 ├[art le]
 └[nom dessin]]
[pp
 └[d' prep]
   [np
    ├[art un]
    └[nom boa]]]
[prop_rel
 └[qui pro_rel]
   [vp
    └[verbe
       ├[aux a]
       └[part_passé
         avalé]]
      [np
       ├[art un]
       └[nom éléphant]]]
```

FIG. 3.6 – Analyse superficielle en cascade.

La figure 3.5 propose une cascade de trois ensembles de règles simples, et la figure 3.6 illustre l'application de ces règles à la phrase suivante :

(3.20) Le dessin d'un boa qui a avalé un éléphant.

Notons que les éléments utilisés dans les règles, s'ils participent à la création d'un nouvel élément, ne sont plus accessibles au niveau suivant (par exemple, une référence à un `nom` au niveau 1 serait inopérante). Remarquons également qu'un seul résultat est fourni, les ambiguïtés n'étant pas prises en compte (le premier résultat trouvé, l'analyse est interrompue).

3.3.3.2 Les grammaires hors-contexte

grammaire hors-contexte

Une grammaire hors-contexte (ou CFG, pour *context-free grammar*) se compose d'un ensemble de règles de la forme :

$$E \rightarrow E_1 \ldots E_n$$

qui expriment le fait que la séquence d'expressions $\{E_1 \ldots E_n\}$ peut être remplacée (*réécrite*) par un nouvel identifiant *unique* E, en faisant abstraction des éléments qui

Règles		*Exemples*
S	→ *NP VP*	[Le loup]$_{NP}$ [sort de la forêt]$_{VP}$
NP	→ *Pronom*	il
NP	→ *Nom_Propre*	Paul
NP	→ *Det Adj? Nom Adj?*	[Le]$_{DET}$ [petit]$_{ADJ}$ [chaperon]$_{NOM}$ [rouge]$_{ADJ}$
NP	→ *NP PP*	[La fille]$_{NP}$ [de Minos et de Pasiphaé]$_{PP}$
VP	→ *Verbe PP*	[sort]$_{Verbe}$ [de la forêt]$_{PP}$
VP	→ *Verbe NP*	[mange]$_{Verbe}$ [le chat]$_{NP}$
PP	→ *Prep NP*	[de]$_{Prep}$ [la forêt]$_{NP}$
...	→ ...	...

FIG. 3.7 – Règles hors-contexte permettant d'obtenir les constructions vues à la figure 3.3. Avec *S* : *sentence* (phrase) ; *NP* : *noun phrase* (syntagme nominal) ; *VP* : *verbal phrase* (syntagme verbal) ; *PP* : *prepositional phrase* (syntagme prépositionnel) ; *Det* : déterminant ; *Adj* : adjectif ; *Prep* : préposition. Le point d'interrogation '*?*' signifie que l'élément est optionnel.

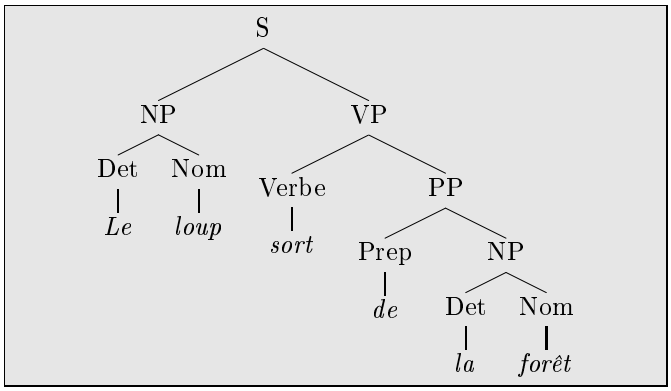


FIG. 3.8 – Arbre syntaxique de la phrase *"Le loup sort de la forêt"*, représentant l'application de certaines règles proposées à la figure 3.7.

l'entourent. L'application de ces règles, illustrées à la figure 3.7 par des regroupements de *constituants* linguistiques, est souvent représentée de façon arborescente, comme le montre la figure 3.8.

Les CFG permettent de refléter les aspects hiérarchiques des constructions grammaticales par l'imbrication des constituants et la *récursivité* des règles.

Comme nous le verrons dans les sections suivantes, de telles grammaires provoquent vite une combinatoire élevée. Par ailleurs, utilisées telles quelles, elles ne fournissent que peu d'informations sur la structure de la phrase, puisqu'elles ne donnent pas la nature des relations qui lient les constituants les uns aux autres.

Plusieurs stratégies d'analyse existent pour traiter des grammaires hors-contexte, de façon descendante ou ascendante d'une part, en profondeur ou en largeur d'autre part. La description, les avantages et les inconvénients de chaque possibilité sont décrits par exemple par *Gardent et Baschung* [85, chap. 2].

3.3.3.3 L'analyse en dépendance

L'analyse syntaxique en dépendance diffère surtout de l'analyse en constituants (comme les règles hors-contexte) par le mode de représentation. Les deux approches

analyse en dépendance

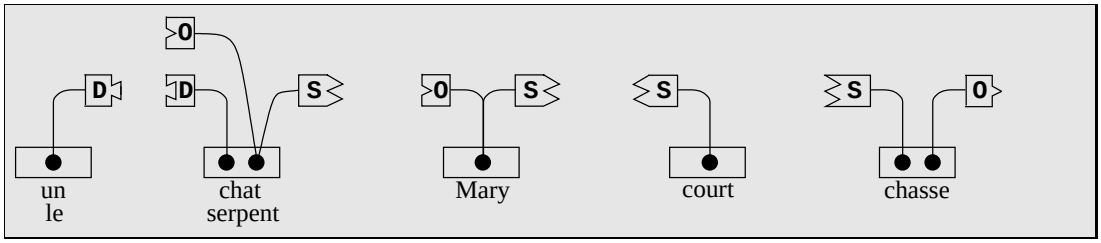


FIG. 3.9 – Définition du lexique de la *Link Grammar*.

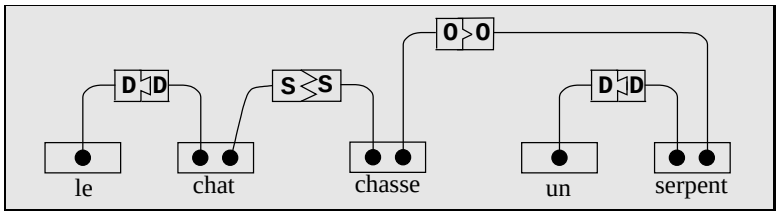


FIG. 3.10 – Analyse syntaxique avec la *Link Grammar*.

n'ont pas de différence au niveau de leur couverture ou de leur expressivité.

L'idée est de relier des mots, et non plus des constituants, en définissant pour chaque catégorie une liste (avec contraintes) d'autres catégories susceptibles de lui être attachées. Puis l'analyseur se charge de mettre en place les liens.

Un exemple de grammaire de dépendance est la *Link Grammar* [216]. Son fonctionnement est rapidement illustré aux figures 3.9 (définition du lexique et des contraintes d'attachement) et 3.10 (application). Notons que les liens ne doivent pas se croiser.

3.3.3.4 Les traits

Les grammaires décrites jusqu'à présent ne suffisent absolument pas à capturer de façon utile le langage naturel. La première raison est qu'il existe de nombreuses restrictions lors du regroupement des mots en constituants. Par exemple, *"*le fille"* ou *"*les loup"* ne sont pas corrects car les accords en genre et en nombre ne sont pas respectés. La seconde raison est que l'on ne souhaite pas seulement reconnaître des constructions, mais aussi en extraire des informations linguistiques, permettant d'effectuer une analyse adaptée à l'application mise en œuvre.

traits

Ces deux problèmes peuvent trouver leur réponse avec l'ajout de *traits*, des paires d'attributs/valeurs attachées aux entrées lexicales et/ou aux constituants formés durant l'analyse. Par exemple, ajouter à certaines entrées du lexique (déterminants, noms, verbes) des traits "genre" et "nombre", prenant respectivement les valeurs *masculin* ou *féminin* d'une part, *singulier* ou *pluriel* d'autre part, permet de proposer, la règle hors-contexte augmentée suivante :

$$NP \rightarrow \quad Det\ Nom$$
si *Det* et *Nom* s'accordent en genre et en nombre

soit

$$\underset{\left[\begin{array}{l}\text{genre : G}\\ \text{nombre : N}\end{array}\right]}{NP} \rightarrow \underset{\left[\begin{array}{l}\text{genre : G}\\ \text{nombre : N}\end{array}\right]}{Det} \quad \underset{\left[\begin{array}{l}\text{genre : G}\\ \text{nombre : N}\end{array}\right]}{Nom}$$

Ainsi, seuls les syntagmes nominaux accordés sont acceptés par la grammaire, et

```
s --> np, vp.                    nom --> [forêt].
np --> det, nom.                 verbe --> [sort].
vp --> verbe, pp.                prep --> [de].
pp --> prep, np.                 det --> [le].
nom --> [loup].                  det --> [la].
```

FIG. 3.11 – Exemple de règles de grammaire à clauses définies.

l'information du genre et du nombre est propagée pour la suite de l'analyse.

Un autre exemple intéressant est la *forme* et la *sous-catégorisation* des verbes. Ainsi l'information qu'un verbe est à l'impératif permettra d'empêcher qu'on lui attache un sujet. La connaissance de la valence d'un verbe (transitivité* ou pas) évitera certaines ambiguïtés[1].

Les traits peuvent être utilisés à tous les niveaux (phonologique, morphologique, syntaxique, sémantique, pragmatique), et il est possible d'ajouter toutes sortes de traits nécessaires à une application donnée.

3.3.3.5 Les grammaires à clauses définies

grammaire à clauses définies

Les grammaires à clauses définies (ou DCG, pour *Definite Clause Grammars*) sont des programmes écrits dans le langage Prolog[2] qui implémentent directement les grammaires hors-contexte, de la façon simple indiquée par le *code* de la figure 3.11. La stratégie "imposée" par le fonctionnement de Prolog est l'analyse descendante en profondeur, de gauche à droite.

Les DCG ont l'avantage de permettre l'extension des grammaires, par l'ajout d'informations syntaxiques, sémantiques ou autres, véhiculées par chaque constituant. Il est ainsi possible de dépasser le stade de la reconnaissance simple de constructions et, en utilisant les traits* jugés nécessaires, de transmettre tout type d'indications pendant l'analyse. Celles-ci peuvent servir à édifier une interprétation sémantique, des relations grammaticales ou toute autre représentation utile, mais aussi à opérer une analyse plus fine des énoncés.

Ainsi, on peut modéliser les contraintes de genre et de nombre, comme le fait la nouvelle grammaire de la figure 3.12, qui permet de gérer certaines classes d'ambiguïtés syntaxiques décrites à la section 3.3.2 (exemples 3.16 et 3.17, page 54). Le caractère `'_'` indique que la variable n'est pas définie ou pas utilisée. Ce nouvel ensemble de règles a deux avantages :

- Il rejette les syntagmes incorrects comme *"*une jus"*, car le déterminant et le nom ne partagent pas le même nombre.
- Il attache *sans ambiguïté* les adjectifs *"fraîches"* aux *"oranges"* dans *"un jus d'oranges fraîches"*, et *"frais"* au *"jus"* dans *"un jus d'oranges frais"*. En effet, pour le premier NP, le syntagme *"un jus d'oranges"*, dont la tête* est *"jus"*, est au masculin singulier, tandis que *"fraîches"* est au féminin pluriel. L'attachement de *"fraîches"* à *"jus"* est donc impossible. Le second NP est traité de la même façon.

[1] Par exemple, *"clients"* peut être attaché à *"vend"* dans *"Marie vend une tarte aux clients"*, car *"vendre"* est transitif indirect*, mais *"pommes"* sera sans ambiguïté attaché à *"tarte"* dans *"Marie cuisine une tarte aux pommes"*, car *"cuisine"* n'admet pas d'objet indirect.

[2] Prolog est un langage de programmation logique dont un aperçu rapide et clair est donné par *Blackburn et Bos* [27, annexe D]. Les bases essentielles à la compréhension seront précisées au fur et à mesure du texte.

```
np(Nombre, Genre) --> nom(Nombre, Genre), adj(Nombre, Genre).
np(Nombre, Genre) --> nom(Nombre, Genre), prep, np(_, _).
np(Nombre, Genre) --> det(Nombre, Genre), nom(Nombre, Genre).
np(Nombre, Genre) --> nom(Nombre, Genre).
nom(sing, masc) --> [jus].
nom(plur, fem) --> [oranges].
adj(sing, masc) --> [frais].
adj(plur, fem) --> [fraîches].
det(sing, masc) --> [un].
det(sing, fem) --> [une].
prep --> [d'].
```

FIG. 3.12 – Exemple de règles étendues de grammaire à clauses définies.

```
s(Sem) --> np(NP), vp(VP), {append(NP, VP, Sem)}.
np([Nom]) --> det, nom(Nom).
vp([Verbe | PP]) --> verbe(Verbe), pp(PP).
pp(NP) --> prep, np(NP).
nom(animal:loup) --> [loup].
nom(lieu:forêt) --> [forêt].
verbe(mouvement:sortir) --> [sort].
prep --> [de].
det --> [le].
det --> [la].
```

FIG. 3.13 – Exemple de règles étendues de grammaire à clauses définies.

Un autre exemple permet de faire quelques pas dans la sémantique, avec l'enrichissement de la figure 3.11 par des prédicats contenant des informations lexicales sémantiques, comme l'illustre la figure 3.13. Précisons qu'en Prolog, les variables commencent par une majuscule, les constantes et les prédicats par une minuscule. Les crochets `'['` et `']'` représentent une liste, et `[E | Rest]` est une liste commençant par l'élément `E` et continuant par les éléments de la liste `Rest`. Enfin, le prédicat `append(A, B, C)` permet de concaténer la liste `A` et la liste `B` pour obtenir la liste `C`. Dans la syntaxe des grammaires à clauses définies, les instructions Prolog spécifiques sont insérées entre accolades (voir le prédicat `s`).

L'application de ces règles est illustrée à la figure 3.14.

3.3.3.6 Les formalismes plus évolués

De nombreux formalismes de grammaire bien plus évolués, mais aussi bien plus complexes, que ceux que nous avons évoqués ici, ont été développés durant les dernières décennies. Les grammaires *à traits* comme GPSG (Generalized Phrase Structure Grammar [1, chap. 2]) ou LFG (Lexical Functional Grammar [1, chap. 1]) ou les grammaires d'*unification* comme FUG (Functional Unification Grammar [124]) ou surtout HPSG (Head-driven Phrase Structure Grammar [1, chap. 3][26]), sont des théories linguistiques à part entière, articulant souvent le lexique, la syntaxe et la sémantique dans une même représentation.

Les applications personnelles de l'analyse syntaxique que nous proposons de décrire dans la suite de ce mémoire ne nécessitent pas la mise en œuvre de telles grammaires, et ce d'autant moins que leur complexité pose des problèmes de robustesse. Nous ne nous étendrons donc pas sur ce sujet.

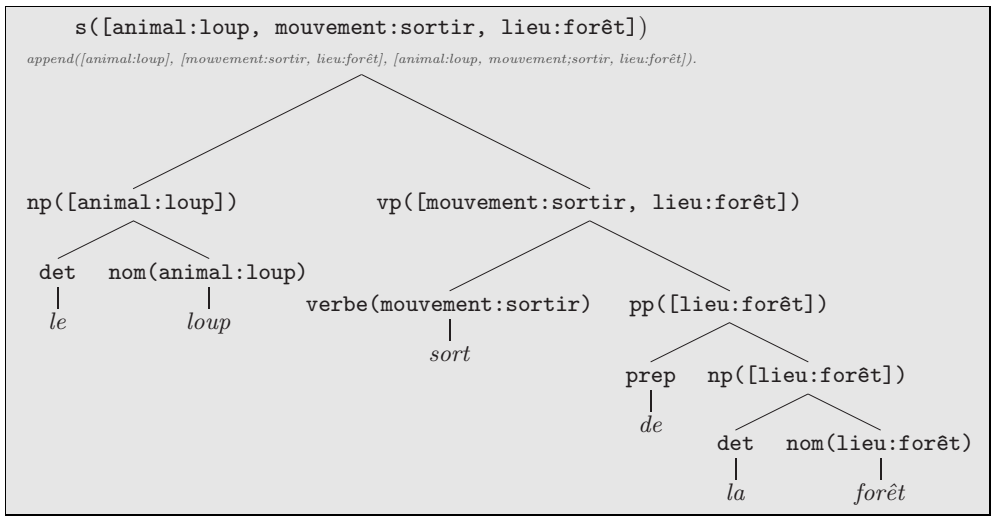


FIG. 3.14 – Propagation des informations par la DCG de la figure 3.13.

3.4 Sémantique

Les applications des analyses sémantiques sont diverses et plus ou moins ambitieuses. Un apport sémantique peut servir à réduire les ambiguïtés syntaxiques, à mieux cibler des concepts (par exemple en recherche d'information), mais sa finalité globale est de représenter formellement[1] l'information véhiculée par un énoncé et éventuellement d'en inférer de nouvelles connaissances ou une réponse à la question posée (si l'énoncé est une question).

Nous abordons tout d'abord la sémantique *grammaticale*, qui s'attache à construire le sens de l'énoncé global, puis à la sémantique *lexicale*, qui étudie la contribution des mots à ce sens et toutes les ambiguïtés qu'ils provoquent.

3.4.1 La représentation du sens

Nous décrivons ici comment il est possible de symboliser le sens d'un énoncé par une représentation *logique*, à l'aide de prédicats.

D'autres formalismes ont été proposés (voir de courts exemples à la figure 3.15), que nous ne traiterons pas, notamment les *réseaux sémantiques* ②, les *dépendances conceptuelles* ③ et les représentations par cadres (*frames* ④).

3.4.1.1 La représentation logique

La formule logique (exemple ① de la figure 3.15), grâce à un langage formel connu, possédant une syntaxe simple et dénuée d'ambiguïté, offre la possibilité :

– de représenter un énoncé :

(3.21) Je ne te hais point.
$$\exists x \exists y \; locuteur(x) \wedge auditeur(y) \wedge \neg haïr(x, y)$$

– d'attribuer des *valeurs de vérité* binaires à une expression donnée :

[1] Les mots sont importants, il s'agit bien entendu d'une *représentation* et non pas d'une *compréhension*, et son caractère *formel* s'oppose par essence au fait que les langues ne le sont pas, ou qu'au moins les auteurs ne les traitent pas de façon formelle [66].

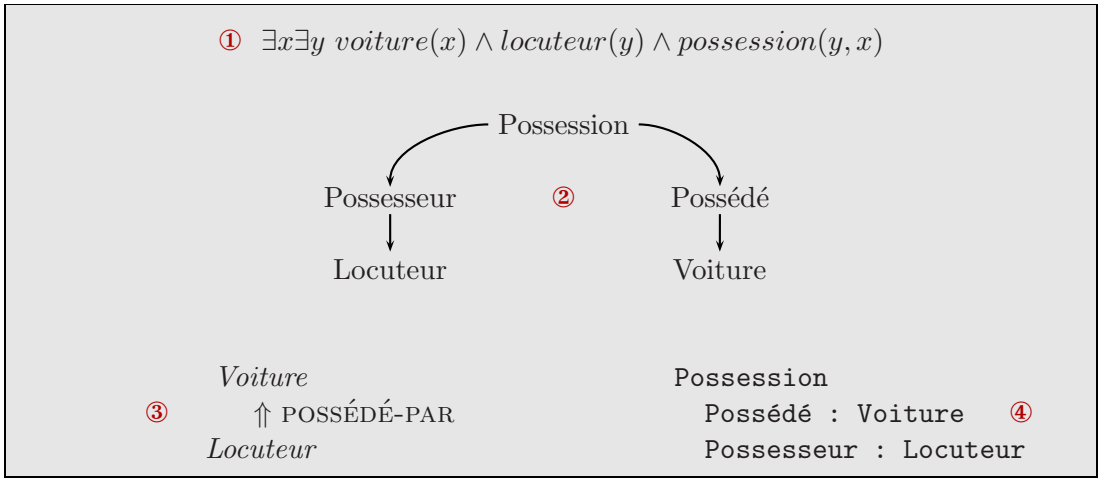


FIG. 3.15 – Différentes représentations sémantiques de la phrase *"J'ai une voiture"* (d'après *Jurafsky et Martin* [119, chap. 14]).

(3.22) $empereur(Napoléon)$ est vrai ssi Napoléon est empereur.

– de raisonner sur des connaissances et des énoncés :

(3.23) si $\forall x(homme(x) \Rightarrow mortel(x))$
et $homme(Socrate)$
alors $mortel(Socrate)$

Mais la logique des prédicats du premier ordre est loin de pouvoir traiter tous les phénomènes des énoncés en langage naturel, dont le sens ne se limite pas à leur seule valeur de vérité. En particulier :

– La logique gère sans état d'âme certaines subtilités du langage :

(3.24) Je suis en Espagne. $\rightsquigarrow \exists x\ locuteur(x) \wedge est_situé(x, Espagne)$

(3.25) C'est en Espagne que je suis. $\rightsquigarrow \exists x\ locuteur(x) \wedge est_situé(x, Espagne)$

– Seules les *propositions* auxquelles on peut attribuer aisément une valeur de vérité sont représentables. Il est par exemple difficile de formaliser la modalité (*"Il est possible/probable/nécessaire que..."*) ou les souhaits (*"J'espère que vous viendrez"*).

– Cette formalisation ne distingue pas la valeur linguistique de la valeur logique, et ne rejette pas les énoncés vrais mais absurbes (*Les Pyramides sont en Egypte et cet immeuble a 5 étages.*[1]).

Ces différents phénomènes n'ayant que peu de répercussions sur le travail que nous décrivons dans la seconde partie de ce mémoire, nous ne nous attarderons pas sur les moyens de les prendre en compte.

3.4.1.2 Un exemple de représentation, la DRT

DRT

DRS

référent

Dans la Théorie de Représentation du Discours (ou DRT pour *Discourse Representation Theory* [121]), le sens des énoncés est représenté par des Structures de Représentation du Discours (DRS). Une DRS est une paire composée d'un ensemble de *référents de discours* et d'un ensemble de *contraintes*, ou de *conditions*, sur ces réfé-

[1] Même si la frontière entre une phrase absurde et une phrase qui fait du sens est mince et dépend beaucoup du contexte (*"J'habite à Saint-Etienne et je n'aime pas le football"*).

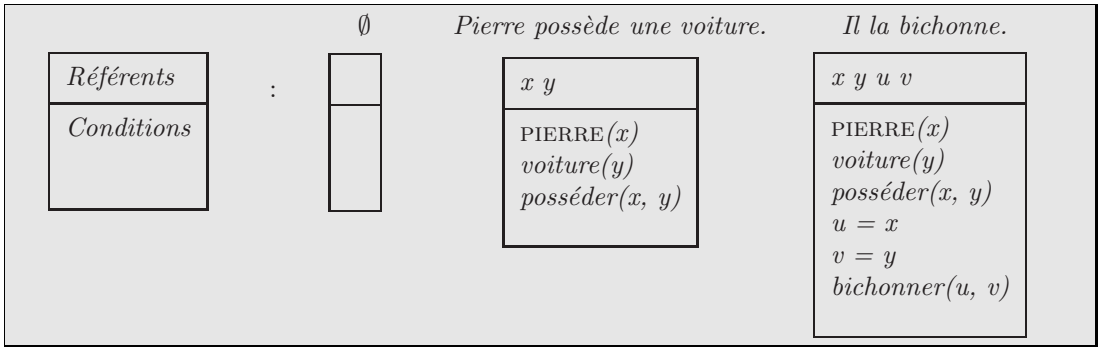


FIG. 3.16 – Exemple de structure de représentation du discours (DRS).

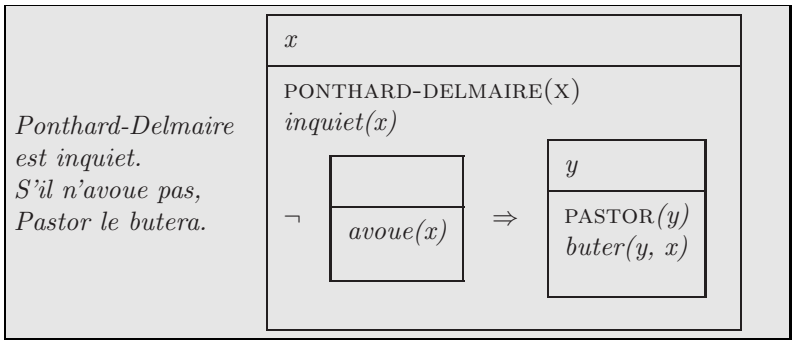


FIG. 3.17 – Exemple d'imbrication de DRS.

rents. Cette structure est généralement présentée à l'aide de "boîtes" à deux étages. Une DRS s'enrichit à chaque nouvel énoncé, comme le montre la figure 3.16.

Les *conditions* de la DRS peuvent contenir des prédicats, mais aussi d'autres DRS, éventuellement liées par des opérateurs $\neg$, $\vee$, $\Rightarrow$, comme en logique classique (voir figure 3.17). En revanche, les quantificateurs ne sont pas nécessaires, ils sont exprimés par les "boîtes".

Définie ainsi, une telle structure est équivalente à des expressions logiques traditionnelles. Pourtant, la DRT permet de gérer de façon élégante la négation, la portée des quantificateurs et les phénomènes de références comme les anaphores* ou les ellipses* (voir la section 3.4.3).

Chaque terme du lexique peut être associé à une représentation DRS de base, selon sa catégorie grammaticale. Ceci permet d'adapter la DRT à de nombreuses stratégies d'analyse sémantique (voir plus loin – section 3.4.2) [27, vol. II, chap. 2].

3.4.1.3 Les événements

L'utilisation de prédicats uniques pour représenter les événements (et plus généralement les verbes) et les éléments qui y sont rattachés pose un certain nombre de problèmes théoriques et pratiques pour la détermination des rôles thématiques. Dans notre exemple 3.21, *"haïr"* étant un verbe transitif, un prédicat binaire (agent + thème) a été utilisé. Cette approche trouve très vite ses limites. Prenons l'exemple du verbe transitif *"manger"*, qui peut être utilisé dans les conditions suivantes [119, chap. 14] :

rôle thématique

(3.26) a. J'ai mangé. *(emploi intransitif)*

b. J'ai mangé un sandwich.

c. J'ai mangé [un sandwich] [dans mon bureau].

d. J'ai mangé [un sandwich] [avec les doigts].

e. J'ai mangé [un sandwich] [dans mon bureau] [pour le dîner].

f. ...

Utiliser un prédicat $manger(x, y)$ ne suffit bien sûr pas. Une autre possibilité serait d'utiliser un prédicat contenant l'ensemble des rôles possibles ($manger(v, w, x, y, z)$ pour le sujet, le thème (l'objet), le lieu, la manière, le moment), mais c'est à la fois trop riche (les variables resteront la plupart du temps non instanciées) et trop limitant (on peut manger *"avec les doigts et avec dégoût"* ou *"pour le dîner à 20h30"*, etc.).

réification

La solution la plus réaliste est de représenter les événements comme des objets, au même titre que les noms, et de multiplier les prédicats. C'est le principe de la *réification*. L'action de manger introduit donc la clause $\exists m\ manger(m)$ ou $\exists m\ evt(m, manger)$, puis chaque nouveau syntagme se rattachant au verbe provoque l'insertion d'un nouveau prédicat :

(3.27) J'ai mangé [un sandwich] [dans mon bureau] [pour le dîner].
$\exists m\ \exists s\ \exists b\ \exists d\ manger(m) \wedge sandwich(s) \wedge bureau(b) \wedge dîner(d)$
$\wedge\ agent(m, \text{SPEAKER}) \wedge thème(m, s) \wedge lieu(m, b) \wedge temps(m, d)$

3.4.2 L'analyse sémantique

L'analyse sémantique a pour but d'associer à une séquence de mots une représentation interne de son sens. Le formalisme choisi, le niveau de détails, la caractère complet ou partiel de l'analyse, les constructions reconnues, dépendent de l'utilisation que l'on souhaite faire du résultat et des connaissances actuelles en la matière.

Nous présentons ici quatre méthodes différentes d'analyse sémantique : l'analyse profonde, dont le but est d'obtenir une représentation logique complète de l'énoncé ; l'interprétation sémantique des relations grammaticales, qui s'appuie, comme la première, sur une analyse syntaxique totale ; les grammaires sémantiques, qui modélisent des domaines très spécifiques ; et les patrons sémantiques, qui détectent des informations prédéfinies dans un texte.

3.4.2.1 L'analyse profonde par compositionnalité

De nombreuses méthodes d'analyse s'inspirent fortement du *principe de compositionnalité* [114] et de la grammaire de Montague [173, 122], qui permettent d'associer l'élaboration du sens d'une phrase aux règles syntaxiques qui la régissent. Ainsi le sens d'un constituant est la composition des sens des éléments qu'il contient. Cette progression de bas en haut est illustrée par la figure 3.18 [1].

On voit que cette façon de procéder semble assez puissante, mais pose un certain nombre de difficultés pratiques.

La première réside dans la nécessité de faire correspondre les variables introduites par certaines catégories (dans notre exemple, x et y, par les articles) à des positions dans les prédicats générés par d'autres éléments (principalement les noms et les verbes – *manger(x, y)* ou *manger(y, x)* ?).

[1] Une illustration de la construction d'une DRS est également donnée pour l'exemple à la figure 3.19.

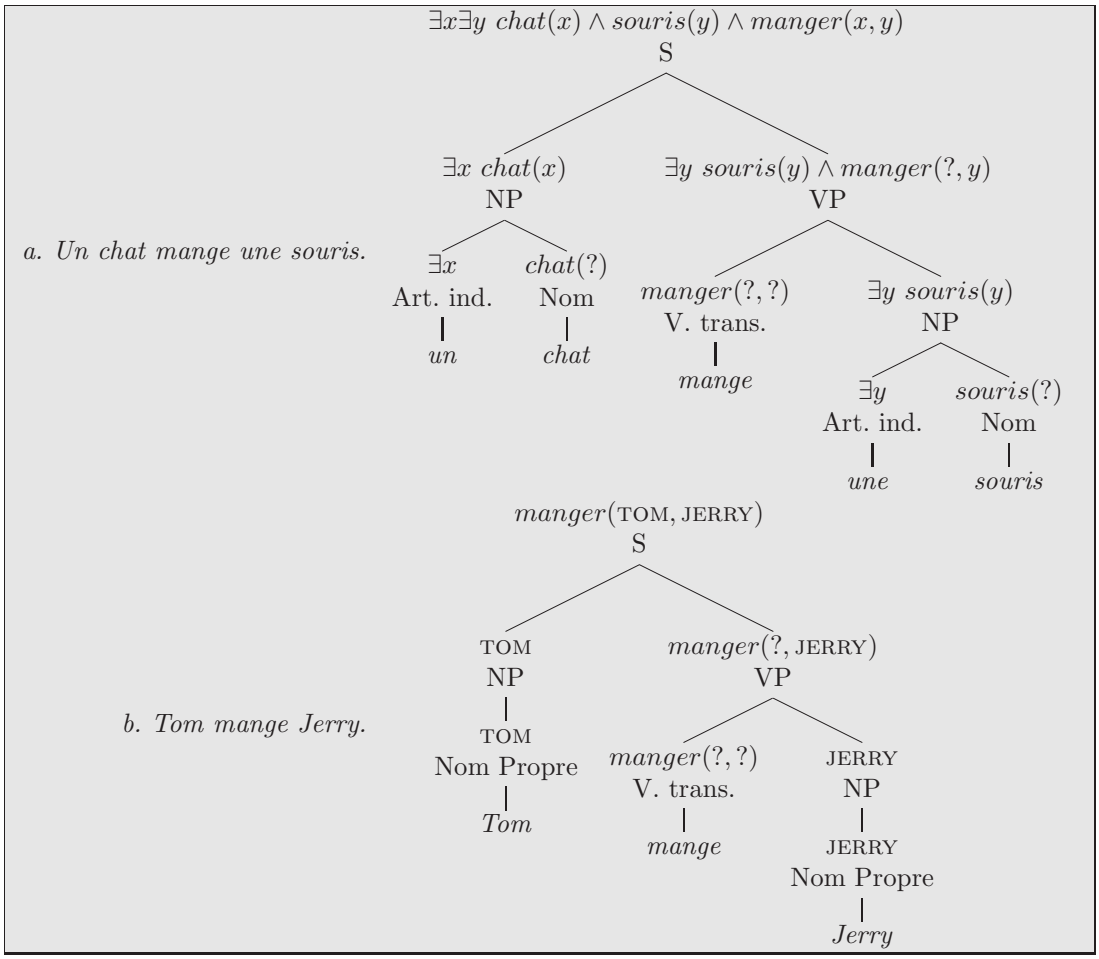


FIG. 3.18 – Composition de la représentation sémantique parallèlement à l'analyse syntaxique.

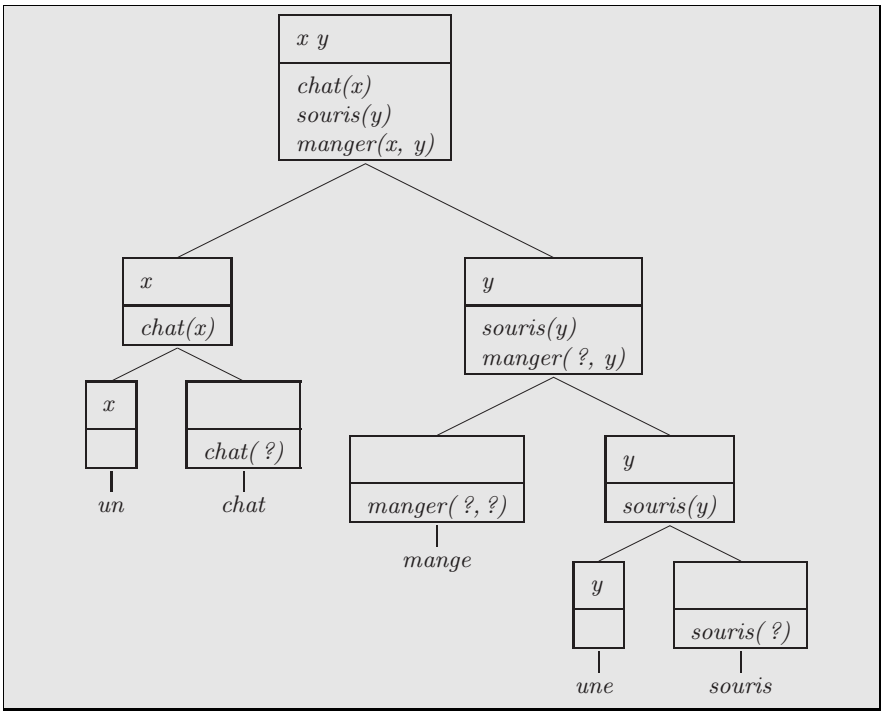


FIG. 3.19 – Construction d'une structure de représentation du discours (DRS).

De plus, une même règle syntaxique ne doit donner lieu qu'à un seul type de comportement sémantique, sous peine de devoir multipler les règles redondantes [27, chap. 2]. On voit par exemple que les deux phrases *"Un chat mange une souris"* et *"Tom mange Jerry"* sont sémantiquement assez proches[1]. Pourtant les représentations finales et la façon d'y parvenir présentent des différences notables. Dans le cas des noms propres (3.18.b), il suffit d'insérer la représentation de *"Jerry"* en seconde position du prédicat *manger* pour obtenir le syntagme verbal (VP). Reprendre cette méthode en 3.18.a conduirait à la formule regrettable $manger(?, \exists y\ souris(y))$. Il est cependant impensable d'associer des actions sémantiques différentes selon la "provenance" du constituant à chaque niveau syntaxique. On perdrait tout l'intérêt du regroupement en constituants substituables entre eux.

Enfin, les représentations intermédiaires (VP, NP...) sont *incomplètes*, c'est-à-dire que certaines variables sont encore à instancier (point d'interrogation dans les figures). Un autre enjeu est donc de parvenir à marquer et gérer l'information manquante.

L'introduction du lambda-calcul (λ-calcul) fournit un formalisme capable de répondre à ces problèmes. Ce formalisme est élégant et puissant, ses bases théoriques sont fortes et séduisantes pour les chercheurs intéressés par la sémantique formelle. Mais pour la conception de systèmes opérationnels dont la finalité n'est pas la représentation formelle des énoncés, sa mise en œuvre est complexe et peu robuste face à des entrées grammaticalement imparfaites. L'annexe B.3 fournit quelques détails concernant cette approche. Les paragraphes suivants abordent d'autres types de techniques d'interprétation sémantique orientées vers des domaines et/ou des applications particulières.

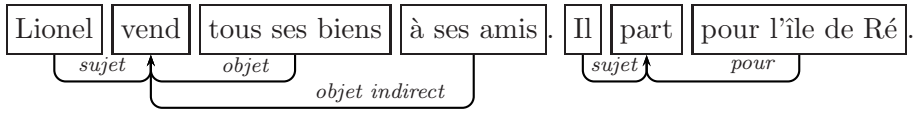
3.4.2.2 L'interprétation sémantique des relations grammaticales

relations grammaticales

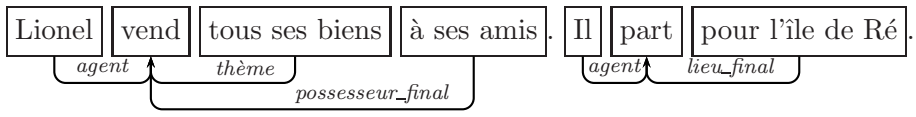
rôle thématique

Les relations grammaticales[2] s'opposent aux relations sémantiques par le fait qu'elles n'attribuent pas de rôles thématiques aux divers éléments, mais se contentent de fournir les liens syntaxiques. Les exemples suivants illustrent les relations grammaticales (3.28.a) et les rôles thématiques (3.28.b) liant les constituants d'un même discours.

(3.28) a.



b.



L'obtention des rôles thématiques dans le cas général nécessite des connaissances lexicales évoluées (ici, la vente est une action qui conduit à un changement de possesseur d'un objet et à un changement inverse de possesseur d'argent ; le verbe *"partir"* implique un passage d'un lieu à un autre). L'idée est donc de mettre en place un certain nombre de règles spécifiques à l'application voulue, transformant certains rôles syntaxiques précis en rôles sémantiques, et permettant d'obtenir les relations souhaitées tout en laissant de côté les constructions sans intérêt.

Le principe de conversion est relativement simple. Ainsi, les quelques règles de la figure 3.20 permettent de traduire la représentation a en introduisant les rôles de

[1] Même si quelques connaissances pragmatiques supplémentaires permettent de conclure que la seconde est impossible !

[2] Voir la section 3.3.1.3, page 53.

sujet (verbe_vente)	→	**agent**
objet (verbe_vente)	→	**thème**
objet_indirect (verbe_vente)	→	**possesseur_final**
sujet (verbe_mouvement)	→	**agent**
pour (verbe_mouvement)	→	**lieu_final**

FIG. 3.20 – Quelques règles d'interprétation des relations grammaticales.

REQUÊTE_VOL	→	VOL
REQUÊTE_PRIX	→	"Combien coûte" VOL " ?"
REQUÊTE_HEURE_DÉPART	→	"A quelle heure" VERBE_DÉPART VOL " ?"
VOL	→	("le" | "un") "vol" PROVENANCE DESTINATION
VOL	→	VILLE ("-" | "/") VILLE
DESTINATION	→	("à" | "vers" | "pour") VILLE
VERBE_DÉPART	→	("partir" | "décoller")
PROVENANCE	→	("à partir")? "de" VILLE
VILLE	→	"Saint-Etienne" | "Toulouse" | "Nice" | "Brisbane" | "Bujumbura" | "Tokyo" ...

FIG. 3.21 – Exemple de grammaire sémantique.

l'exemple b. Mais les règles de correspondance peuvent être bien plus complexes si nécessaire.

Utiliser le résultat d'une analyse syntaxique permet de s'affranchir des aléas de la construction des phrases et d'utiliser des structures déjà formalisées et moins ambiguës ; on peut alors convertir les relations grammaticales en relations sémantiques, de la façon souhaitée, avec le niveau de finesse souhaité.

3.4.2.3 Les grammaires sémantiques

grammaires sémantiques

Lorsque les besoins sémantiques de l'application que l'on met en œuvre sont limités et clairement définis, les *grammaires sémantiques* de *Burton* permettent de définir des règles très spécifiques, en mettant de côté des constructions moins utiles [29].

Les délimitations utilisées par l'analyseur ne sont plus celles des grammaires classiques (*syntaxiques*), comme les catégories grammaticales ou les syntagmes divers, mais des classes conçues autour d'un domaine particulier et d'une tâche précise.

Ainsi l'exemple classique de la base de données d'une compagnie aérienne est typique des besoins que peut couvrir une grammaire sémantique. La figure 3.21 propose une grammaire simple permettant d'analyser (entre autres) les requêtes suivantes :

(3.29) a. Combien coûte un vol de Saint-Etienne à Nice ? (→ REQUÊTE_PRIX)

b. A quelle heure décolle le Tokyo-Toulouse ? (→ REQUÊTE_HEURE_DÉPART)

c. Saint-Etienne/Toulouse (→ REQUÊTE_VOL)

Les mots *"Saint-Etienne"*, *"Toulouse"*, etc. ne sont pas des noms propres mais de noms de VILLE, la construction *"pour Saint-Etienne"* n'est plus un syntagme prépositionnel, mais une DESTINATION.

De cette façon, ces grammaires combinent les aspects syntaxiques et sémantiques, ainsi que les particularités de l'application, dans un même cadre. Elles permettent de

se focaliser sur les informations nécessaires, et de produire une représentation *ad hoc* de l'énoncé selon le but recherché par la suite, ce qui rend généralement inutile tout autre traitement linguistique.

3.4.2.4 Les patrons sémantiques

template matching

La technique des *patrons sémantiques* (ou correspondance de régions, ou surtout *template matching*) utilise généralement une analyse syntaxique de surface* pour reconnaître les groupes nominaux et verbaux de base (sans compléments), avant de détecter de simples séquences de termes prédéfinies, permettant de remplir les champs de cadres eux aussi prédéfinis. Cette approche est particulièrement appropriée aux applications d'extraction d'informations concernant un domaine délimité à l'avance. La figure 3.22 donne un exemple complet d'interprétation sémantique par ce moyen, avec la phrase suivante :

(3.30) La Finlande a battu le Canada 2 à 0 lors d'un match du premier tour du tournoi masculin de hockey sur glace.

Il est important de remarquer que cette approche de permet pas de traiter des constructions non répertoriées, mais qu'elle est robuste malgré cela. Ainsi on peut imaginer qu'un service de messagerie en ligne, proposant de la publicité ciblée, pourra analyser les messages de Lionel indiquant qu'il part pour l'île de Ré (voir l'exemple 3.28) et, à l'aide de quelques patrons suggérés par son client (agence de voyage), détecter l'information intéressante au milieu d'un message traitant de divers autres sujets, et proposer à l'intéressé des billets pour l'île de Ré.

3.4.3 Les ambiguïtés sémantiques

Si le lait cru ne convient pas à votre bébé,
faites-le bouillir.
Otto Jespersen [116].[1]

Il y a une pile de matières inflammables près de votre
voiture. Il faudra vous en débarasser.
Georges Burns And Gracie Allen, série Américaine.[2]

Les ambiguïtés sémantiques au niveau grammatical[3] sont en général des problèmes d'attribution de relations implicites (qui ne sont indiquées par des mots particuliers) entre constituants.

Chacun de ces problèmes est l'objet de nombreuses recherches, s'étendant souvent sur le plan pragmatique. Nous citons ici quelques-uns des plus fréquents.

3.4.3.1 L'anaphore

L'anaphore consiste à reprendre un segment du discours antérieur par un autre élément grammatical, pour éviter la répétition. Ces références sont très difficiles à repérer (à *résoudre*) automatiquement. Les cas les plus fréquents sont les *anaphores pronominales* dans lesquelles un pronom est utilisé pour remplacer un groupe nominal. Leur

[1] "If the baby does not thrive on raw milk, boil it.", cité par Jerry Hobbs [109].
[2] "There's a pile of inflammable trash next to your car. You'll have to get rid of it.", cité par Jerry Hobbs [109].
[3] Les ambiguïtés au niveau lexical sont traités à la section 3.4.4.

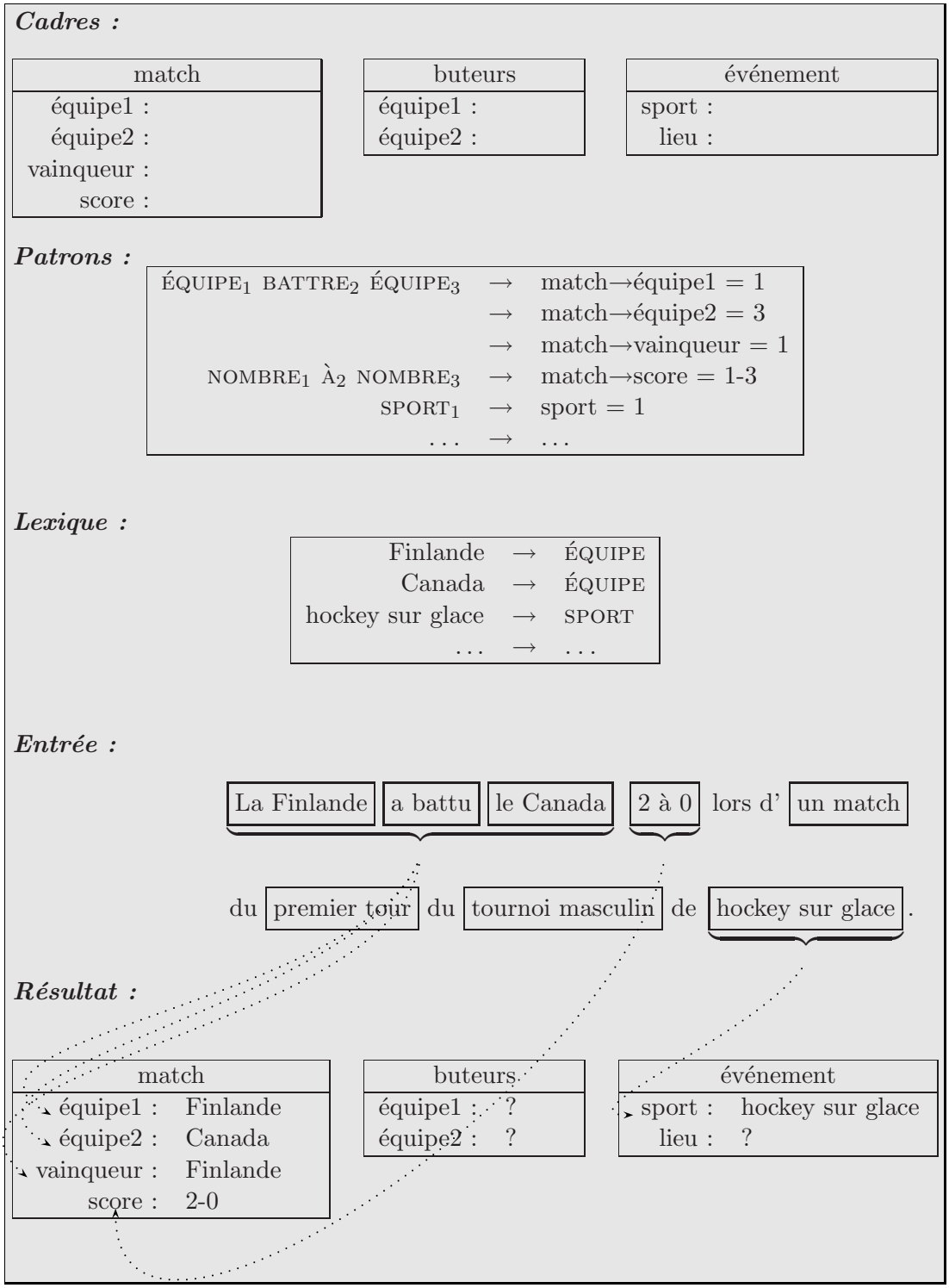


FIG. 3.22 – Patrons sémantiques (*template matching*).

résolution peut être simple quand il n'existe pas d'ambiguïté (exemple 3.31), mais se complique très vite (3.32, 3.33) jusqu'à entrer dans le domaine de la pragmatique (3.34).

anaphore

(3.31) Jacques$_1$ était furieux. Il$_1$ s'était disputé avec Georges.

(3.32) Dominique$_1$ rencontra Collin$_2$ à un congrès. Ils$_{1+2}$ se réconcilièrent.

(3.33) Nicolas$_1$ rencontra Dominique$_2$ dans un couloir. Il$_?$ lui$_?$ en voulait toujours.

(3.34) a. Pierre$_1$ empoisonna Sam$_2$. Il$_2$ mourut.
b. Pierre$_1$ empoisonna Sam$_2$. Il$_1$ fut arrêté.

Outre les anaphores pronominales, de nombreuses combinaisons de références sont possibles, comme l'utilisation d'adjectifs possessifs (exemple 3.35) ou de constituants entiers (3.36), mais aussi la référence à d'autres éléments que des groupes nominaux (3.37).

(3.35) La cage$_1$ du gorille$_2$ s'ouvrit. Sa$_1$ serrure devait être mal fermée.

(3.36) Le gorille$_1$ accélèra le pas vers le juge$_2$. Le quadrumane$_1$ avait une idée derrière la tête.

(3.37) Je ne peux donner la suite de l'histoire$_1$. Cela$_1$ serait pourtant délectable.

résolution des pronoms

Les procédés de résolution automatique des pronoms recourent souvent à de la connaissance du monde [213]. Les deux familles de méthodes purement sémantiques (sans connaissance extra-linguistique) les plus populaires en ce qui concerne les références à des groupes nominaux (cas le plus simple) sont l'algorithme naïf proposée par *Hobbs* [109] et la méthode des focus de *Sidner* [213][1]. Le premier suggère que le meilleur constituant candidat lors de l'occurrence d'un pronom est le dernier NP de la phrase courante ou, à défaut, le premier NP des phrases précédentes. La seconde distingue le

focus

focus *acteur*, l'élément jouant un rôle actif à un moment donné du discours, et le focus du *discours*, l'élément le plus important à un moment donné du discours. Le premier est généralement le dernier constituant ayant eu le rôle d'*agent* dans la phrase, tandis que le second est choisi parmi les éléments non agents. Un pronom en position d'agent se réfèrera au focus acteur, un pronom dans une autre position sera lié au focus du discours.

Ces deux approches sont illustrées à la figure 3.23 avec trois phrases pour lesquelles elles donnent des résultats différents, notamment l'exemple proposé par *Hobbs* [109] :

(3.38) The castle in Camelot remained the residence of the king until 536 when he moved it to London.[2]

3.4.3.2 L'ellipse

Un autre moyen de faire référence à un élément introduit au préalable en évitant la répétition est de pratiquer l'ellipse. Il s'agit d'omettre de cet élément dans une construction qui aurait dû le contenir. On peut appliquer l'ellipse à toutes sortes de constituants :

(3.39) a. Les Stéphanois portent des écharpes vertes et les Toulousains préfèrent les rouges et noires.

b. Les Stéphanois aiment le football et les Toulousains le rugby.

c. Les Stéphanois détestent les Parisiens et les Toulousains aussi.

Si ces exemples sont des formes absolument correctes de la langue française, les ellipses sont surtout utilisées dans le dialogue de façon plus informelle, pour revenir sur des thèmes déjà abordés :

[1] D'autres théories permettent de réduire les candidats possibles par des restrictions sémantiques simples, sans conclure.

[2] "Le chateau de Camelot demeura la résidence du roi jusqu'à 536, lorsqu'il la déplaça à Londres." Nous avons conservé la version anglaise car la traduction n'offre aucune ambiguïté grâce à la flexion féminime du pronom *"la"*, qu'on ne retrouve pas dans l'Anglais *"it"*.

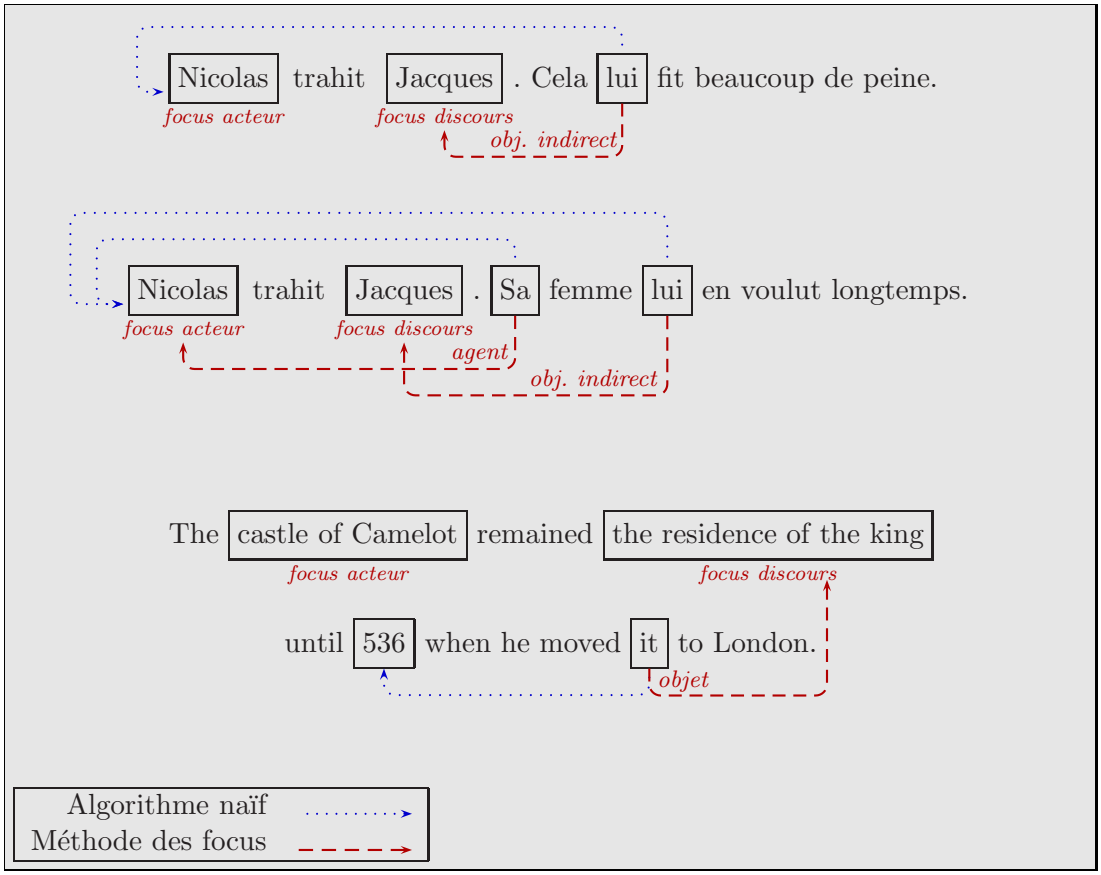


FIG. 3.23 – Résolution des pronoms, algorithme naïf et méthode des focus.

(3.40) – Je préfère aller au cinéma, et toi ? – Au théâtre.

Les concepteurs de systèmes de dialogue pour l'interrogation des bases de données ont tout particulièrement travaillé sur ce type de constructions, pour permettre des dialogues du type [106] :

(3.41) – Quelle est la taille du [navire] Santa Inez ?
– 127 mètres.
– Du navire nucléaire le plus rapide ?

Deux approches pour résoudre les ellipses sont utilisées. La première considère que la syntaxe de la phrase contenant une ellipse correspond (phénomène de l'ellipse mis à part) à celle de la phrase précédente [9, chap. 14]. Le "trou" provoqué par l'ellipse trouve ainsi sa correspondance. L'autre approche est celle de la proximité sémantique (avec la méthode des grammaires sémantiques vues à la section 3.4.2.3). Ainsi dans l'exemple précédent, les deux suites de mots *"du Santa Inez"* et *"Du navire nucléaire le plus rapide"* sont regroupées dans la même catégorie (BATEAU). On en déduira que le reste de la première question (*"Quelle est la taille du"*) doit être ajouté à la seconde [106].

3.4.3.3 La portée des quantificateurs

Ce problème concerne l'ambiguïté parfois provoquée par l'introduction de quantificateurs (*tous, chaque, un*) dont la portée est incertaine, parfois même pour l'humain

qui lit la phrase. Ainsi, dans les exemples suivants, un seul et unique chien suit-il tous les hommes, ou chaque homme est-il accompagné d'un chien différent ? Y a-t-il un ou plusieurs problèmes d'environnement préoccupants ? Existe-t-il un seul piano soulevé par tous les hommes, ou soulèvent-t-ils chacun leur piano ?

(3.42) a. Un chien suit chaque homme qui passe la porte du bar.

b. Un problème d'environnement préoccupe tous les politiciens sérieux.

c. Tous les hommes soulevèrent un piano.

Encore une fois, cette problématique est particulièrement importante pour les interfaces d'interrogation de bases de données en langage naturel. Les solutions proposées se situent au niveau sémantique, nous ne les détaillerons pas ici.

3.4.4 La sémantique lexicale

Ici la mer est bleue et les nuits sont blanches
Publicité pour le Club Med.

Au lieu d'acheter une voiture, achetez une SAAB
Publicité pour SAAB.[1]

Les sections précédentes ont traité de la sémantique grammaticale, en considérant plus les catégories que les mots eux-mêmes. Pour prendre un seul exemple, nous avons représenté le mot *"bureau"* dans la section 3.4.1.3 par le prédicat *bureau(x)*, sans nous préoccuper de distinguer les différents sens que pouvait prendre ce mot (la pièce ou le meuble en particulier[2]).

De plus, connaître les propriétés des concepts ou des objets désignés par les mots peut constituer un moyen de limiter les ambiguïtés de tous types. Ainsi le rattachement du pronom *"it"* dans l'exemple 3.38 devient immédiat si l'on sait que *"move"* implique un déplacement et que ni un nombre (*"536"*) ni un château ne sont déplaçables. De la même façon, l'anaphore 3.36 est résolue facilement en sachant qu'un gorille est une sorte de quadrumane. Enfin l'adjectif possessif *"sa"* de l'énoncé 3.35 est immédiatement relié à la cage si l'on a la connaissance nécessaire (une cage possède une serrure).

Ces exemples illustrent quelques relations importantes pouvant exister entre les mots d'un lexique :

polysémie
- la *polysémie* et l'*homonymie* sont la propriété de certaines formes graphiques (signifiants) de renvoyer à plusieurs sens (signifiés) différents (*"bureau"* est une forme polysémique)[3].

synonymie
- la *synonymie* est le lien entre deux mots ayant le même sens.

hyponymie
- l'*hyponymie* est la relation d'inclusion entre deux mots dont l'un (l'*hyponyme*) est plus spécifique que l'autre (l'*hyperonyme*). Ainsi le gorille est un hyponyme du quadrumane, la fleur est un hypéronyme de la tulipe.

méronymie *holonymie*
- la *méronymie* et l'*holonymie* expriment la relation de partie à tout : la serrure est une partie (méronyme) de la cage. De même, le bâtiment contient (est un holonyme de) une pièce.

[1] Ces utilisations de la polysémie (premier exemple) et de l'hypéronymie (second exemple), ainsi que de nombreuses autres, ont été collectées et analysées par Michel Ballabriga [19].

[2] Mais le *Trésor de la Langue Française* distingue plus de 20 sens différents pour la graphie *"bureau"*.

[3] La différence entre les deux notions réside dans le fait que l'homonymie lie deux signifiés sans aucune relation sémantique (comme *"son"* – le bruit ou le terme de meunerie) tandis que les termes polysémiques comportent des traits en commun (comme le *"bureau"*).

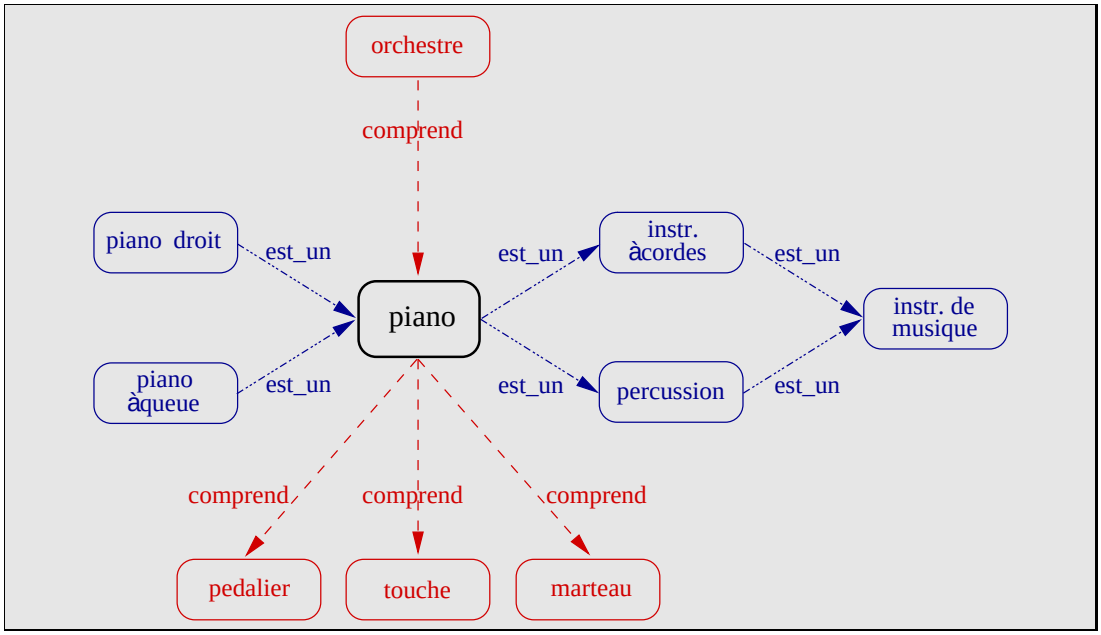


FIG. 3.24 – Graphe de relations lexicales centré sur le terme *"piano"*.

Il est possible de représenter les relations lexicales entre les mots par un graphe, dont une petite portion est donnée à la figure 3.24, centrée sur la notion de *"piano"*. Des bases répertoriant les différentes relations existant entre les mots ont été créées, la plus connue et la plus utilisée (pour le domaine général et la langue anglaise) étant WordNet [161].

WordNet

La section 3.5.3 développe les problèmes spécifiques que posent ces différents phénomènes en recherche d'information.

3.4.5 Vers la pragmatique

On peut prouver que le progrès social est bien meilleur avec du sucre.
Eugène Ionesco, *La Cantatrice Chauve*

Le terme de *pragmatique* regroupe un grand nombre de domaines de recherche, englobant en fait tous les problèmes qui ne peuvent pas être résolus avec la syntaxe et la sémantique. A partir du constat qu' *"un énoncé ne peut prendre tout son sens que si on le replace dans son milieu d'origine"*, elle s'attache à *"l'étude de l'environnement d'une phrase, au moment où elle est émise"* [115, p. 69]. Elle implique l'utilisation de connaissances extra-linguistiques sur le contexte du discours (phrases précédentes, situation et histoire commune des personnages, etc.) et sur le monde en général.

pragmatique

Les études à ce sujet supposent un apport important de connaissances, mais également une analyse fine des phénomènes impliqués. Les applications pratiques sont rares. Cependant il nous semble utile d'énumérer quelques-uns des domaines concernés pour terminer notre tour d'horizon des enjeux de l'analyse du langage et pour souligner le chemin qui reste à parcourir.

La déictique est l'ensemble des allusions directes des interlocuteurs au contexte de l'énonciation : il peut s'agir de références à des entités déjà introduites (comme les *anaphores* déjà abordées) ou supposées connues par les interlocuteurs (comme

le lieu et l'heure courante, phrase 3.43) ou même par exemple montrées avec le doigt dans le langage oral (*déictique gestuelle*, phrase 3.44).

(3.43) J'ai rencontré la reine d'Angleterre *ici même*. C'était *l'année dernière*.

(3.44) Séparons-nous. Je pars avec *toi, toi et toi*, et *vous* partez de votre côté.

Les implicatures conversationnelles concernent ce que l'on peut déduire d'une phrase énoncée en dehors de sa signification littérale. Dans les dialogues qui suivent, la phrase prononcée par B comporte des implications de ce type, dont la signification est résumée après.

(3.45) A : Le voisin est-il chez lui ?
B : Sa voiture est devant le portail.
⇒ le voisin est probablement chez lui, car sa voiture est devant le portail.

(3.46) A : Je suis en panne d'essence.
B : Il y a un garage à deux pas d'ici.
⇒ Il y a un garage près d'ici, vous pourriez y aller pour alimenter votre voiture en essence.

(3.47) A : Georges et Donald vont-ils s'arrêter là ?
B : Sont-ils des hommes modérés ?
⇒ Georges et Donald ne vont bien sûr pas s'arrêter là !

Les présuppositions regroupent les informations que l'on peut déduire sur le contexte lors de l'énonciation de certaines phrases. Encore une fois, contentons-nous d'exemples (ce qui suit la flèche est la présupposition) :

(3.48) Le Roi de France est sage.
⇒ il existe un Roi de France.

(3.49) Jean regrette d'avoir fait ses études à Toulouse.
⇒ Jean est une personne identifiable par le locuteur et le destinataire du message. Jean a fait ses études à Toulouse.

(3.50) Si le vice-chancelier invite Simone de Beauvoir, il regrettera d'avoir une féministe à sa table.[1]
⇒ Simone de Beauvoir est une féministe.

(3.51) Si le vice-chancelier invite le président des États-Unis, il regrettera d'avoir une féministe à sa table.[1]
⇒ le vice-chancelier a également invité une féministe.

Pour avoir plus de détails sur ces problèmes et d'autres (actes de langage, métaphores, structure de la conversation, etc.), le lecteur peut consulter les ouvrages spécialisés [142, 58].

3.5 Le traitement de la langue dans la recherche d'information

La contribution du traitement automatique de la langue (TAL) dans la recherche d'information peut sembler modeste au vu de la problématique générale de l'analyse du langage naturel que nous avons abordée dans les sections précédentes. Il semble en effet illusoire, dans l'état actuel des technologies, d'imaginer une véritable "compréhension" du langage par une machine, ne serait-ce qu'en termes de masse de connaissance nécessaire et de temps passé à manipuler cette connaissance [9, chap. 15].

[1] Exemple de Karttunen cité par [142]

Malgré cela, de nombreux aspects de la linguistique informatique sont utilisés en recherche d'information, avec plus ou moins de succès, que ce soit aux niveaux morphologique*, syntaxique* ou sémantique* [135]. Les systèmes utilisant des bases de connaissance importantes se sont, quant à eux, cantonnés à des applications très spécifiques dans des domaines particuliers [141].

3.5.1 Variations morphologiques

Les traitements morphologiques appliqués à la RI visent tous à opérer des regroupements de mots ayant la même base sémantique, mais ayant subi une flexion* ou une dérivation*. Ces techniques peuvent être utilisées pour regrouper des mots autour d'une base commune (lors de l'indexation) ou à l'inverse pour étendre une requête avec tous les mots dérivés susceptibles d'apparaître dans les documents. Dans les deux cas le but est d'augmenter le rappel* en considérant plus de mots. Le danger est bien sûr de diminuer la précision* si des mots sémantiquement éloignés sont traités par erreur.

stemming

La racinisation (ou *stemming*) la plus courante utilise une approximation des phénomènes linguistiques d'une langue donnée, comme les mécanismes habituels de conjugaison, d'accords en genre et en nombre, ou de dérivation, et tente de supprimer les suffixes tout en regroupant les différents allomorphes (variantes graphiques d'une même racine, comme *"produi-re"* et *"produc-teur"*, ou *"ir-ait"* et *"all-ons"* [231, 86]). Les algorithmes les plus courants pour ce travail sont ceux de Lovins [145] et de Porter [181] (Jacques Savoy pour le Français [206, 207]). Notons que généralement la racinisation d'un mot par ce moyen ne conduit pas à un mot existant (par exemple, la racinisation des mots *"relativisme"* et *"relativisation"* conduiront tous deux à la pseudo-racine *"relativ"*). De plus des mots aux sens éloignés peuvent se voir attribuer une racine commune par ces algorithmes, comme par exemple "cheval", "chevalerie", "chevalier" et "chevalet" [208], avec des effets indésirables en termes de précision. D'autres méthodes tentent d'étendre les requêtes avec des termes morphologiquement proches, acquis de manière plus élaborée [166].

D'autres algorithmes utilisent la catégorie grammaticale (étiquetage morphosyntaxique*), aidés de dictionnaires ou de connaissances linguistiques plus évoluées, pour effectuer une analyse flexionnelle et dérivationnelle plus fine. Ainsi, Flemm [168] utilise un analyseur morphosyntaxique (Brill Tagger [33] ou TreeTagger [211]) et un système à base de règles et d'exceptions pour découvrir quelles opérations morphologiques ont conduit au mot proposé.

L'efficacité des algorithmes de *stemming* est généralement évaluée de façon indirecte, en observant l'effet de leur utilisation sur les résultats des moteurs de recherche. Cette efficacité dépend beaucoup de la langue étudiée [13] et est dans la plupart des cas assez limitée [167]. Récemment une comparaison des algorithmes les plus utilisés en Anglais a été effectuée indépendamment du processus de recherche d'information [59].

3.5.2 Variations syntaxiques

Les connaissances syntaxiques utilisées en pratique par la recherche d'information sont relativement modestes, et se cantonnent généralement à l'étude des syntagmes nominaux*. On peut distinguer :

– Les termes "complexes", des lemmes* formés de deux ou plusieurs termes, qui sont plus précis et moins ambigus que les termes séparés (comme *"pomme de terre"*). Des techniques ont été mises en place pour indexer les termes complexes, par des méthodes statistiques (co-occurrences de termes dans les corpus) ou syntaxiques

(ensembles de patrons prédéfinis) [167].
- Les termes "structurés", que l'on ramène aux relations de dépendance existant entre les termes qu'ils contiennent. Ainsi, les suites de mots *"abattage d'arbre"* et *"les arbres ont été abattus"* peuvent être ramenés à une forme commune de type *"arbre+abattre"* où *"arbre"* est la tête du syntagme. On peut obtenir les deux formes (et d'autres) par application de règles de transformations syntaxiques particulières [68, 222].

L'application de ces différentes techniques à la recherche d'information nécessite des adaptations profondes des méthodes d'indexation et de pondération des termes [167], pour des résultats variables, souvent positifs mais rarement décisifs.

3.5.3 Variations sémantiques

Les variations sémantiques considérées en recherche d'information sont essentiellement d'ordre lexical. Les différents phénomènes lexicaux décrits à la section 3.4.4 peuvent être regroupés en deux classes de problèmes pour la recherche d'information :

- Les relations liant des lexèmes différents mais sémantiquement proches. La synonymie* et l'hyponymie* ont en effet pour conséquence que les termes utilisés dans une requête ne sont pas les mêmes que ceux employés dans des documents en rapport avec cette requête[1], ce qui nuit à la précision*.
- La multitude de sens que peut prendre une forme graphique donnée (polysémie*, homonymie). Le résultat en recherche d'information est qu'un terme employé dans une requête peut se retrouver avec des sens éloignés dans des documents non pertinents[2,3]. Dans ce cas c'est le rappel* qui est touché.

L'utilisation du premier phénomène permet soit d'enrichir les requêtes en utilisant les mots ayant des sens approchés ou en liaison avec les mots initiaux, soit d'effectuer une indexation* plus complexe basée sur les concepts plutôt que sur les mots eux-mêmes. La base WordNet [161] permet ce travail pour des corpus généralistes. Il est aussi possible d'acquérir automatiquement des lexiques sémantiques à partir de corpus [46]. L'utilisation de telles bases de façon automatique se heurte aux difficultés de sélection des relations appropriées et à une accentuation du problème de la polysémie. Il est alors indispensable de mettre en place des mécanismes pour "désambiguïser" les termes polysémiques [13]. La plupart des études sur le sujet [200, 236, 237, 94, 217] aboutissent à des résultats peu convaincants ; notamment les résultats des expérimentations utilisant la base WordNet [69] sont assez mitigés [238].

3.6 Les interfaces de requêtes en langage naturel pour les bases de données

Une autre application du TAL qui nous intéresse particulièrement est la conception d'interfaces en langage naturel pour l'interrogation des bases de données.

[1] Ainsi *Furnas et al.* [84] estime entre 7 et 18 % (selon le domaine, en Anglais) la probabilité pour que deux personnes attribuent le même mot à un objet donné.

[2] Dans la même étude [84], la probabilité pour que deux personnes utilisant un terme donné désignent le même objet est estimée entre 15 et 73 %.

[3] Le Petit Robert annonce ainsi, dans son édition de 2006, un total de 60000 mots et de 300000 sens, soit en moyenne 5 sens par mot, et bien plus pour les mots courants.

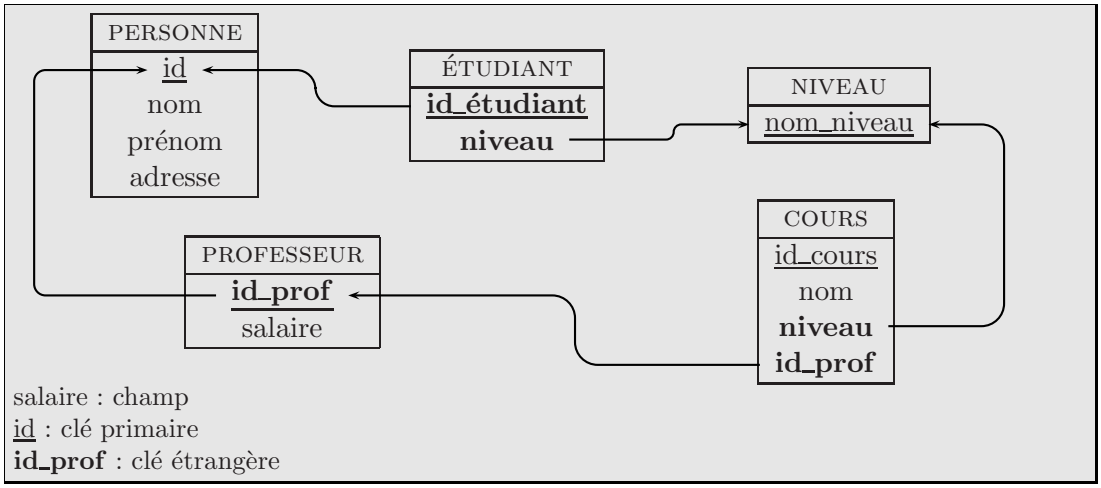


FIG. 3.25 – Exemple de tables de bases de données.

Qui enseigne les mathématiques à Dupont ?

```
SELECT P2.nom, P2.prénom
FROM PERSONNE P1, PERSONNE P2, ETUDIANT E, COURS C
WHERE C.niveau = E.niveau
  AND E.id_étudiant = P1.id
  AND P1.nom = "Dupont"
  AND C.nom = "Mathématiques"
  AND C.id_prof = P2.id
```

FIG. 3.26 – Exemple de requête en SQL pour une base de données.

3.6.1 Intérêts

L'accès aux bases de données est en général effectué à l'aide d'un langage structuré, le plus souvent SQL [72]. La syntaxe de ce langage nous importe peu ; un exemple en est toutefois donné à la figure 3.26, pour donner une idée de sa complexité. La requête concerne les tables décrites à la figure 3.25.

Les avantages d'un système permettant à l'utilisateur de formuler sa requête en langage naturel plutôt qu'en SQL sont évidents. *Hendrix* [105] donne un bon aperçu des capacités attendues d'un tel système, des aspects techniques (accès multiples) jusqu'aux finesses d'analyse (comme la correction orthographique). Une interface en langage naturel évite surtout l'apprentissage d'un langage formel complexe, ainsi que la connaissance rigoureuse de la structure de la base [191]. Elle permet également le *dialogue* avec l'humain [37], et ainsi, notamment, la référence à des questions déjà posées.

Un exemple d'un tel dialogue pourrait être :

1> *Qui enseigne les mathématiques ?*
→ `Taberly, Meallares, Robert, Filustier`
2> *Lequel enseigne en 2ème année ?*
→ `Robert`
3> *En 1ère année ?*

→ `Filustier`

4> *Tous les enseignants ont-ils un cours ?*

→ `oui.`

Ces échanges donnent un aperçu des possibilités offertes pour améliorer le confort d'utilisation d'un système de gestion de base de données ; les trois dernières questions illustrent également des problèmes qui se posent de façon particulièrement aiguë dans ce domaine, et qui ont été décrits précédemment[1] : les références anaphoriques* (question 2), les ellipses* (question 3) et la portée des quantificateurs (question 4[2]).

Les autres problèmes "classiques" d'ambiguïté, également abordés plus haut dans ce chapitre[3], ne sont pas de reste. Il est important de noter que toute erreur d'analyse, à quelque niveau que ce soit, conduit à une interprétation erronée de la requête, et donc à un résultat totalement faux.

3.6.2 Architectures

Les choix importants concernant l'architecture d'une interface pour les bases de données sont d'une part la place accordée à la syntaxe des énoncés, et d'autre part le poids et le mode de représentation de la sémantique.

Ces choix influent sur les caractéristiques du système, notamment en termes de couverture du langage, de portabilité vers une autre base, de robustesse.

De nombreuses architectures ont été mises en œuvre, nous les avons séparées en trois grands groupes : les patrons sémantiques, les grammaires sémantiques et l'utilisation de langages de représentation intermédiaires.

3.6.2.1 Les patrons sémantiques

patrons sémantiques

Nous avons montré précédemment[4] un exemple d'utilisation des patrons sémantiques pour l'extraction d'information. Il est aussi possible de les appliquer aux requêtes en langage naturel [11]. La syntaxe est alors presque absente du système.

Par exemple, les patrons simplistes de la figure 3.27 permettent d'indiquer les actions à mettre en œuvre si le mot *"enseigne"* apparaît suivi par un nom de matière (→ sélection des enseignants pour cette matière), ou le mot *"matières"* suivi par un niveau (→ sélection des matières pour ce niveau). Remarquons que le lexique n'est plus (seulement) composé d'une liste de mots, mais dépend également du contenu de la base de données (ainsi les noms de matières sont enregistrés dans la table COURS).

Ainsi les deux requêtes simples qui suivent sont analysées par les deux patrons de la figure 3.27 :

(3.52) Qui enseigne les mathématiques ?

(3.53) Quelles matières sont enseignées en 1ère année ?

L'avantage essentiel de cette approche est sa simplicité (même si bien entendu les patrons sont en réalité plus complexes que ceux indiqués ici). Cependant de nombreuses constructions linguistiques ne peuvent être traitées, et surtout des énoncés peuvent être mal interprétés.

---

[1] Section 3.4.3, page 68.

[2] Une machine peut en effet interpréter que l'on demande s'il existe un cours qui est donné par tous les enseignants, ou bien (plus probablement) si chaque enseignant s'est vu attribué au moins un cours.

[3] Section 3.3.2, page 53.

[4] Section 3.4.2.4, page 68.

Patrons :

"enseigne" ... MATIÈRE	→	`SELECT P.nom FROM COURS C, PERSONNE P WHERE C.nom =` MATIÈRE `AND C.id_prof = P.id`
"matières" ... NIVEAU	→	`SELECT C.nom FROM COURS C WHERE C.niveau =` NIVEAU
...	→	...

Lexique :

`SELECT DISTINCT nom FROM COURS`	→	MATIÈRE
`SELECT DISTINCT nom_niveau FROM NIVEAU`	→	NIVEAU

FIG. 3.27 – Patrons sémantiques pour l'interrogation de bases de données.

REQUÊTE_ENSEIGNANT	→	PERSONNE_ENSEIGNANT ENSEIGNE_ACTIF MATIÈRE
REQUÊTE_MATIÈRE	→	"Quelles matières" ENSEIGNE_PASSIF "par" ENSEIGNANT
REQUÊTE_MATIÈRE	→	"Quelles matières" ENSEIGNE_PASSIF "en" NIVEAU
PERSONNE_ENSEIGNANT	→	("quel enseignant" | PERSONNE_QUESTION)
PERSONNE_QUESTION	→	"qui" | "quelle personne"
ENSEIGNE_ACTIF	→	"enseigne" | "donne le cours de" ...
ENSEIGNE_PASSIF	→	"être enseigné" ...
ENSEIGNANT	→	(`SELECT DISTINCT id_prof FROM PROFESSEUR`)
COURS	→	(`SELECT DISTINCT nom_niveau FROM NIVEAU`)
MATIÈRE	→	("la" | "le" | "les")? (`SELECT DISTINCT nom FROM COURS`)

FIG. 3.28 – Grammaire sémantique pour l'interrogation de bases de données.

3.6.2.2 Les grammaires sémantiques

grammaires sémantiques

Nous avons décrit les grammaires sémantiques à la section 3.4.2.3 (page 67), en donnant un exemple s'adaptant bien aux interfaces de bases de données. L'avantage de ces grammaires est que la représentation arborescente qu'elles produisent est bien adaptée au domaine traité, et donc que le lien avec les objets de la base de données et la construction de la requête formelle sont facilités.

Un nouvel exemple permettant de traiter notamment les exemples 3.52 et 3.53 de la section précédente est traité aux figures 3.28 et 3.29 (arbre sémantique, que nous pouvons comparer avec l'arbre syntaxique de la même phrase, présenté plus loin – figure 3.30 – pour constater qu'ils ne correspondent ni par les catégories, ni par la structure).

Une explication très détaillée du fonctionnement d'un tel système est la description de LADDER par *Hendrix et al.* [106]. Les limites y sont également clairement exposées. Parmi celles-ci, on peut citer :

– le manque de robustesse face à des constructions non prévues (notamment des

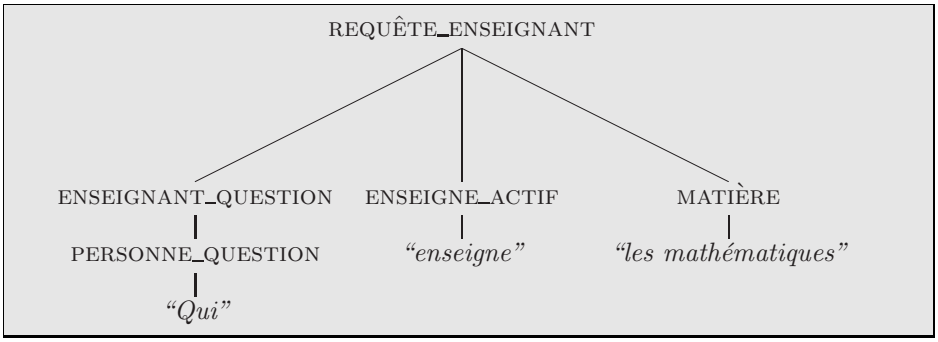


FIG. 3.29 – Représentation arborescente de l'analyse par une grammaire sémantique.

assertions[1]) ;
- l'irrégularité de la couverture (les règles étant très spécifiques, un phénomène peut être très bien couvert tandis qu'un autre est délaissé[2]) ;
- les ambiguïtés syntaxiques et sémantiques ;
- la difficulté de porter l'application dans un nouveau domaine. Pourtant, le système LADDER tente de limiter ce problème en modulant les différentes étapes de l'analyse ; mais la non-portabilité est une contrainte indissociable de la solution des grammaires sémantiques. En effet, les informations lexicales, syntaxiques et structurelles sont combinées dans le même ensemble de règles, sans distinction.

3.6.2.3 Les langages de représentation intermédiaires

L'utilisation d'un ou plusieurs langages intermédiaires de représentation de la requête est un choix très commun. Elle permet de séparer l'analyse en deux parties, dont la première est indépendante de l'application.

Le principe, inspiré du système Lunar [261], l'un des pionniers en la matière, est d'effectuer une analyse syntaxique "classique" de la requête, à l'aide d'une grammaire générale, pour obtenir une première représentation formelle de l'énoncé (figure 3.30). La méthode employée pour traiter cette forme intermédiaire varie alors selon les systèmes, mais le principe général est d'incorporer séparément les connaissances concernant la structure et/ou le domaine de l'application.

Dans Lunar, IRUS [21] ou MASQUE [10], chaque mot de la requête est transformé en un morceau de relation logique, le tout permettant d'aboutir à une expression logique applicable à la base de données[3]. Par exemple, dans Lunar, le pronom "which" introduit l'expression `FOR EVERY X ... PRINTOUT X`, indiquant que tous les éléments d'un type donné doivent être retournés à l'utilisateur. Les questions fermées introduisent le mot-clé `TEST`. En joignant les expressions, on obtient la requête finale. Ainsi, la question *"Does S13004 contains Europium in plag ?"* est traduite en :

[1] Les assertions sont des phrases déclaratives, les plus courantes dans le langage de tous les jours, mais peu employées pour l'interrogation des bases de données, où la forme interrogative est plus appropriée.

[2] Les auteurs donnent l'exemple de la voix passive, qu'il est nécessaire de modéliser pour chaque construction impliquant des verbes. Ce n'est pas le cas des systèmes basés sur les relations grammaticales comme Lunar, qui se place à un niveau d'abstraction supérieur.

[3] Ce qui s'approche de la technique d'interprétation des relations grammaticales décrite à la section 3.4.2.2, page 66.

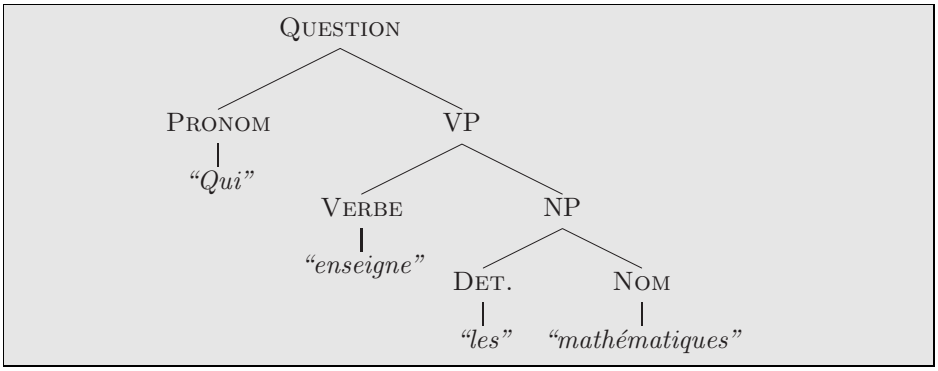


FIG. 3.30 – Représentation arborescente obtenue par l'analyse syntaxique.

```
(TEST (CONTAIN (NPR = X3 / (QUOTE S13004)) (NPR = X4 / QUOTE EU))
                    (NPR = X5 / QUOTE PLAG))))
```

D'autres systèmes proposent une analyse sémantique profonde, liée à l'analyse syntaxique (analyse compositionnelle[1]). C'est le cas de JANUS [108] ou de Chat-80 [256].

Les connaissances qu'il est nécessaire d'ajouter dans ce type d'approches ont une forme très différente de celles des patrons sémantiques par exemple. Dans ce dernier cas, on énumère les constructions possibles concernant l'application, et on propose une interprétation en termes d'interrogation de la base. Ici, l'énoncé lui-même étant traité à part, les connaissances sont insérées de façon beaucoup plus conceptuelle, par un thésaurus, par exemple[2], ou une modélisation du domaine plus approfondie, comme illustré à la figure 3.31 pour une base de données concernant les cours d'une université [54].

Les systèmes utilisant des représentations intermédiaires ont l'avantage de simplifier la portabilité de l'interface, puisqu'une grande partie du processus est indépendante du domaine d'application. Ceci ne signifie pas qu'un changement du type de données est facile ; il est cependant plus localisé. Les deux inconvénients majeurs sont d'une part que l'obtention d'une représentation intermédiaire ne garantit pas le succès de la traduction finale (en quelque sorte, chaque intermédiaire est un risque de plus de se tromper), et d'autre part qu'il est difficile de transformer cette représentation en un langage non conçu dans ce but, comme le standard SQL. C'est pourquoi nombre de systèmes utilisent leur propre langage formel d'interrogation, ce qui nécessite de mettre en place un système de gestion de la base de données propre à l'application.

3.6.3 Inconvénients et limites des interfaces

Le premier problème pointé par les utilisateurs des interfaces est que le langage n'est pas si naturel qu'on pourrait s'y attendre, et que l'usage du système nécessite un apprentissage des structures acceptées ou refusées par l'interface. La couverture n'est pas toujours évidente à comprendre, c'est-à-dire que les raisons pour lesquelles une requête est bien analysée et une autre ne l'est pas sont parfois obscures [11]. Une étude de *Dekleva* [60] montre que, sans entraînement et sans aucune modification, 53,8 % des questions sont traitées correctement (par le système INTELLECT, une interface commerciale parmi les plus connues).

[1] Voir la section 3.4.2 page 64.
[2] Comme présenté à la section 3.4.4, page 72.

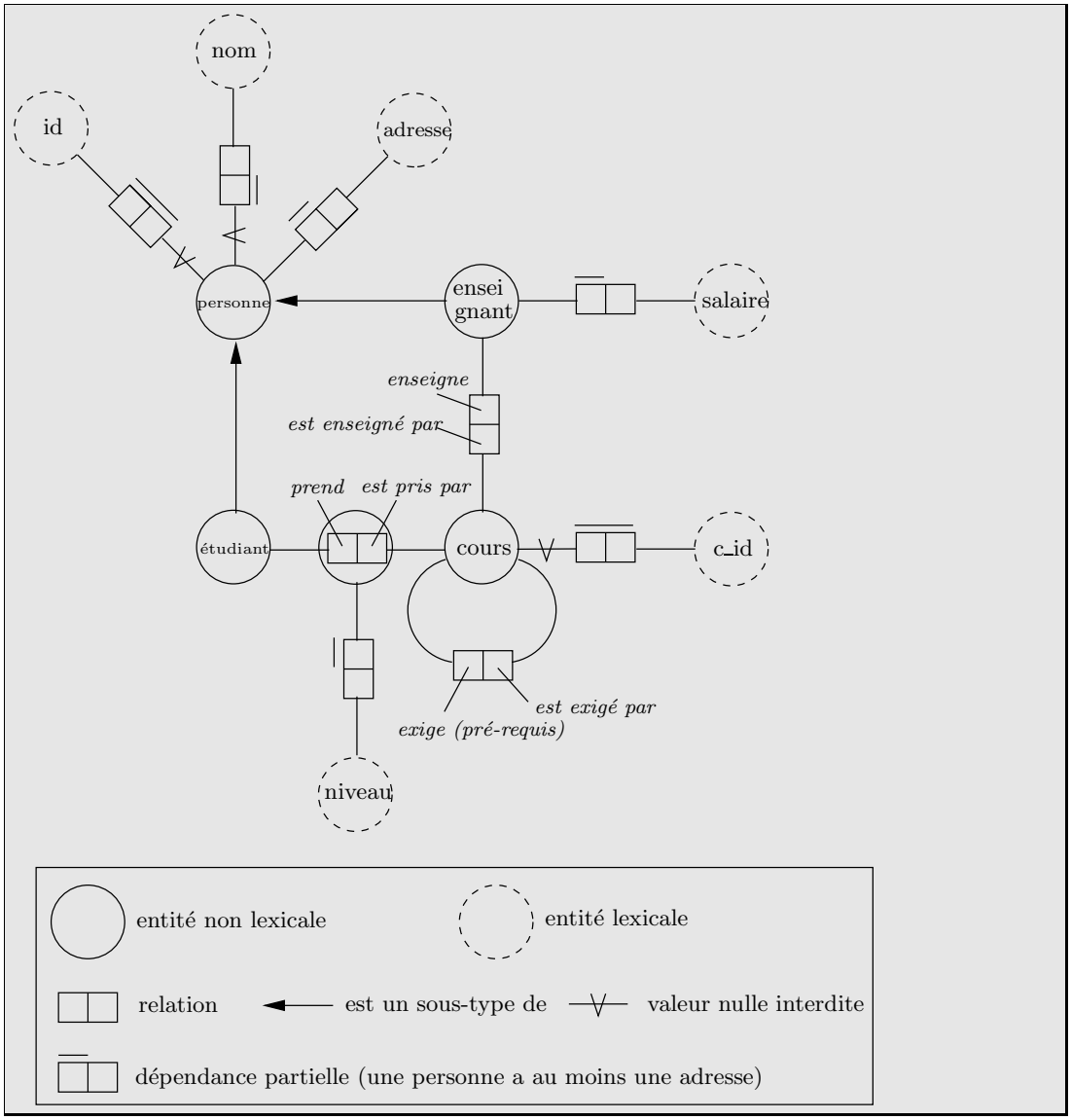


FIG. 3.31 – Modélisation du domaine de la base de données (selon *Copestake et Sparck Jones [54]*).

De plus, lorsque l'interface échoue dans l'analyse d'une requête, il est difficile de savoir si l'erreur vient d'un problème de couverture *linguistique* (qui peut être corrigée en reformulant la demande) ou de couverture *conceptuelle*, c'est-à-dire une question concernant des informations non contenues par la base.

Une autre cause d'égarement de l'utilisateur est que celui-ci s'imagine parfois avoir affaire à un système *intelligent*, capable de raisonnement. Il pose alors des questions faisant appel à du sens commun ou à des capacités de déduction que la machine ne possède absolument pas.

Enfin, il arrive que le système renvoie une réponse erronée, due à une mauvaise interprétation de la requête. Dans ce cas, le risque est que l'utilisateur ne se rende pas compte que la sortie du système ne correspond pas à son besoin.

Ces problèmes de relation avec la clientèle sont accompagnés d'un obstacle technique. En effet, la plupart des systèmes d'interrogation en langage naturel ne sont pas

portables, c'est-à-dire qu'il est impossible, ou très difficile, d'appliquer une interface à un nouveau domaine de connaissance ou à un nouveau système de gestion [54].

Des recherches ont tenté de réduire certains de ces défauts, avec une reformulation de la requête, un diagnostic d'erreur ou des modules génériques, mais ces écueils ont empêché le développement commercial à grande échelle des interfaces en langage naturel. La conclusion de *Capindale et Crawford* [35] (cités par *Androutsopoulos et al.* [11]) est que *"le langage naturel est une méthode d'interaction efficace pour les utilisateurs novices ayant une bonne connaissance de la base de données, qui demandent des réponses précises dans un domaine limité"*[1].

3.7 Les interfaces en langage naturel pour la recherche d'information semi-structurée

Cette section aborde directement la problématique principale du travail décrit dans la seconde partie de ce mémoire. Lancées en 2003 avec la première campagne d'INEX sur le sujet, les recherches concernant les interfaces de requêtes en langage naturel (ILN) pour la recherche d'information semi-structurée n'en sont qu'à leurs premiers pas.

La RI structurée combinant des aspects des bases de données avec ceux de la recherche d'information, il va de soi que les ILN pour ce domaine mêlent également les intérêts et les particularités de ces deux spécialités. S'il n'existe pas vraiment d'interfaces de ce type pour la recherche d'information traditionnelle (il s'agit plutôt d'extraction de mots-clés – voir la section 3.5), nous avons vu à la section précédente que les recherches avaient été très actives à ce niveau pour l'accès aux bases de données.

Cette dernière section introduit ce nouvel axe de travail, en mettant en avant ses motivations et ses spécificités et en décrivant de façon plus détaillée la tâche "Langage Naturel" d'INEX. Nous examinons également les deux seules approches (outre la nôtre) qui ont été l'objet de communications.

3.7.1 Motivation

Certaines motivations présentées ici sont également importantes dans le cadre des bases de données. Cependant nous allons voir que les ILN pour les collections XML correspondent à des besoins différents et possèdent des caractéristiques différentes.

3.7.1.1 Complexité des langages de requêtes

La première motivation qui pousse à s'intéresser à la conception d'interfaces en langage naturel vient du fait que l'expression du besoin d'information dans un langage structuré, possédant une syntaxe et une sémantique formalisées (comme ceux décrits à la section 2.4.2), est trop complexe pour la majorité des utilisateurs. *O'Keefe et Trotman* [172] ont étudié ce problème avec cinq langages de requêtes structurés (HyTime, DSSSL, CSS, XPath, XIRQL) et ont conclu qu'aucun d'eux n'apportait la simplicité d'utilisation nécessaire.

A titre d'exemple, nous pouvons citer le cas de la campagne INEX. Pour cette campagne, comme nous l'avons expliqué précédemment[2], les requêtes (ou *topics*) sont

[1] "Natural language is an effective method of interaction for casual users with a good knowledge of the database, who perform question-answering tasks, in a restricted domain."

[2] Voir la section 2.5.4, page 38.

écrites par les participants eux-mêmes, c'est-à-dire des experts en RI, habitués à manipuler des langages de requêtes. Comme nous l'avons déjà signalé, en 2003, le langage XPath [245] était utilisé. 63 % des requêtes proposées en premier lieu comportaient des erreurs syntaxiques ou sémantiques majeures. Pas moins de 12 tours de corrections ont été nécessaires pour aboutir à un ensemble de topics satisfaisants.

Cette situation a conduit à l'adoption en 2004 de NEXI, un langage très simplifié et beaucoup moins expressif. Le taux d'erreurs initial est tombé à 12 % et le nombre de révisions nécessaires a chuté. Pourtant ces chiffres restent très élevés pour des experts.

De plus, *van Zwol et al.* [233] a confirmé en 2005 par une étude auprès d'utilisateurs novices que la difficulté de formuler des requêtes, même en NEXI, était rédhibitoire.

Quoi qu'il en soit, pour des utilisateurs occasionnels comme pour des experts, le langage de tous les jours ("naturel") semble être le moyen le plus simple et le plus intuitif d'exprimer un besoin d'information.

3.7.1.2 Connaissances de la structure

Outre leur apprentissage, les langages de requêtes formels requièrent de la part de l'utilisateur une très bonne connaissance de la structure des documents dans lesquels le système cherche les réponses aux questions posées. La sémantique de cette structure (des balises) doit également être maîtrisée.

Par exemple, pour retrouver une information provenant d'un résumé, d'une section, d'un élément bibliographique ou d'une figure, un utilisateur devra savoir si ces types d'éléments sont effectivement correctement identifiés dans la structure, et quel est le marquage (les noms de balises) correspondant (respectivement `'abs'`, `'sec'`, `'bb'` et `'fig'` dans les articles de l'IEEE d'INEX). Cette information est contenue dans la ou les DTD ou Schemas XML, mais il s'agit encore d'un nouveau langage à connaître, de plus de temps à consacrer et de nouvelles informations à retenir. Rappelons que la DTD des articles de l'IEEE comporte 192 éléments différents, celle de l'encyclopédie Wikipedia plusieurs centaines.

En outre, il existe des situations pour lesquelles le propriétaire de la collection ne souhaite pas permettre l'accès à ces détails pour des raisons de respect de la vie privée, de maintenance ou de sécurité.

3.7.1.3 Les collections hétérogènes

collections hétérogènes

Dans le cas général, la recherche d'information doit se faire sur un ensemble de documents qui ne partagent pas la même structuration (qui possèdent des DTD ou des Schemas différents). On dit que la collection de recherche est "hétérogène". Même en considérant que les différentes structures sont relativement similaires (avec des unités de recherche semblables), les marquages utilisés sont très divers. Ainsi un paragraphe peut être isolé par des balises `'p'`, `'para'` ou `'paragraph'`. Des acronymes sont souvent utilisés (`'st'` pour *section title*), ainsi que des abréviations plus obscures (`'atl'` pour les titres dans IEEE).

Ainsi, une requête formelle doit être modifiée d'une structure à l'autre, et une recherche dans une collection hétérogène nécessite un nombre parfois considérable de formulations différentes. Une interface en langage naturel, autorisant l'utilisateur à exprimer son besoin de façon conceptuelle, peut permettre de résoudre ce problème, en analysant la phrase indépendamment du langage de requête et en effectuant ensuite une traduction en plusieurs expressions (une par DTD).

3.7.1.4 Une utilisation "intelligente" de la structure

Enfin, les documents utilisant une structure bien conçue et sémantiquement forte pour baliser le texte peuvent rendre la "compréhension" de la requête plus facile. De plus, les subtilités de la recherche dans les documents XML font que la requête d'un humain, même si elle est syntaxiquement correcte et sémantiquement adéquate, peut ne pas être formulée de la meilleure façon possible. Tout utilisateur habitué des moteurs de recherche (sur l'Internet par exemple) sait que ses requêtes, même composées de simples mots-clés, sont des adaptations de son besoin réel aux capacités du moteur qu'il utilise et qu'il a appris à connaître. Nous verrons que, dans le cas des documents XML, cette adaptation est difficile et qu'une expression trop naïve provoque des erreurs.

3.7.2 Ce que les ILN pour XML ne sont pas

Nous avons distingué dans l'introduction du chapitre 2 les différents types d'accès à l'information, dont les réponses technologiques sont notamment la recherche d'information, les bases de données et les systèmes de question/réponse.

Nous explorons ici les connexions et les différences entre la problématique que nous décrivons ici et ces axes de recherche, en insistant sur ce que les ILN pour XML ne sont pas.

3.7.2.1 Une technique de plus pour la recherche d'information

Comme nous l'avons vu (section 3.5), les techniques de traitement automatique de la langue adaptées à la recherche d'information ont été d'une efficacité relativement limitée. Les techniques fines sont finalement peu utilisées. Une des explications possibles est que les conditions habituelles de la RI traditionnelle font que les SRI n'ont pas besoin de "comprendre" la requête pour la traiter. En particulier, le fait que la recherche concerne de larges portions de texte permet d'utiliser des méthodes statistiques, qui s'avèrent plus efficaces que les approches linguistiques, elles-mêmes encore trop peu abouties.

A l'opposé, des domaines traitant des textes plus courts, comme par exemple le "question/réponse", font couramment usage du traitement de la langue. Ainsi nous pouvons espérer que le cas des documents XML apporte un nouveau champ d'étude plus heureux aux partisans des méthodes linguistiques.

auto-explicativité

Enfin, nous avons déjà argumenté contre le principe d'auto-explicativité des doxels retournés par le système (section 2.2.1). Si l'on abandonne ce principe, il devient nécessaire de ne plus rechercher l'information uniquement dans les éléments eux-mêmes, mais également dans leur *contexte*. Nous montrerons dans le chapitre 6 que l'analyse linguistique peut avoir un rôle important dans ce nouvel enjeu.

3.7.2.2 Des interfaces pour les bases de données

BD vs. *RI*

Si quelques-unes des motivations évoquées ci-dessus sont communes avec le domaine des interfaces pour les bases de données, les problématiques restent bien distinctes :

interrogation stricte

– Nous avons déjà souligné que l'interrogation des bases de données est une interrogation *stricte* ; il ne s'agit pas de recherche d'information. L'utilisateur sait quel type de connaissance est stockée dans la base, son besoin est précis et une requête correcte conduit nécessairement à une réponse correcte. Une compréhension erronée par une ILN conduit à des résultats totalement inutiles – voire à

aucun résultat. Ceci implique que l'analyse de la question en langage naturel doit interpréter le besoin de façon parfaite et non ambiguë, faute de quoi la réponse finale est incorrecte et l'utilisateur non satisfait. Pour cette raison les interfaces en langage naturel pour les bases de données ne s'appliquent qu'à des domaines restreints (comme les données géographiques, médicales, etc.) avec des langages également restreints (en fait pseudo-naturels).

interrogation vague

A l'opposé, en recherche d'information semi-structurée, tout comme en RI classique, le besoin d'information est défini de façon vague, et il n'existe pas de réponse parfaite à la question. Une interface en langage naturel devient alors **un composant à part entière du processus de recherche**. Il peut notamment interpréter la requête de façon imparfaite mais retourner malgré tout des résultats utiles. D'un autre côté, il est tout à fait possible d'imaginer qu'une telle interface parvienne à renvoyer de *meilleurs* résultats que des requêtes manuelles (ce qui n'a aucun sens en base de données[1]).

– Qui plus est, les demandes de la recherche d'information n'entraînent, contrairement à l'interrogation des bases de données, aucune opération sur les données, comme les calculs mathématiques, les concaténations, l'agrégation, la restructuration, etc. La réponse est un ensemble d'éléments XML qui font partie de la collection initiale. **Cette différence modifie profondément le type de questions qui seront posées à un système**, et tout autant les approches choisies pour analyser les requêtes en langage naturel. Dans le cas des bases de données, on attend (et on exige) des phrases conventionnelles, exprimant un besoin finalement prévu à l'avance. L'enjeu pour le concepteur d'une interface est surtout de répertorier les constructions qu'un utilisateur sera susceptible d'employer pour parvenir à ses fins.

  En revanche, et paradoxalement, la relative simplicité conceptuelle des requêtes de RI donne une liberté accrue à l'utilisateur pour compliquer à souhait la forme de ses entrées. Seules les éventuelles allusions à la structure des documents permettent de cadrer la syntaxe employée. Les formes négatives sont volontiers exploitées pour préciser les éléments qu'il jugera non pertinents, et l'expression s'étend souvent sur plusieurs phrases. On part cependant de l'hypothèse, et nous reviendrons sur ce point, que les constructions utilisées sont généralement concises, sans ratiocination excessive, et que la personne réfléchit à son besoin avant de formuler sa requête. De plus, aucune recherche n'a encore porté sur la possibilité de dialoguer avec le système après l'obtention du résultat, et chaque requête est traitée de façon indépendante des précédentes.

Pour toutes ces raisons, développer des interfaces pour la recherche d'information XML est un domaine de recherche séparé, nécessitant ses propres solutions innovantes.

Enfin, contrairement aux bases de données, le format XML semble promis à une utilisation par le grand public, notamment au travers de l'Internet. Bien que des requêtes formelles, structurées, non ambiguës et lisibles par des machines soient indispensable pour supporter le processus de recherche, le besoin d'interfaces plus simples devrait devenir de plus en plus important dans le futur.

En retour, les ILN pour les collections XML doivent analyser toutes les requêtes, complexes ou non, syntaxiquement correctes ou non, écrites dans un langage naturel non restreint, même si cette analyse est partielle ou imparfaite. De plus on attend des applications plus générales et des systèmes indépendants du domaine.

[1] On n'aborde pas ici la question de l'efficacité des requêtes en termes de performances en temps de calcul ou en espace utilisé.

3.7.2.3 Des systèmes de question/réponse

Les utilisateurs des systèmes de question/réponse posent des questions fermées, demandant une réponse courte et factuelle ("Quand Napoléon est-il mort ? → "en 1821"). Ces systèmes doivent sélectionner un segment de texte pertinent dans la collection. Dans le cas de la RI pour XML, l'unité de recherche, le doxel, bien que flexible, reste formellement délimitée par la structure du document ; de plus, le besoin d'information est souvent plus général et évoqué par des requêtes ouvertes.

Néanmoins, il existe certains types de requêtes concernant des éléments petits et précis (comme les dates de publications), et pour lesquelles les enjeux dans deux domaines sont similaires. De plus, la recherche d'éléments XML peut servir de recherche de passages efficace dans une phase préalable de pré-traitement en question/réponse.

3.7.3 La tâche "Langage Naturel" d'INEX

Le traitement automatique de la langue naturelle s'est invité pour la première fois à INEX lors de l'édition de 2004. L'unique consigne était alors de n'exploiter que le champ de *description* en langage naturel pour effectuer la recherche[1]. L'intermédiaire utilisé par la tâche *ad-hoc*, c'est-à-dire la requête NEXI écrite par l'auteur, était donc interdite aux participants de cette tâche. Ceux-ci utilisaient toutes les techniques de RI et de TAL qu'ils souhaitaient mettre en œuvre, et renvoyaient un ensemble ordonné de résultats (ou *run*), de la même façon que pour la tâche *ad-hoc*. Malgré un faible nombre de participants, les résultats se sont révélés encourageants, mais encore nettement en dessous des résultats des systèmes utilisant les requêtes formelles manuelles [90].

De plus, un biais théorique important empêchait une évaluation rigoureuse des systèmes. En effet, la comparaison des équipes participatrices concernait la sortie de leurs systèmes respectifs. La performance finale dépendait donc à la fois de la qualité de l'interface de traduction et de celle du moteur de recherche. Il était donc difficile d'interpréter les résultats.

En 2005, une nouvelle épreuve fut proposée. La possibilité fut offerte aux participants de réaliser une interface en langage naturel, transformant la partie *description* de la requête en une liste de mots-clés (*title*) et en un titre NEXI (*castitle*). Dans cette tâche, appelée NLQ2NEXI, aucun moteur de recherche n'était nécessaire. Les requêtes automatiquement générées par les participants étaient lancées sur un moteur de recherche commun fourni par les organisateurs (système $\mathcal{S}$ – voir la figure 3.32). NLQ2NEXI

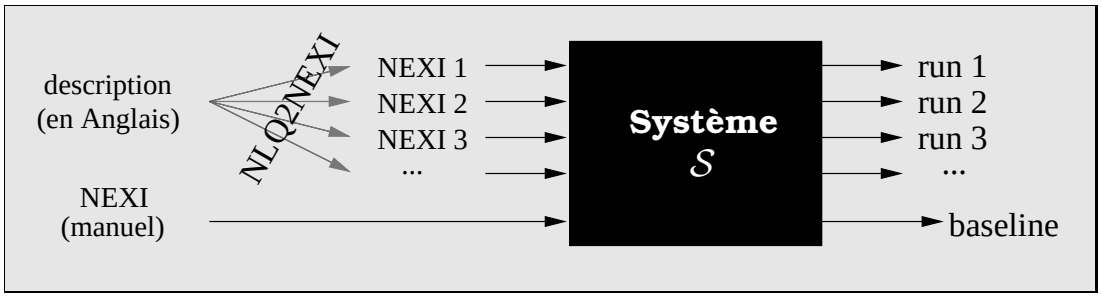


FIG. 3.32 – La tâche NLQ2NEXI.

Dans cette configuration, le moteur est donc une donnée constante. On peut ainsi considérer que l'évaluation des résultats (*run i* sur la figure) est en fait, indirectement, une évaluation des requêtes. Celle-ci a alors deux aspects :

[1] Voir les détails concernant INEX, les requêtes, le langage NEXI, la tâche *ad-hoc* et son évaluation à la section 2.5.4, page 38.

The/DT relationship/NN and/CC comparisons/NNS between/IN radial/JJ basis/NNS functions/NNS and/CC multi/NN layer/NN perceptions/NNS

Find/XIN sections/XST about/XBD compression/NN in/IN articles/XST about/XBD algorithms/NNS

FIG. 3.33 – Etiquetage morphosyntaxique enrichi (*Woodley et Geva*).

– Une comparaison des interfaces entre elles.
– Une comparaison de chaque interface avec la requête manuelle, elle aussi lancée sur le moteur $\mathcal{S}$, aboutissant à un run "témoin", ou *baseline*.

Les mêmes mesures d'évaluation que pour la tâche *ad-hoc* étaient utilisées. Cette fois les résultats ont été très positifs, les performances des requêtes issues des interfaces en langage naturel se montrant comparables, voire meilleures dans certains cas que celles des requêtes manuelles. Nous détaillerons notre propre approche pour la tâche NLQ2NEXI dans le chapitre 5.

3.7.4 Les premières approches

Nous présentons ici les techniques de *Woodley et Geva* [260] et de *Hassler*[1] visant à traduire les requêtes en langage naturel vers des expressions en NEXI.

3.7.4.1 Pré-traitement

La première étape de la traduction est la distinction entre les constructions concernant la structure et celles concernant le contenu des documents recherchés.

Woodley et Geva effectuent une analyse morphosyntaxique de la requête avec Brill Tagger [33] et y incorporent principalement trois types d'informations supplémentaires à l'aide de dictionnaires spécifiques :

– les éléments structurels (*"structure"*, ou XST : références directes à des balises, comme *"abstract"* pour les résumés, *"figures"*, etc.) ;
– les indicateurs de relations (XBD pour *"boundaries"*), qui indiquent un lien entre deux éléments structurels ou entre un élément structurel et du contenu (comme les mots *"about"*, *"containing"*, etc., indépendamment de leur catégorie grammaticale) ;
– les instructions (XIN : *"find"*, *"retrieve"*) qui permettent de distinguer les éléments cibles* des éléments supports*.

Deux exemples d'étiquetage de requête sont donnés à la figure 3.33[2]. Notons que les deux dernières propriétés (XBD et XIN) sont indépendantes de la collection de travail, tandis que la première (XST) demande des connaissances préalables et diffère selon la structure des documents.

[1] Le travail de l'Université de Klagenfurt, par Marcus Hassler et Abdelhamid Bouchachia, n'a fait l'objet d'aucune publication jusqu'à présent. Ce que nous en savons nous vient de discussions informelles avec M. Hassler.

[2] Les exemples de cette section sont donnés en Anglais. En effet, les travaux décrits sont effectués dans le cadre d'INEX, avec des collections et des requêtes dans cette langue. Nos propres travaux ont été effectués sur l'Anglais. De plus, les catégories grammaticales utilisées désormais sont les catégories de la Penn Treebank, énumérées à l'annexe B.1.2.

```
Query: Request+
Request: CO_Request | CAS_Request
CO_Request: NounPhrase+
CAS_Request: SupportRequest | ReturnRequest
SupportRequest: Structure [Bound] NounPhrase+
ReturnRequest: Instruction Structure [Bound] NounPhrase+
```

FIG. 3.34 – Quelques patrons sémantiques présentés par *Woodley et Geva*.

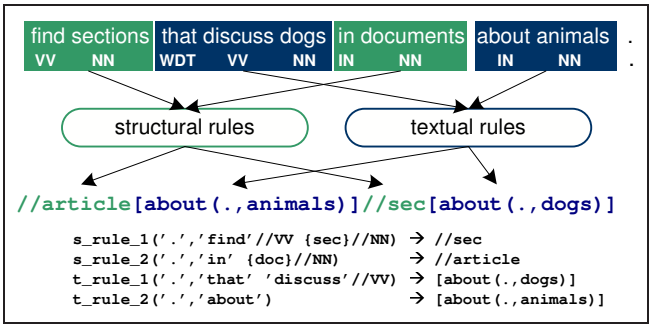


FIG. 3.35 – Analyse de la requête par *Hassler*.

Une analyse syntaxique superficielle est alors utilisée pour distinguer les éléments multi-termes (comme *figure caption* pour la légende d'une figure), mais aussi pour regrouper les syntagmes nominaux simples.

3.7.4.2 Analyse

Puis *Woodley et Geva* utilisent la technique des patrons sémantiques (décrite dans la section 3.4.2.4) pour faire le lien entre les éléments détectés. Ils partent de l'idée que des requêtes concernant la structure forment un contexte particulier pour lequel un nombre limité de patrons suffit [259] (voir figure 3.34). Ceux-ci permettent tout de même de détecter les constructions de base ainsi que certaines négations, la voix passive et quelques tournures anaphoriques [260].

Hassler emploie également des patrons sémantiques, mais basés directement sur les lemmes des termes de la requête et leurs catégories syntaxiques, obtenus avec TreeTagger [211]. Ceci est illustré par un exemple à la figure 3.35. Contrairement à l'autre approche présentée, les liens entre les éléments structurels et le contenu ne sont pas linguistiquement motivés ; il est supposé que le contenu textuel est inclu dans le dernier élément structurel introduit, ce qui est une simplification importante.

3.7.4.3 Formulation de la requête NEXI

La formulation des requêtes NEXI à partir de l'analyse sémantique est opérée de façon simple. L'élément cible est repéré par les termes d'instruction (XIN) chez *Woodley et Geva* ; *Hassler* décrète que la cible est le premier élément structurel de la phrase.

De façons très similaires pour les deux équipes, l'élément cible est placé à droite de la requête avec son contenu (comme le montre la figure 3.35). Ensuite, les parties du support sont disposées à gauche, en partant de l'ancêtre le plus éloigné.

Ainsi que le format de NEXI le permet, les syntagmes simples sont entourés par des guillemets.

3.7.4.4 Limites

La technique des patrons sémantiques est très efficace pour les requêtes exprimées de façon traditionnelle (c'est-à-dire les constructions prévues à l'avance). Elle évite les ambiguïtés syntaxiques mais manque de robustesse, puisqu'une phrase ne correspondant à aucun patron se verra analysée comme un sac de mots sans lien, ceci conduisant à de grandes imprécisions. En conséquence, l'approche de *Hassler*, qui ne comporte aucune phase syntaxique, est surtout performante dans la tâche *Content-Only* (CO), pour laquelle aucune référence à la structure n'est faite.

De plus, aucune des deux approches ne tient compte des allusions implicites à la structure, c'est-à-dire des formes qui introduisent une référence à une balise sans la nommer explicitement. Par exemple, dans l'exemple suivant, la construction *"documents écrits par"* (*"documents written by"*) introduit un auteur (balise `'author'`).

(3.54) We are looking for the documents written by Jiawey Han.
author

Prendre en compte de tels phénomènes demanderait un ajout de renseignements dépendants de la structure des documents, et donc une perte de généricité. S'en abstenir empêche pourtant de traiter bon nombre de tournures de phrases courantes.

Si les informations liées au type de corpus étudié semblent indispensables, au moins au niveau lexical (reconnaissance des termes se référant aux balises), personne n'a utilisé jusqu'à présent de connaissances concernant le domaine des documents (sur l'informatique par exemple pour les articles de l'IEEE). Le choix de l'encyclopédie Wikipedia [61], traitant des domaines extrêmement diversifiés, comme nouvelle collection en 2006 à INEX, n'incitera pas les chercheurs à s'engager dans cette voie.

Deuxième partie

Extraction et recherche d'information en langage naturel dans les documents semi-structurés

Chapitre 4

Les contextes de lecture

Avant d'aborder dans les chapitres suivants notre travail concernant l'interrogation en langue naturelle des collections semi-structurées représentées en XML, nous présentons ici une étude relative au traitement de ces collections, préliminaires à la phase d'interrogation, et même à celle d'indexation des documents.

Nous avons présenté le langage XML dans le chapitre 1, et avons déjà insisté sur le caractère flexible de la structure qu'il permettait de mettre en place. Cette flexibilité est d'ailleurs pour beaucoup dans le succès des technologies XML pour le stockage et l'échange de données numériques. Cependant, nous avons également souligné (section 1.3) les difficultés inhérentes à la mise en œuvre des techniques habituelles d'analyse automatique des documents, notamment pour le traitement automatique de la langue et la recherche d'information. Rappelons que nous ne considérons ici (et dans tout ce mémoire) que l'approche dite "orientée document"* de la technologie XML, c'est-à-dire que la plus grande part du contenu des documents est supposée être du texte en langage naturel.

Ce sont les libertés que le langage XML permet de prendre avec ce texte que nous allons étudier dans ce chapitre. Ainsi, un groupe de mots (ou même un seul mot) peut être coupé par une balise, ou par un élément contenant du texte extérieur au contexte ; d'autre part, des éléments contextuellement très éloignés peuvent être physiquement proches dans le document XML, et inversement. Sans mise en page tenant compte de la *sémantique* des balises, un lecteur ignorant cette sémantique (ou un processeur XML) perd beaucoup d'information dans cette profusion de marquage.

Pourtant il suffit de connaître la "clé", cette sémantique perdue du balisage, pour retrouver sans effort le sens du texte.

Ces caractéristiques propres aux documents semi-structurés ne posent pas uniquement des problèmes de "haut niveau" (comme l'extraction d'information ou les relations sémantiques entre balises, dont les enjeux sont l'objet d'initiatives telles que XML Schema [249] ou le Semantic Web [247] et dont nous ne nous préoccupons pas ici), mais aussi des difficultés plus élémentaires, comme la simple conservation du texte original, que nous présentons dans la section 4.1. En recherche d'information, ces difficultés sont en général ignorées ou esquivées à l'aide de techniques *ad hoc*, propres à chaque corpus de données, nécessitant un apport sémantique *manuel* et une perte d'information structurelle non négligeable.

Nous proposons (section 4.2) un cadre théorique permettant de traiter ces difficultés en introduisant le concept de *contexte de lecture* et en le mettant en relation avec des travaux antérieurs concernant la classification des balises. Les deux dernières sections sont consacrées aux solutions pratiques que nous avons mises en œuvre, tout d'abord

pour déterminer de façon automatique les aspects sémantiques des balises nécessaires à l'utilisation des contextes de lecture (section 4.3), puis pour gérer les contenus XML en tenant compte de cette nouvelle notion (section 4.4).

4.1 Problématique

Les différentes facettes du problème que nous présentons dans cette section sont décrites à l'aide d'exemples inventés ou adaptés d'exemples réels. Le but est de présenter de façon concise et claire les aspects que nous voulons mettre en évidence, ainsi que de souligner leurs incidences sur le processus de recherche d'information. La contrepartie est que l'illustration peut paraître artificielle, mais il nous semble que l'utilisation d'exemples réels aurait nuit à la facilité de lecture et à la concision de l'exposé.

4.1.1 Rattachement des mots

De nombreux phénomènes (indications de forme, notes, mises à jour, transcriptions, insertion de champs de métadonnées) provoquent certaines anomalies dans la continuité physique du texte, et empêchent de distinguer correctement les mots.

Ainsi, dans l'exemple 4.1.a ci-dessous, les mots *'Petit'* et *'Prince'* sont coupés par des balises `'pc'` (petites capitales). Pour retrouver les véritables mots, il est nécessaire de supprimer totalement les balises.

Mais si l'on applique cette méthode aux exemples 4.1.b et 4.1.c, les caractères *'Antoinede'* et *'1943Son'* seront reconnus comme des mots, au lieu de *'Antoine'*, *'de'*, *'1943'* et *'Son'*. Ici les balises `'prenom'`, `'nom'` et `'note'` doivent être remplacées par un blanc.

Pour finir, *'Courrier'* est un terme unique dans 4.1.d, et la `'correction'` qui coupe le mot doit être considérée séparément.

(4.1) a.
```
<titre>
    Le P<pc>etit</pc> P<pc>rince</pc>
</titre>
```

b.
```
<auteur>
    <prenom>Antoine</prenom><nom>de St-Exupéry</nom>
</auteur>
```

c. Livre publié en 1943<note>Son auteur est mort l'année suivante.</note>.

d.
```
<transcription>
    Son premier roman était Courr<correction>14/06/05 ; Le premier
    transcripteur n'avait mis qu'un seul 'r'.</correction>ier Sud.
```

4.1.2 Proximité logique

proximité logique

Il est nécessaire de distinguer la proximité *physique* d'un fichier XML de ce que nous pouvons appeler une proximité *logique*. Tandis que la proximité physique est définie par la position des termes dans le document, la proximité logique dépend de leur organisation dans la structure. Ainsi, dans l'exemple 4.2, les mots "élections" et "aux Etats-Unis" sont physiquement consécutifs, mais logiquement éloignés (nous donnerons une définition plus formelle de la proximité logique dans la section suivante).

Cette distinction est particulièrement intéressante en recherche d'information, principalement lorsque la requête est composée de termes complexes. Supposons que l'on recherche des renseignements sur les élections aux Etats-Unis. L'ensemble d'exemples proposé ci-dessous illustre le fait que la proximité physique des termes "élections" et "Etats-Unis" dans une partie de document ne garantit pas sa pertinence. La pertinence est, en effet, plutôt liée à la proximité logique. Ainsi, les exemples 4.2 et 4.4.c ne sont pas pertinents pour notre recherche, malgré la proximité physique des termes de la requête, tandis que 4.3.a, 4.3.b et 4.3.c, dans une situation comparable (mots séparés par des balises), sont pertinents. Enfin 4.4.a et 4.4.b concernent bien le sujet recherché, malgré les éléments XML insérés.

(4.2)
```
<infos>
    <item>Dernier sondage, à quatre jours des élections</item>
    <item>Aux Etats-Unis, une fausse alerte provoque la panique dans un avion</item>
</infos>
```

(4.3) a.
```
<par>
    Les élections aux <gras>Etats-Unis</gras> sont prévues pour 2008.
</par>
```

b.
```
<titre>
    Les commentaires de Noam Chomski concernant les
    <italique>élections</italique> aux
    <souligne>Etats-Unis</souligne>.
</titre>
```

c.
```
<titre>
    <pc>Elections</pc> aux <pc>Etats-Unis</pc>.
</titre>
```

(4.4) a.
```
<transcription_orale>
    Les nouvelles ne parlent plus que des élections aux Etats-
    <commentaire>Une porte claque.</commentaire>Unis.
</transcription_orale>
```

b.
```
<paragraphe>
    En 2004, les élections<nbp>Nous parlons sûr des élections
    présidentielles</nbp> aux Etats-Unis furent moins controversées
    qu'en 2000.
</paragraphe>
```

c.
```
<résumé>
    Cet article traite du prochain voyage du président roumain
    <nbp>Traian Basescu, qui a remporté de justesse les dernières
    élections</nbp> aux Etats-Unis.
</résumé>
```

Notons que cette idée de proximité est également applicable aux listes, ce qui est particulièrement intéressant. Les listes sont composées d'une introduction, d'un ou plusieurs éléments et parfois d'une conclusion [7]. Dans ce cas, il existe la même proximité logique entre l'introduction et chaque élément de la liste, et entre chaque élément et la conclusion ; ceci est indépendant du nombre et de la longueur des éléments.

(4.5) Les$^{1}_{1}$ macronutriments$^{2}_{2}$ sont$^{3}_{3}$ constitués$^{4}_{4}$ par$^{5}_{5}$:
<liste>
<item>les$^{6}_{6}$ glucides$^{7}_{7}$, appelés plus communément sucres, qui peuvent être séparés en glucides simples et glucides complexes.</item>
<item>les$^{6}_{21}$ lipides$^{7}_{22}$, constituants majeurs des matières grasses, qui englobent les acides gras saturés, mono-insaturés et polyinsaturés.</item>
<item>les$^{6}_{36}$ protéines$^{7}_{37}$, constituées d'acides aminés souvent indispensables car non synthétisables par l'organisme ...</item>
</liste>

Dans cet exemple, les mots *"lipides"* et *"protéines"* sont *physiquement* éloignés du terme *"macronutriments"* (numérotation du bas), mais si l'on prend en compte la proximité *logique* (numérotation du haut), les trois constituants (*"glucides"*, *"lipides"* et *"protéines"*) sont chacun à la même distance de l'introduction.

4.1.3 Traitement automatique du langage

Les fichiers XML orientés document sont, tout comme les documents plats (non structurés), un terrain d'étude intéressant pour les chercheurs désirant appliquer des techniques de traitement automatique des langues à la recherche d'information. Mais dans le cas du XML, un problème supplémentaire réside dans la difficulté de préserver le fil de la lecture de l'être humain, indépendamment des éléments structurels, ce qui est bien sûr indispensable en traitement de la langue.

C'est en particulier nécessaire pour effectuer une analyse morphosyntaxique automatique correcte[1]. En effet, les logiciels d'étiquetage morphosyntaxique déterminent la catégorie grammaticale d'un mot ambigu en utilisant le contexte dans lequel il est utilisé (les termes l'entourant).

Les dépêches suivantes constituent un exemple simple en ce qui concerne l'étiquetage morphosyntaxique.

(4.6) <depeches>
<item *type="exposition"*>Le peintre De Marcis, un artiste ignoré</item>
<item *type="divers"*>Suspect en garde à vue dans l'affaire Rolin</item>
</depeches>

Analysés séparément (par l'outil TreeTagger [211] pour la langue française dans notre exemple), les textes des dépêches sont étiquetés correctement (4.6.a, 4.6.b). Mais accoler les deux textes (sans ponctuation – 4.6.c) avant l'analyse conduit à une erreur (ou en tout cas à une interprétation différente).

(4.6.a) ... *un* *artiste* *ignoré*
... DET :ART NOM ADJ

[1] L'étiquetage morphosyntaxique est expliqué à la section 3.2.1, page 49.

(4.6.b) ***Suspect*** *en* *garde* *à* *vue* ...
NOM PRP NOM PRP NOM ...

(4.6.c) ... *un* *artiste* *ignoré* ***Suspect*** *en* *garde* *à* *vue* ...
... DET :ART NOM ADJ **ADJ** PRP NOM PRP NOM ...

Le problème se pose aussi pour des analyses plus avancées en termes de syntaxe et/ou de sémantique. L'ensemble d'exemples proposé à la section précédente peut être lu "à l'envers" pour illustrer ceci. En particulier, dans l'exemple 4.4, toute analyse linguistique devra bien évidemment considérer comme un tout non dissociable les phrases *"Les nouvelles ne parlent plus que des élections aux Etats-Unis"* (4.4.a), *"En 2004, les élections aux Etats-Unis furent moins controversées qu'en 2000."* (4.4.b) et *"Cet article parle du prochain voyage du président roumain aux Etats-Unis"* (4.4.c), et cela malgré les éléments qui les interrompent. Ici un traitement approprié des balises est nécessaire pour retrouver le texte "initial" (celui qui est destiné à être lu par l'être humain).

4.2 Le contexte de lecture

Il nous semble possible de rendre compte de l'ensemble des caractéristiques présentées ci-dessus en définissant la notion de "contexte de lecture".

4.2.1 Définition

contexte de lecture

Un *contexte de lecture* est une petite partie de texte, syntaxiquement et sémantiquement indépendante et homogène, qu'une personne peut lire d'une seule traite, sans interruption. Dans un document XML en particulier, les contextes de lecture ne respectent pas forcément la linéarité du texte, et dépendent fortement des éléments structurels mis en place. Par exemple :

- Les balises '`item`' de l'exemple 4.6 provoquent un changement de contexte de lecture, car les textes qu'elles contiennent sont sémantiquement distincts et syntaxiquement non compatibles.
- Les balises '`pc`' (petites capitales) de l'exemple 4.1.a apparaissent à l'intérieur d'un contexte de lecture (*"Le Petit Prince"*), ne l'interrompent pas et ne le modifient pas.
- L'élément '`correction`', dans l'exemple 4.1.d, forme un nouveau contexte de lecture qui s'insère dans un contexte existant, celui-ci reprenant par la suite (deux contextes différents : *"Son premier roman était Courrier Sud."* et *"14/06/05 ; Le premier transcripteur n'avait mis qu'un seul 'r'."*).
- Enfin, les listes sont encore plus particulières. Il est possible de les interpréter de façons différentes, selon l'application souhaitée. On peut par exemple estimer que l'ensemble de la liste, introduction et conclusion comprises, compose un seul et même contexte de lecture. Mais il est également possible de considérer que chaque élément d'une liste représente un contexte de lecture indépendant, et que l'introduction et la conclusion sont communs à tous (on aurait donc trois contextes différents dans l'exemple 4.5 : *"Les macronutriments sont constitués par les glucides, appelés plus communément sucres ..."*, *"Les macronutriments sont constitués par les lipides, constituants majeurs des matières grasses ..."* et *"Les macronutriments sont constitués par les protéines, constituées d'acides aminés ..."*). Mais ceci pose des problèmes, notamment de ponctuation, que nous ne souhaitons pas aborder ici.

Les élections aux <gras>Etats-Unis</gras> sont prévues pour 2008.

FIG. 4.1 – Contextes de lecture et balises transparentes

Ainsi, chacun des problèmes exprimés dans la section 4.1 peut se définir ou s'énoncer en utilisant le concept de **contexte de lecture** :

- **rattachement des mots.** Deux chaînes de caractères forment un mot[1] si elles sont accolées sans espace *dans le même contexte de lecture*.
- **proximité logique.** Deux mots sont logiquement consécutifs s'ils sont consécutifs *dans le même contexte de lecture*.
- **traitement automatique du langage.** Une analyse morphosyntaxique, syntaxique, sémantique ou autre doit considérer les contextes de lecture, et non pas le texte dans l'ordre d'apparition dans le document.

4.2.2 Contexte de lecture et classification des balises

Nous reprenons ici à notre compte les trois classes différenciées par *Lini et al.* [143] et abordées dans la section 1.3.2 [2]. Pour chacune d'entre elles, nous reprenons la description donnée par les auteurs, puis nous proposons deux nouvelles définitions. La première définition est exprimée en fonction des contextes de lecture ; la seconde est une définition constructive adoptant une vision plus linguistique. Nous montrerons qu'elle est également plus sujette à des ambiguïtés diverses. Nous la proposons en vue de la catégorisation automatique des balises présentée à la section 4.3.

$$\text{Soient} \begin{cases} n & \text{le nom de la balise dont nous souhaitons déterminer la classe} \\ e_n & \text{un élément XML de nom } n \\ p(e_n) & \text{l'élément parent de } e_n \ (p(e_n) \text{ contient } e_n) \end{cases}$$

Les exemples sont repris des illustrations de la proximité logique (section 4.1.2, page 95).

4.2.2.1 Balises transparentes

soft tags

Les *balises "transparentes"* ("soft" tags) identifient notamment les passages devant subir un formatage particulier, ou ayant une signification spécifique (une citation par exemple). Elles sont "transparentes" lorsqu'on lit le texte. C'est le cas notamment des balises `'gras'`, `'italique'` et `'souligne'` dans les exemples 4.3.a et 4.3.b. Nous en donnons les définitions suivantes :

Définition $\mathcal{D}_1^T(n)$: *Une balise transparente s'inscrit dans le contexte de lecture courant sans le modifier ni l'interrompre (figure 4.1).*

Définition $\mathcal{D}_2^T(n)$: *L'élément* e_n *est un élément transparent s'il est possible de supprimer le balisage et d'obtenir un texte intelligible dans* $p(e_n)$.

[1] La définition du mot se doit bien entendu d'être plus précise, mais cette question ne nous préoccupe pas dans ce contexte.

[2] Page 14.

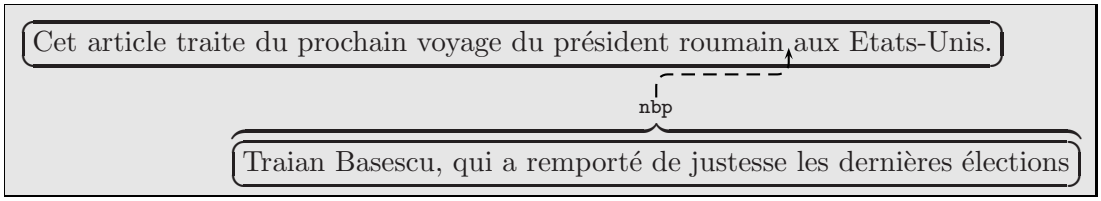


FIG. 4.2 – Contextes de lecture et balises de saut

L'application de cette dernière définition à nos exemples conduit aux textes suivants. L'astérisque en début de ligne indique une phrase incorrecte. Ainsi, seules les balises `'gras'`, `'italique'`, `souligne'` et `'sc'` sont des balises transparentes (cas 4.3'). Remarquons que si la plupart des phrases incorrectes ont une syntaxe défectueuse, la première et la dernière par exemple, si ce n'était les problèmes d'espaces, nécessiteraient des connaissances pragmatiques* pour reconnaître qu'elles sont absurdes.

(4.2') * Dernier sondage, à quatre jours des électionsAux Etats-Unis, une fausse alerte provoque la panique dans un avion

(4.3') a'. Les élections aux Etats-Unis sont prévues pour 2008.

b'. Les commentaires de Noam Chomski concernant les élections aux Etats-Unis.

c'. Elections aux Etats-Unis.

(4.4') a'. * Les nouvelles ne parlent plus que des élections aux Etats-Une porte claque.Unis.

b'. * En 2004, les électionsNous parlons bien sûr des élections présidentielles aux Etats-Unis furent moins controversées qu'en 2000.

c'. * Cet article traite du prochain voyage du président roumainTraian Basescu, qui a remporté de justesse les dernières élections aux Etats-Unis.

4.2.2.2 Balises de saut

jump tags

Les *balises de "saut"* ("jump" tags) représentent des éléments insérés dans le texte courant, comme les notes de bas de page. Les éléments `'commentaire'` et `'nbp'` (note de bas de page) des exemples 4.4.a à 4.4.c sont des éléments de saut.

Définition $\mathcal{D}_1^S(n)$: *Une balise de saut est insérée à l'intérieur d'un contexte de lecture (figure 4.2).*

Définition $\mathcal{D}_2^S(n)$: *L'élément* e_n *est un élément de saut s'il est possible de supprimer l'élément entier (balisage + contenu) et d'obtenir autour un texte intelligible dans* $p(e_n)$.

Dans les exemples qui suivent, la seconde définition ($\mathcal{D}_2^S(n)$) montre que `'commentaire'` et `'nbp'` (note de bas de page) sont des éléments de saut (cas 4.4").

(4.3") a". * Les élections aux sont prévues pour 2008.

b". * Les commentaires de Noam Chomski concernant les aux .

c". * aux .

(4.4") a". Les nouvelles ne parlent plus que des élections aux Etats-Unis.

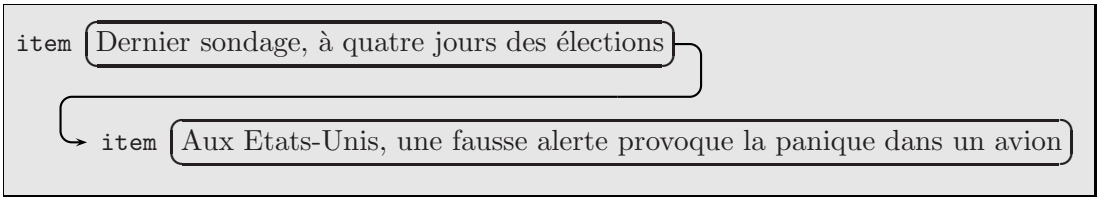


FIG. 4.3 – Contextes de lecture et balises dures

b". En 2004, les élections aux Etats-Unis furent moins controversées qu'en 2000.

c". Cet article traite du prochain voyage du président roumain aux Etats-Unis.

4.2.2.3 Balises dures

hard tags

Les *balises "dures"* ("hard" tags) interrompent la "linéarité" d'un texte. Elles apportent une structuration forte au document (titres, chapitres, paragraphes). Ainsi, les balises `'infos'` et `'item'` sont toutes les deux "dures" dans l'exemple 4.2 de la section précédente.

Définition $\mathcal{D}_1^D(n)$: *Une balise dure provoque un changement de contexte de lecture (figure 4.3).*

Définition $\mathcal{D}_2^D(n)$: Une balise dure a des caractéristiques spécifiques. Pourtant, pour se cantonner avec certitude à nos trois classes et par souci de simplicité, nous proposons : *Une balise dure n'est ni une balise de saut ni une balise transparente.*

4.2.2.4 Commentaires

Il est évident que les définitions constructives $\mathcal{D}_2$ ne peuvent avoir une application isolée ; on ne peut considérer chaque élément séparément, pour les raisons suivantes :

– Si l'élément parent $p(e_n)$ n'a pas de contenu mixte (c'est-à-dire s'il ne contient aucune donnée textuelle, mais uniquement d'autres éléments XML), il sera impossible de conclure, comme le montre l'exemple suivant pour les `'italiques'` :

(4.7)
```
<titre>
    <italiques>Les technologies de l'Espace</italiques>
</titre>
```

– Si la portion de texte considérée contient d'autres éléments que du texte et des éléments de type n, il est impossible de reconstituer avec certitude le texte réel. En effet, nous ignorons en théorie la classe des autres balises :

(4.8)
```
<titre>
    Les <gras>voyages humains</gras> vers
    <ital>Mars</ital> pourraient avoir lieu plus tôt que prévu.
</titre>
```

Dans cet exemple, en s'intéressant à la balise de type `'ital'`, la classe de `'gras'` est inconnue. On ignore donc quelle partie de texte doit être conservée pour l'analyse.

– Dans certains cas, les deux définitions $\mathcal{D}_2^t$ et $\mathcal{D}_2^s$ peuvent s'appliquer à un même élément, et ainsi conduire à une ambiguïté concernant le type (transparent ou de saut) d'une balise :

(4.9) Napoléon Bonaparte <note>qui naquit en Corse en 1769</note> mourut à l'âge de 52 ans après avoir marqué le monde de son empreinte.

(4.10) Le professeur <bold>Stephen Hawkins</bold> travaille sur les lois qui gouvernent l'Univers.

Nous montrerons dans la section suivante comment ces difficultés peuvent être contournées.

4.3 Classification automatique

Pour traiter de façon spéficique certaines classes de balises, quelle que soit la classification (balises dures, transparentes, de saut [49], balises d'emphase [235], balises équivalentes [3]), la liste des balises appartenant à telle ou telle classe est toujours obtenue manuellement (par l'"intuition") ou en utilisant le nom des balises ou encore les commentaires présents dans la DTD [3]. Mais l'intuition prend du temps et a ses limites (par exemple la définition d'une balise d'emphase n'est pas claire), les noms de balises sont rarement de "vrais" mots et les commentaires, leur clarté, leur disposition et surtout leur présence, sont très dépendants de la personne qui les a écrits, ce qui en fait des indices bien trop approximatifs.

Nous proposons ici une méthode pour séparer automatiquement les balises entre les trois classes que nous avons décrites précédemment (balises dures, transparentes, de saut). Cette méthode s'affranchit des contraintes liées à la DTD, puisqu'elle ne l'utilise pas du tout. Elle repose sur les caractéristiques des balises en termes de comportement des contextes de lecture, ainsi que sur une procédure basée sur une analyse syntaxique du texte.

4.3.1 Description de l'approche

4.3.1.1 Approche globale et nouvelles définitions

Nous utilisons les définitions $\mathcal{D}_2$ introduites ci-dessus. Nous remplaçons le terme *"texte intelligible"*, suffisant pour un être humain, mais bien sûr trop imprécis pour un algorithme automatique, par *"texte syntaxiquement correct"*. Nous verrons plus loin comment nous opérons l'analyse syntaxique. Ainsi :

Définition $\mathcal{D}_{2'}^T(n)$: *L'élément* e_n *est un élément transparent s'il est possible de supprimer le balisage et d'obtenir un texte syntaxiquement correct dans* $p(e_n)$. — *soft tags*

Définition $\mathcal{D}_{2'}^S(n)$: *L'élément* e_n *est un élément de saut s'il est possible de supprimer l'élément entier (balisage + contenu) et d'obtenir autour un texte syntaxiquement correct dans* $p(e_n)$. — *jump tags*

Définition $\mathcal{D}_{2'}^D(n)$: La définition ne change pas par rapport à $\mathcal{D}_2^D$: *Une balise dure n'est ni une balise de saut ni une balise transparente.* — *hard tags*

Nous avons décrit plus haut les différentes raisons pour lesquelles ces définitions n'étaient pas applicables efficacement à chaque balise rencontrée séparément. A ces arguments s'en ajoute un nouveau concernant l'analyse syntaxique assistée par ordinateur. Cette technologie est loin d'avoir résolu le problème pour le langage courant. Les analyses erronées restent légion, quel que soit le soin apporté à la conception de

la grammaire ① [1]. Enfin, il est fréquent que des passages de texte ne respectent pas la grammaire d'une langue déterminée, mais soient composés d'abréviations, de fonctions mathématiques, ou d'autres formes de chaînes de caractères qu'un analyseur ne peut reconnaître de façon générique ②. Ainsi, dans des articles scientifiques (type de documents que nous avons tout particulièrement étudié), les formes suivantes sont très courantes :

(4.11) a. La source de bruit <gras>v(t)</gras> représente l'influence de...
b. The Timetables of Technology (rev.), <it>No. 4, 83</it>.
c. Pour chaque Y tel que <ss>symptom</ss>(X, Y) est vrai...

Toutes ces raisons nous incitent donc à adopter une approche plus globale des types de balises. En effet, une classe (transparent, dur ou de saut) est attribuée à un type (un nom) de balise. Les cas où des balises d'un même type peuvent être utilisées de deux manières différentes sont extrêmement rares et souvent dus à des abus de la part des auteurs. Ainsi, nous pouvons mettre en place, en complément de l'analyse syntaxique, une analyse statistique des résultats obtenus. En effet, même si les éléments ne comprenant aucun contenu mixte sont écartés, même si ceux comprenant plusieurs types de balises ne sont pas traités, même si quelques cas provoquent une ambiguïté[2] ③, et même si des phrases correctes sont négligées par l'analyseur ①, notre hypothèse est qu'un nombre suffisant de cas seront traités de façon appropriée pour aboutir à des résultats significatifs (au dessus d'un seuil s ④), et ainsi conclure sans équivoque sur la classe des noms de balises.

De plus, nous estimons que la probabilité pour que l'application des deux définitions $\mathcal{D}_{2'}^{T}$ et $\mathcal{D}_{2'}^{S}$ sur la même balise conduisent toutes deux à des phrases correctes (phénomène illustré par les exemples de Napoléon – 4.9 – et Stephen Hawkins – 4.10) est relativement faible ③.

La conclusion de ces réflexions est que les cas pour lesquels le système conclura *à tort* qu'une classe *s'applique* à une balise donnée devraient être très minoritaires ⑥, à condition d'avoir une analyse syntaxique qui ne valide pas trop de constructions incorrectes ⑤. De cette façon, même s'il arrive souvent que le système conclue, toujours à tort, qu'une définition *ne s'applique pas* ⑦, la différence entre ces deux cas sera suffisamment importante pour permettre de parvenir à se prononcer de façon pertinente ④.

Tout ceci est illustré à la figure 4.4.

4.3.1.2 Analyse syntaxique

Ici l'analyse syntaxique a pour seul but de regrouper des suites de mots. Aucun traitement autre que la simple reconnaissance des structures n'est nécessaire. Nous nous contentons donc de mettre en place une analyse morphosyntaxique[3] avec le logiciel TreeTagger [211] [4], suivie d'une analyse superficielle avec le logiciel *Cass* [5] [5], de Steven Abney, présenté au chapitre 3 [6], dont nous conservons les règles suggérées par l'auteur[7].

Cass

Le problème des ambiguïtés syntaxiques, non gérées par *Cass*, ne nous préoccupe pas. Des rattachements sémantiquement erronés nous importent peu, nous souhaitons

[1] Les numéros entourés de cette section correspondent aux numéros de la figure 4.4.
[2] Voir les explications de tout ceci à la section 4.2.2.4.
[3] Voir la section 3.2.2, page 50.
[4] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html
[5] http://www.vinartus.net/spa/
[6] Section 3.3.3.1, page 55.
[7] A l'exception d'un niveau nommé `nmess`, qui provoque un certain nombre d'analyses erronées, ce que nous voulons éviter à tout prix.

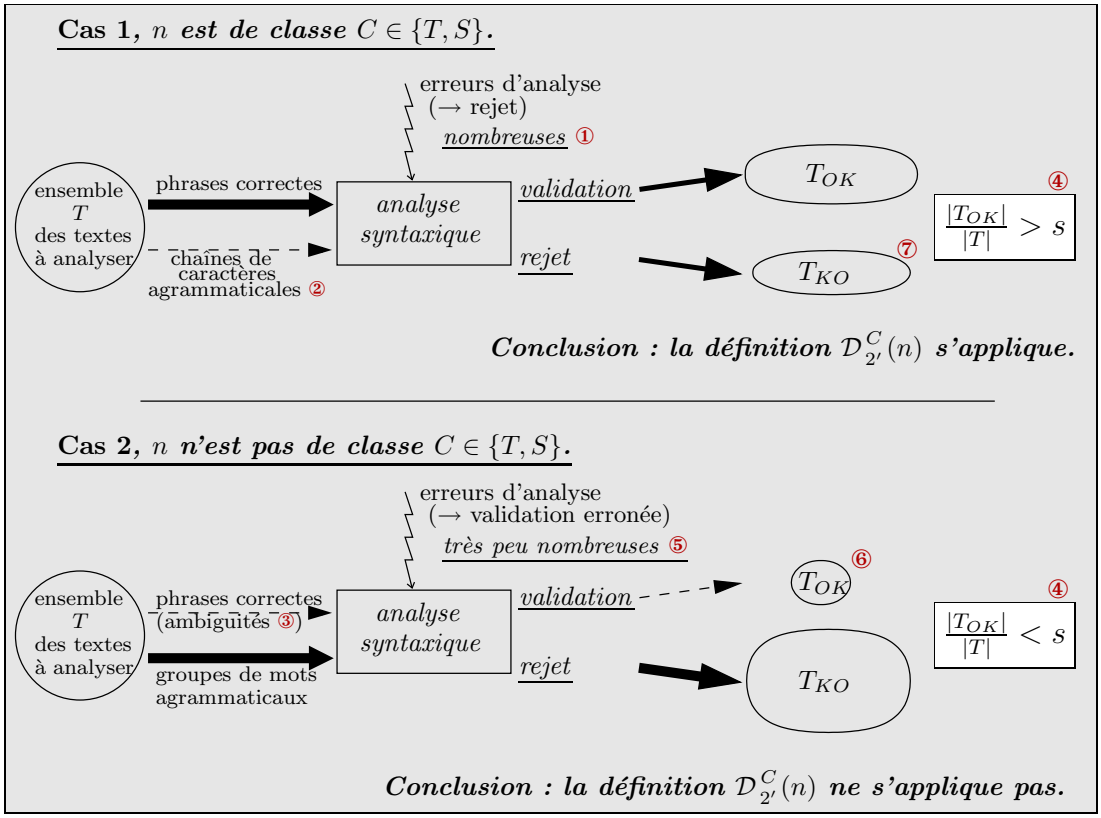


FIG. 4.4 – Vers une application automatique des définitions $\mathcal{D}_{2'}$.

simplement prouver qu'un énoncé respecte une grammaire donnée ou pas. De plus, les collections de documents pouvant être très volumineuses, l'efficacité en termes de temps de calcul est importante, et le logiciel d'Abney affiche de très bons résultats à ce niveau. Enfin, nos contraintes de précision exprimées plus haut trouvent une réponse adaptée dans l'application d'une analyse superficielle. En effet, l'analyse de *Cass* ne concerne que des constituants basiques et ne risque pas de valider des énoncés incorrects.

4.3.1.3 Algorithme de détermination automatique

Pour déterminer si un type de balise n est de classe C ($C \in \{T, S\}$ – transparente ou de saut), on applique l'algorithme suivant :

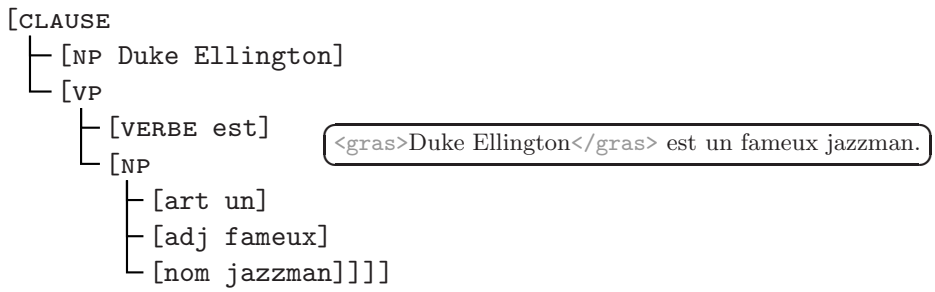
1. *Collecte des éléments.* A partir de la collection complète, tous les éléments parents d'au moins une balise de type n sont sélectionnés ($p(e_n)$).
2. *Filtrage.* Seuls les éléments auxquels il est possible d'appliquer la définition $\mathcal{D}^C_{2'}$ sont conservés (voir section 4.2.2.4) :
   – $p(e_n)$ doit contenir une ou plusieurs parties textuelles, et pas seulement d'autres nœuds.
   – $p(e_n)$ ne doit pas contenir d'éléments autres que du texte et des éléments de type n.
3. *Sélection des portions de texte à analyser.* Pour chaque élément parent $p(e_n)$, le texte adéquat est sélectionné en fonction de la définition concernant la classe recherchée ($\mathcal{D}^C_{2'}(n)$). Pour la transparence ($C = T$), on enlève le balisage et on conserve tout le texte ; pour le saut ($C = S$), on supprime les éléments de type n.

Dans le cas où le texte contient plusieurs phrases (*i.e. si une ponctuation forte apparaît dans le texte*), on réduit la portion de texte à la phrase contenant l'élément étudié. Ainsi, pour $C = T$:

(4.12) <nouvelles>
Sports – Tennis : la Belge <italiques>Kim Clijsters</italiques> est forfait pour l'Open d'Australie. Elle est toujours en convalescence après sa blessure au poignet.
</nouvelles>
⇝ la Belge Kim Clijsters est forfait pour l'Open d'Australie.

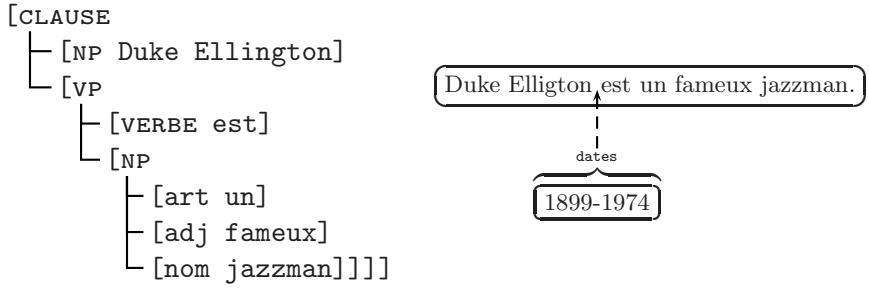
4. *Analyse syntaxique.* On effectue une analyse syntaxique de chacun des textes obtenus. Les critères de validation sont différents selon la classe :
   – Pour la transparence ($C = T$), on considère que l'analyse est valide si un même regroupement syntaxique englobe le contenu d'un élément e_n et du texte situé autour de cet élément :

   (4.13)<gras>Duke Ellington</gras> est un fameux jazzman.

   – Pour le saut ($C = S$), l'analyse est valide si un même regroupement syntaxique englobe du texte situé de part et d'autre de l'emplacement de l'élément e_n.

   (4.14)Duke Ellington <dates>1899-1974</dates> est un fameux jazzman.

5. *Validation statistique.* Si la proportion de textes validés par l'analyseur syntaxique est supérieure à un seuil s, alors on considère que le type n représente des balises de la classe C.

Cette procédure est illustrée par la figure 4.5.

En pratique, pour chaque type n, on effectue d'abord l'analyse pour la classe des balises transparentes. Si on parvient à la conclusion que n fait partie de cette classe, on arrête l'analyse[1]. Dans le cas contraire, on continue pour déterminer si le type n fait partie de la classe "de saut". Si ce n'est pas le cas, on en conclut que les balises n sont des balises dures.

[1] La raison de cet ordre est d'abord que les balises transparentes sont plus fréquentes que les balises de saut. Il s'avère de plus au vu des résultats finaux que les conclusions formulées pour les balises transparentes sont beaucoup plus fiables.

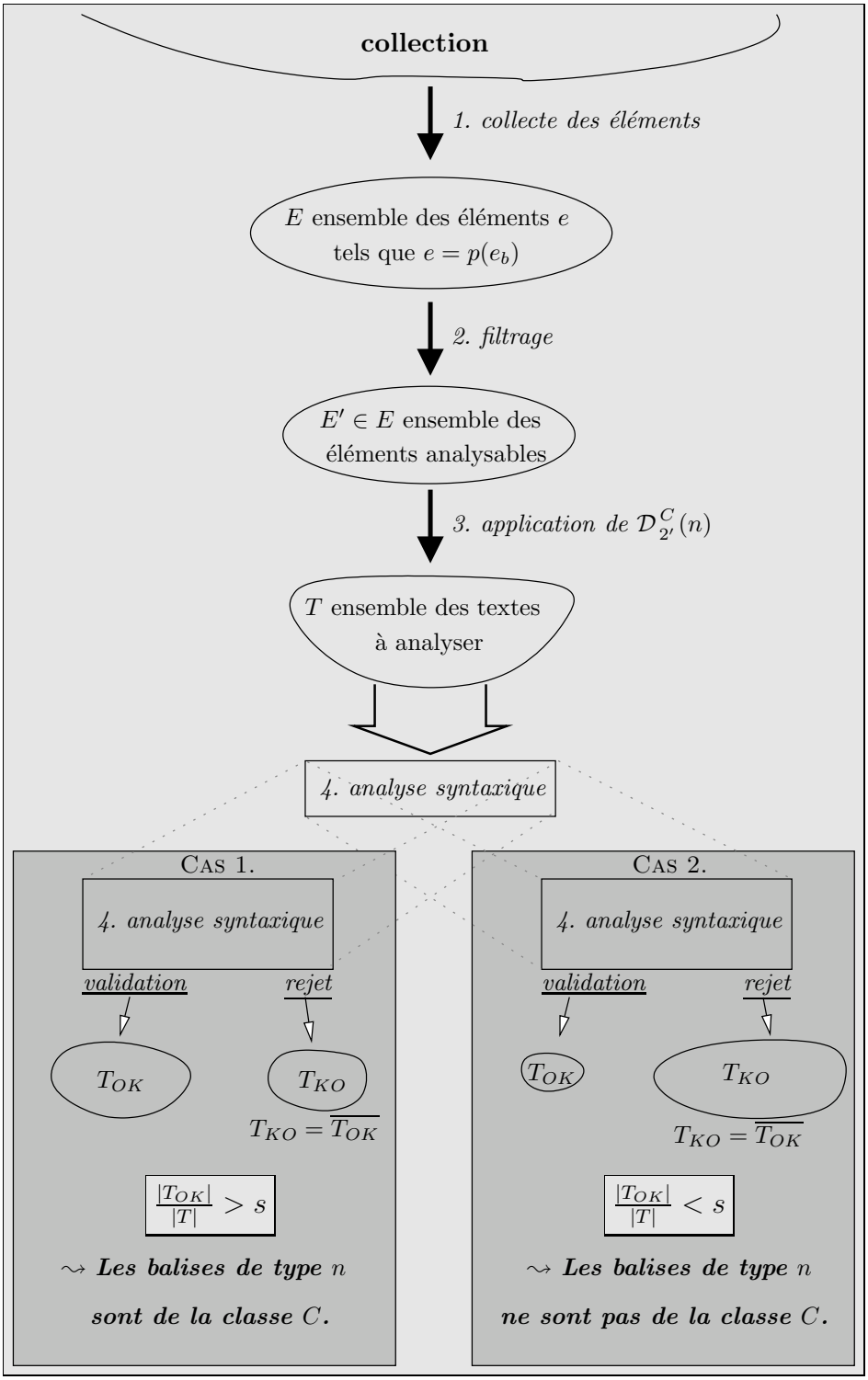


FIG. 4.5 – Attribution d'une classe C à un type de balise n.

4.3.1.4 Expérimentations

Les expérimentations que nous avons menées concernent la langue anglaise. La collection utilisée est celle d'INEX 2004, comprenant 12000 documents et présentée à la section 2.5.4.1. La structure de ce corpus représente aussi bien du balisage logique, comme les sections, les paragraphes, les titres, que des balises de présentations. Au

total, 192 types d'éléments différents sont présents dans la DTD.

4.3.2 Résultats

4.3.2.1 Commentaires préliminaires

Seuil. Comme nous l'avons expliqué, nous estimons que les chances pour que notre analyse syntaxique valide des phrases pour des balises ne répondant pas en réalité à la définition (phrase ambiguë dans le texte original ou erreur de l'analyseur) sont très faibles. En revanche les raisons pour lesquelles l'analyseur peut rejeter des phrases correspondant à la définition sont nombreuses. C'est pourquoi nous avons choisi d'utiliser un seuil s qui peut sembler très bas : 20 %. Nous verrons qu'il suffit amplement lorsque la collection est suffisamment grande (dans nos expérimentations sur la collection INEX complète, tout seuil compris entre 10 et 40 % aurait conduit à des résultats identiques).

Balises dures contre le reste du monde. Nous avons montré qu'un élément devait avoir un contenu mixte (texte + élément e_n) pour pouvoir être analysé. Pour de nombreux types de balises, cette configuration n'arrive jamais dans la collection. C'est le cas des balises logiques comme `'bdy'` (*body*, le corps du texte), `'sec'` (section), `'p'` (paragraphe), `'bib'` (bibliographie), `'bm'` (*back matter*, la partie finale), mais aussi `'au'` (auteur), `'atl'` (titre), `'abs'` (*abstract*, le résumé) et les éléments comme les listes ou les cellules de tableaux. Conformément à l'algorithme mis en place, ces balises sont donc classées comme *dures*, ce qui semble bien correspondre avec la réalité.

Notations. Les notations suivantes sont utilisées dans l'exposé de nos résultats :

- N = nombre total d'apparitions d'éléments de type n dans la collection ;
- $p_f(C, n)$ = pourcentage de ces éléments conservé après le filtrage (C représentant la classe étudiée ($C \in \{T, S\}$) ;
- $s_f(C, n)$ = pourcentage des éléments conservés dont l'analyse syntaxique valide la définition $\mathcal{D}^C_{2'}$;
- $s_t(C, n)$ = rapport entre le nombre de balises validées et le nombre total de balises dans la collection ($s_t = s_f \times p_f$).

Etant donné le nombre élevé (192) des différents types de balises contenus dans la collection, il serait trop long de détailler ici les résultats pour chacun d'entre eux. Nous nous focalisons ici sur les résultats intéressants. Comme nous l'avons dit, les éléments structurants comme les paragraphes, les sections, les listes, les tableaux, les figures, et les données comme les auteurs ou les titres ne franchissent pas la barrière du filtrage et sont directement classés dans les éléments durs.

4.3.2.2 Balises transparentes

La table 4.1 donne un certain nombre de résultats pour l'algorithme de recherche des balises *transparentes*. Dans la première colonne, avec le nom de la balise, figure entre parenthèses la "sémantique" donnée par les commentaires de la DTD. La colonne la plus importante est donnée en caractères gras ($s_f(T, n)$), elle donne le taux des éléments traités (après filtrage) dont le système a validé la correspondance avec la définition.

Pour les balises d'emphase[1], la plupart des scores se situent entre 40 et 70 % d'analyses syntaxiques correctes après filtrage. Pour deux d'entre elles (`'large'` et `'ub'`), le

[1] Nous appelons balises d'"emphase" tout balisage *procédural* et toute marquage de *présentation*

Type (n)	N	$p_f(T,n)$	$s_f(T,n)$	$s_t(T,n)$
a (link)	91	60.44 %	**74.55 %**	45.05 %
ariel	5800	42.38 %	**67.98 %**	28.81 %
b (bold)	160171	38.03 %	**61.40 %**	23.35 %
bi (emphasis)	4233	12.93 %	**58.32 %**	7.54 %
bu (emphasis)	206	29.61 %	**31.15 %**	9.22 %
bui (emphasis)	53	13.20 %	**42.86 %**	13.21 %
it (italics)	1070877	32.94 %	**60.46 %**	19.92 %
large	1	100.00 %	**100.00 %**	100.00 %
math	76270	61.06 %	**17.84 %**	10.89 %
ref (hyperlink)	395946	79.21 %	**2.86 %**	2.27 %
rm (roman)	3548	42.45 %	**49.60 %**	21.05 %
scp (small caps)	114255	52.62 %	**72.70 %**	49.22 %
ss (typeface)	4402	62.49 %	**50.97 %**	31.85 %
sub (subscript)	291661	19.64 %	**21.80 %**	4.28 %
super	44567	31.18 %	**28.93 %**	9.02 %
tt (typeface)	47517	64.45 %	**59.72 %**	38.45 %
u (underlined)	2713	30.48 %	**37.97 %**	11.57 %
ub (medium bold)	2	50.00 %	**100.00 %**	50.00 %
url	25050	54.32 %	**48.39 %**	26.28 %

TAB. 4.1 – Résultats des recherches de balises transparentes.

nombre trop faible d'apparitions de la balise rend les résultats non significatifs (même si ce sont dans les deux cas de "bons" résultats non significatifs !).

Un élément de type `'math'` introduit un environnement mathématique, et `'super'` et `'sub'` sont uniquement utilisés dans ce contexte. Les résultats pour ces types de balises ne sont pas aussi clairs que les autres ; nous avons déjà fait remarquer (section 1.3) qu'on pouvait les considérer comme une exception dans la classification.

Les résultats obtenus pour les balises `'a'`, `'url'` et `'ref'` sont particulièrement intéressants. Les deux premiers représentent quasiment la même information, c'est-à-dire des liens vers des adresses Internet. La différence est que la balise `'a'`, tout comme son homologue en HTML, possède un attribut `'href'` dans laquelle l'adresse est indiquée, et peut donc contenir du texte en Anglais dans son contenu. En revanche, `'url'` ne contient que l'adresse. Ceci résulte en des utilisations sensiblement différentes des deux types de balises, mais, dans les deux, celles-ci interrompent rarement le contexte de lecture :

(4.15) a. Additional information can be found at <url>http ://www...</url>.
(des informations supplémentaires sont disponibles sur <url>...</url>.)

b. The <a *href="http ://www..."*>complete survey results showed that ...

(voir *Coombs et al.* [51] et la section 1.3, page 12). Tous les éléments concernés par la présentation du texte, indépendamment de sa structure, sont compris dans cette catégorie. Nous ne faisons pas dans ce cas la distinction entre balisage de présentation physique et logique. Ainsi, les balises HTML `'EM'` et `'STRONG'` (dites *logiques* parce qu'elles précisent un *rôle* – "emphase" et "fort" – dont le traitement dépend du système de mise en page) sont des balises d'emphase, au même titre que `'B'` ou `'I'` (balises *physiques*, décrivant directement une *apparence*, respectivement *bold* – gras – et *italics* –italiques). A l'opposé, `'TITLE'`, `'H1'` ou `'P'`, qui concernent la structure du document (titres et paragraphe), ne sont pas des balises d'emphase.

Type (n)	*N*	$p_f(S,n)$	$s_f(S,n)$	$s_t(S,n)$
math	76270	61.06 %	**47.68 %**	29.11 %
ref (hyperlink)	395946	79.21 %	**35.86 %**	28.40 %

TAB. 4.2 – Résultats des recherches de balises de saut.

(Les <a>résultats complets de l'étude ont montré que ...)

Enfin, `'ref'` regroupe à lui seul les liens vers les références bibliographiques, vers les figures, les notes de bas de page et d'autres éléments. Son mode d'utilisation est caractéristique des balises de saut ; son taux d'analyses validantes pour les balises transparentes est proche de zéro.

4.3.2.3 Balises de saut

La table 4.2 contient des résultats obtenus pour la procédure des balises de saut. Nous n'avons conservé que les balises `'ref'` et `'math'`, qui sont les seules ayant reçu un score non nul à cette étape (rappelons que les balises ayant été validées comme balises transparentes n'ont pas continué le processus).

Les scores sont dans ce cas moins tranchés que les précédents. La balise `'math'` a toujours des résultats difficiles à expliquer. Pour les références (`'ref'`), seuls 36 % des textes ont été validés par l'analyseur, ce qui suffit cependant à classer ce type dans la classe de saut.

4.3.2.4 Résultats finaux

Avec un seuil de 20 %, nous obtenons en fin de compte les résultats suivants pour la classification des balises. La comparaison manuelle des résultats avec le rôle sémantique des balises est notre seul moyen d'évaluation. Nous avons ajouté entre parenthèses des termes qui proviennent directement des commentaires utilisés dans la DTD ainsi que des déclarations d'entités[1], ainsi qu'une explication en Français :

- Balises transparentes :
  - tface : `'tt'` (typewriter font) and `'ss'` (sans serif) : polices de caractères ;
  - fweight : `'b'` (bold) and `'ub'` (medium) : graisse des caractères ;
  - fslant : `'it'` (italics) and `'rm'` (roman) : apparence des caractères ;
  - fspec : `'scp'` (small caps) : petites capitales ;
  - fattr : `'u'` (underlined) : souligné ;
  - tpos : `'sub'` (subscript) and `'super'` (superscript) : modes indice et exposant (en environnement mathématique) ;
  - tsize : `'large'` : taille des caractères ;
  - emph : `'ariel'`, `'bi'`, `'bu'`, `'bui'` : apparence des caractères ;
  - ref : `'a'` and `'url'` (link to url) : liens vers des URL.
- Balises de saut :
  - ref : `'ref'` (hyperlink) : références diverses ;
  - `'math'` : environnement mathématique.
- Balises dures :

[1] Une entité peut avoir plusieurs rôles en XML. Dans le cas qui nous occupe, il s'agit du regroupement d'un groupe de noms de balises en une seule dénomination. Ceci a des avantages techniques mais a également pour effet d'indiquer un regroupement sémantique.

– toutes les autres...

4.3.2.5 Portée

Cette classification automatique ne s'applique pas à tous les ensembles de documents XML. En particulier, nous rappelons que seule la vue orientée document* est concernée. Une autre contrainte importante est que le corpus considéré doit avoir une taille suffisante (somme toute assez faible – voir la section suivante).

Ces restrictions étant notées, ajoutons que :

– dans le cadre de la recherche d'information, nous sommes justement confrontés à des collections **orientées documents** et **larges**. Les contraintes sont donc naturellement respectées dans ce cas.
– les classes présentées, ainsi que les définitions précises que nous en donnons, ont une réelle utilité, qui ne dépend pas dans l'absolu du fait que la classification est automatique ou pas.

4.3.3 Taille du corpus

La classification automatique ayant une part de traitement statistique, une question importante est celle de la taille de l'ensemble de documents (partageant la même DTD) qu'il est nécessaire d'avoir à sa disposition pour obtenir des résultats pertinents quant à l'application de nos algorithmes.

Plus précisément, la valeur "clé" est le nombre d'occurrences de chaque type de balises. Le corpus d'INEX est bien plus grand que nécessaire dans la plupart des cas, mais malgré sa taille (plusieurs centaines de méga-octets) certaines balises apparaissent un nombre limité de fois (notamment `'large'`, `'ub'`, `'bui'` et `'a'`...[1]).

Pour étudier ce problème, nous avons, *pour chaque type de balise* n, séparé la collection INEX en sous-parties contenant un nombre égal o (o variant au fur et à mesure de l'expérience) de balises n. Nous avons effectué la même partition avec l'ensemble des éléments de type n obtenus après filtrage (Ensemble E' de la figure 4.5, voir section 4.3.1.3).

Nous avons lancé notre système sur ces différentes fractions du corpus. Les résultats de ces expérimentations donnent une bonné idée de la masse de données nécessaires[2].

Nous considérons pour cette expérience que les résultats obtenus pour la collection complète sont corrects. Nous souhaitons savoir à partir de quelle taille une sous-collection conduit aux mêmes conclusions que la collection entière. Pour chaque valeur de o et chaque type n, nous nous intéressons donc au pourcentage de sous-collections dont l'analyse permet de déduire la même classe que pour la collection mère.

Ces informations sont récapitulées dans les figures 4.6 et 4.7 pour cinq types de balises représentatifs. La valeur de o, en abscisse, varie de 1 à 2000. La légende rappelle le nombre total de balises de chaque type dans la collection initiale.

Par exemple, dans la figure 4.6, on peut voir que 200 balises de type `'b'` (*bold*, courbe continue) suffisent à conclure dans 80 % des cas que ce type est transparent.

Il n'est pas surprenant de constater que les résultats concernant la collection filtrée

[1] Cependant on peut arguer que des balises si rares n'ont que très peu d'importance, et qu'une erreur les concernant aura des conséquences très limitées.

[2] Les sous-collections n'ont pas la même taille en octets mais contiennent le même nombre de balises d'un type donné. Nous aurions pu diviser le corpus en parties de même taille ; mais nous pensons que cela aurait conduit à une vision trop dépendante de la collection initiale. Le nombre d'apparition des balises est une valeur générale, que l'on peut appliquer à tout autre ensemble de documents.

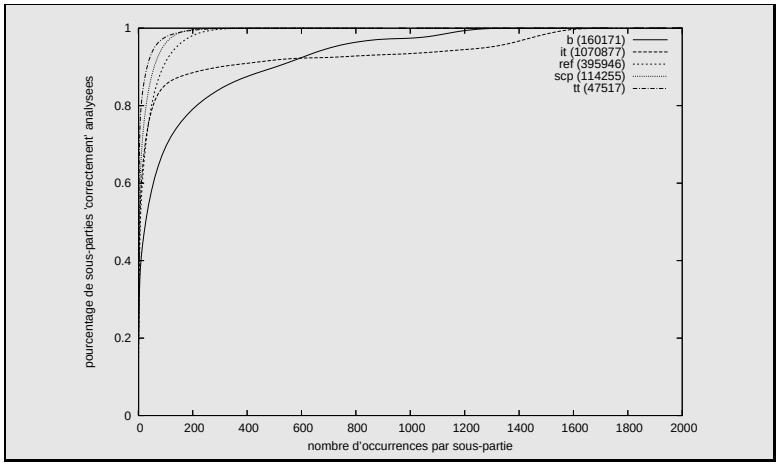


FIG. 4.6 – Influence de la taille de la collection sur la classification automatique des balises ; étude sur la collection non filtrée.

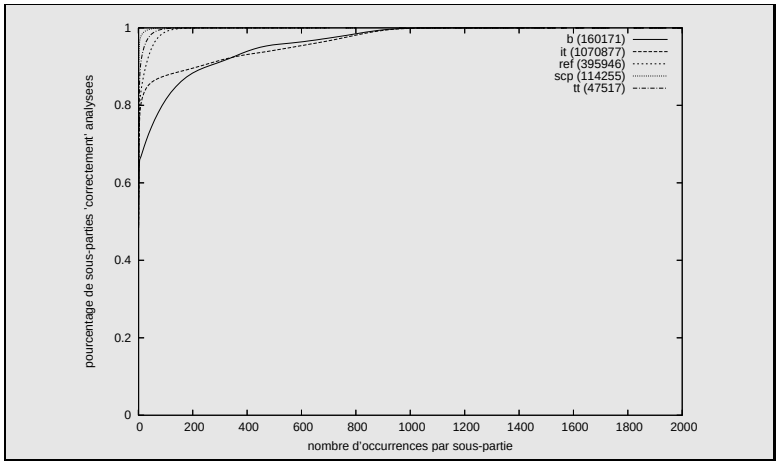


FIG. 4.7 – Influence de la taille de la collection sur la classification automatique des balises ; étude sur la collection filtrée.

sont bien meilleurs. Le fait de ne pas analyser un nombre important d'éléments est préjudiciable. Cela semble indiquer que des efforts portant sur un meilleur filtrage pourraient améliorer les résultats[1].

Les balises `'it'` et `'b'` sont celles qui obtiennent les moins bons résultats : plus de 1000 ou 1500 occurrences sont nécessaires pour obtenir des scores dignes de confiance. Pour tous les autres types (y compris ceux qui ne sont pas représentés ici), entre 100 et 300 occurrences suffisent à conclure.

La raison de cette différence est que les balises `'it'` et `'b'` sont très souvent utilisées dans la collection pour les index ou les informations bibliographiques. La majorité de ces éléments est soit supprimée par le filtrage (par exemple les textes des titres dans les index sont entièrement en caractères gras (`'b'`), et donc de contenu non mixte) soit très difficile à analyser, car contenant de nombreuses abréviations (noms de journaux, dates, numéros de séries...). Ce genre d'éléments est réparti de façon hétérogène dans

[1] Ainsi, pour éviter de supprimer les éléments contenant des enfants de plusieurs types différents (voir section 4.2.2.4), il est possible de mettre en place un algorithme en plusieurs passes dans lequel, une fois la classe d'une balise connue, on lance l'analyse à nouveau sur les autres types pour affiner les résultats.

la collection ; ainsi certaines sous-parties sont composées en quasi-totalité par un index, par exemple, et conduisent donc à des scores très bas. Mais `'it'` et `'b'` sont aussi des balises extrêmement fréquentes, et le nombre d'apparitions nécessaire est très vite atteint.

Rétrospectivement, ces expériences sur la taille du corpus montrent que les résultats concernant des balises rares comme `'a'` et `'bui'` n'étaient pas vraiment significatifs, même si peu surprenants.

4.4 L'outil XGTagger

XGTagger

Nous avons créé l'outil XGTagger dans le but de répondre de façon pratique aux problèmes décrits au début de ce chapitre. Il permet d'adapter n'importe quel système d'analyse textuelle utilisant en entrée du texte pur (non structuré), à des documents XML, en prenant en compte de façon simple mais efficace les aspects liés aux enchevêtrements des contextes de lecture*.

La figure 4.8 indique le fonctionnement global de l'outil. Une description complète et des exemples sont donnés en annexe C.

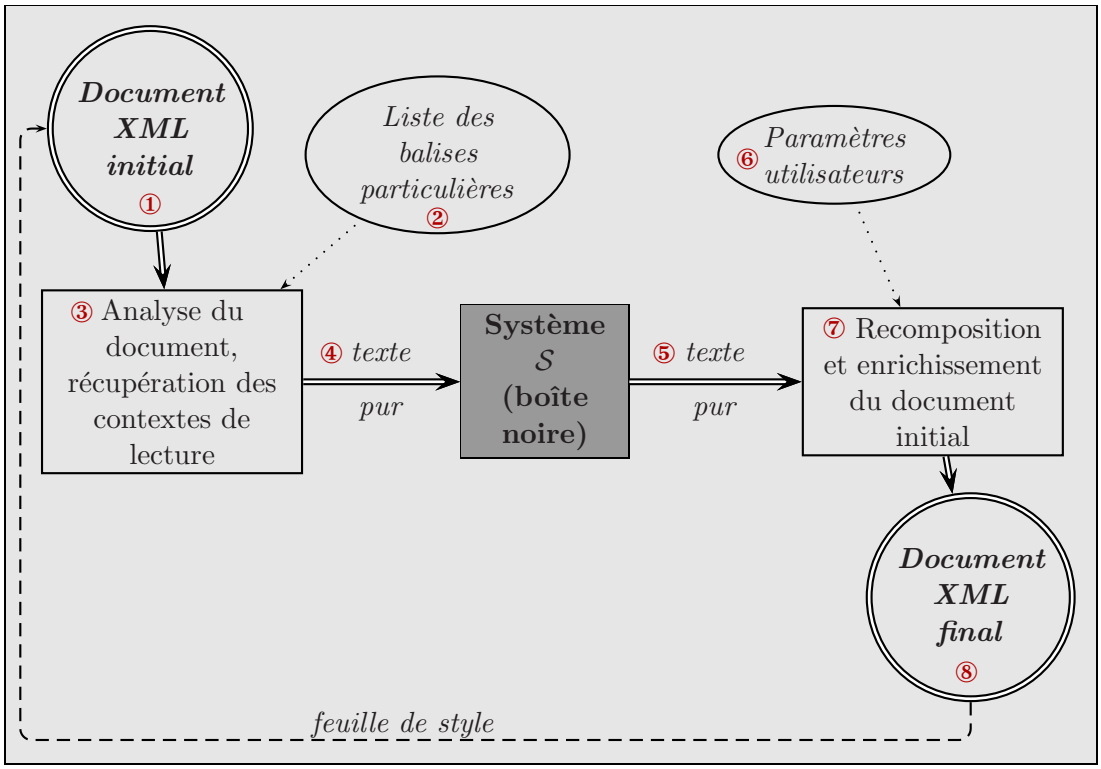


FIG. 4.8 – Schéma de fonctionnement de XGTagger. Le processus est réversible, c'est-à-dire qu'il est possible de retrouver le document initial à partir du document final.

4.5 Conclusion

Tout au long de ce chapitre, nous avons étudié les problèmes posés par la flexibilité de la structure pour l'analyse du contenu des documents XML. A travers la notion de contexte de lecture, puis la classification automatique des balises, et enfin le logiciel XGTagger, nous avons apporté des réponses théoriques et pratiques à ces problèmes.

Tout ceci a pour but de permettre aux utilisateurs de documents XML de conserver tous les avantages des balises, que ce soit au niveau de la structuration ou de la présentation, tout en ayant accès au texte contenu dans ces documents avec la même facilité que pour les documents plats. Ainsi, toutes les méthodes d'analyse textuelle peuvent être appliquées sans aucune perte d'information.

Parmi les tâches facilitées par ce travail, nous pouvons citer, dans le domaine de la recherche d'information, l'indexation. Celle-ci nécessite une reconnaissance correcte des termes, qui est compliquée par l'occurrence fréquente de balises contre les mots, voire au milieu d'eux. De nombreuses autres techniques linguistiques utilisées en recherche d'information, comme la lemmatisation, la racinisation, la prise en compte de la proximité des termes, sont également concernées.

Chapitre 5

Analyse des requêtes en langage naturel

L'utilisation des techniques de traitement de la langue naturelle en recherche d'information traditionnelle (section 3.5) date du développement parallèle de ces deux disciplines à partir des années 1960. De même, des interfaces pour l'interrogation des bases de données en langage naturel ont été développées depuis bien longtemps (section 3.6).

En revanche, dans le domaine de la RI semi-structurée, lui-même relativement nouveau, l'application du traitement de la langue est extrêmement récente. Nous en avons présenté au chapitre précédent une utilisation originale concernant le pré-traitement des données, et nous allons désormais nous intéresser à la mise en œuvre d'une interface d'interrogation.

Nous avons montré à la fin de la première partie de ce mémoire l'intérêt que pouvaient représenter de telles interfaces (section 3.7), ainsi que les particularités d'une telle entreprise, notamment en comparaison des travaux concernant les bases de données.

Les recherches dans ce domaine en étant à leurs tout débuts, nous nous sommes attaché à l'étude des apports nouveaux que pouvaient permettre la linguistique informatique au niveau des spécificités de la RI structurée, en particulier la variabilité de l'unité de recherche et l'imbrication des éléments (section 2.2). Nous n'avons pas cherché à utiliser les techniques "habituelles" en recherche d'information, comme l'extension de requêtes par les variantes sémantiques ou syntaxiques, la lemmatisation* ou n'importe laquelle des nombreuses approches proposées dans la littérature. Appliquer ces idées aux documents XML ne manque évidemment pas d'intérêt, mais tel n'est pas le but que nous nous sommes fixé. Nous nous sommes plutôt employé à démontrer une intuition de départ, à savoir que les constructions linguistiques pourraient permettre d'y voir plus clair dans la structure enchevêtrée des documents, mais également que la réciproque était vraie, c'est-à-dire que la prise en compte de la structuration des corpus faciliterait l'analyse linguistique.

Nous proposons donc ici un modèle de création de requêtes pour des documents XML, se basant sur une description du besoin d'information en langage naturel.

Pour parvenir à une génération efficace de requêtes structurées, une compréhension approfondie des enjeux et des techniques de la RI structurée, de l'indexation à l'interrogation, est indispensable. C'est pourquoi nous avons consacré le chapitre 2 à ce sujet. Une utilisation adéquate des techniques de TAL est bien évidemment nécessaire. Le chapitre 3 en donne un aperçu circonstancié.

Nous détaillons dans ce chapitre les raisons de nos choix concernant le mode d'analyse des requêtes (section 5.1), puis les trois phases principales du processus, qui sont

l'analyse syntaxique (section 5.2), les règles sémantiques (section 5.3) et enfin l'obtention de la requête formelle (section 5.4).

5.1 Enjeux, problèmes et choix

5.1.1 Les contraintes

Les contraintes spécifiques que nous pouvons identifier quant à l'analyse de la requête sont les suivantes :

Structure et contenu. Le système doit permettre une interrogation sur la structure et le contenu (ou le contenu seul) des documents.

Il doit donc reconnaître les allusions à la structure des documents dans la requête de l'utilisateur. Ces allusions sont de plusieurs types :

- Références explicites, utilisées par les utilisateurs connaissant très bien la structure des documents. Exemples : *"les éléments 'atl' "* ou *"les balises 'p' "*.
- Références "quasi"-explicites, emploi de mots "réels" de la langue pour désigner les balises. Ces références ne nécessitent pas une bonne connaissance de la DTD, mais une idée assez nette du type de documents auxquels on a affaire (plusieurs structures possibles). Exemples : *"les titres d'article"* ou *"les figures"*.
- Références implicites, emploi de tournures qui peuvent mettre en œuvre des éléments structurels s'ils existent. Il est possible qu'un utilisateur ignorant la collection qu'il interroge se serve de ces constructions. Exemples : *"Un document écrit par..."* (référence à une éventuelle balise désignant un auteur), *"Une citation de..."* (désignation d'une bibliographie si elle existe, mais une citation – au sens non scientifique du terme – peut se trouver dans le corps du texte, de façon non structurée).

L'interface doit également identifier les relations concernant la structure :

- Relations entre les éléments structurels, en particulier les inclusions. Exemples : *"Un article [...] qui contient une section [...]"* ou *"Une ligne [...] dans un tableau [...]"*
- Relations entre les éléments structurels et le contenu. Exemples : *"Une figure qui parle de..."*, *"Un article sur..."* ou *"Un auteur qui cite..."*.
- Références à la cible*, pour repérer quel(s) élément(s) doit (doivent) être retourné(s) à l'utilisateur. Exemple : *"Je cherche des listes de..."*. Bien entendu, cette référence n'est pas toujours explicite.

Rappelons également que les requêtes peuvent ne faire allusion d'aucune façon à la structure, et ne concerner que le contenu textuel des éléments. L'enjeu est alors de trouver la cible et les relations appropriées. Ce point est notamment traité dans le chapitre suivant.

Robustesse. Les contraintes de "compréhension" de la requête n'étant pas aussi sévères que dans le cas des interfaces pour les bases de données[1], nous pouvons espérer (exiger ?) une robustesse bien supérieure pour les interfaces concernant la recherche d'information dans les documents structurés. Comme nous l'avons dit, l'analyse de la requête est une phase à part entière du processus de recherche, et ce processus doit renvoyer des résultats, même imparfaits.

Nous ne comprenons pas ici la robustesse au sens de la gestion de l'écart entre l'entrée de l'utilisateur et le "bon usage" [75, 243] (correction des fautes de frappe, des fautes d'orthographe, etc.), mais plus simplement en termes de réponse du système lorsque l'énoncé ne correspond pas à la couverture de l'analyseur.

[1] Voir section 3.6 page 76 et section 5.1.2.

Une requête peut être composée d'une ou de plusieurs phrases en langage naturel, avec une syntaxe correcte ou approximative, mais aussi d'un simple groupe nominal, voire d'une liste de mots-clés. Tout cela doit être pris en compte. Si l'analyse d'une phrase échoue, elle doit tout de même collecter toutes les informations possibles sur sa structure. Il est indispensable qu'à l'issue du processus, une requête soit générée.

Généricité. Nous nous fixons comme objectif que l'analyse des requêtes puisse être aussi "générique" que possible, c'est-à-dire la moins dépendante possible du type de structure de la collection d'une part, et du ou des domaine(s) traité(s) dans cette collection d'autre part. C'est ce que nous appelons, par abus de langage, la "généricité". Celle-ci ne pouvant être absolument totale[1], il est important que les parties qui dépendent de la structure ou du domaine soient clairement identifiées et séparées des autres, mais il est également nécessaire de faciliter la mise à jour ou l'adaptation à un nouveau contexte.

5.1.2 Les libertés

Une fois les contraintes énumérées, examinons les hypothèses que nous pouvons faire sur le contexte de notre travail, et qui vont nous permettre de respecter ces contraintes :

Compréhension de la requête. Nous avons vu au chapitre 3 [2] qu'une des différences entre les besoins d'information de la RI et ceux des bases de données était que, dans le premier cas, ces besoins n'impliquent pas d'opérations fines sur les données, comme les calculs de fréquence, de maximum, de moyenne, ou l'agrégation, la jointure, la restructuration des résultats. En conséquence, aucun mécanisme n'a besoin d'être mis en place pour comprendre de telles demandes en langage naturel. Ceci contribue à l'espoir de pouvoir construire des interfaces de requêtes plus robustes et moins dépendantes du domaine[3].

Quantification. Les résultats d'un processus de recherche d'information consistent, quelle que soit la requête, en une liste ordonnée d'éléments supposés pertinents. Les problèmes de portée des quantificateurs[4], très importants pour les bases de données, n'ont pas d'importance ici.

Indépendance des requêtes. Nous considérons que les requêtes sont indépendantes les unes des autres. Aucun dialogue ne s'instaure avec l'utilisateur, et aucune allusion à une précédente demande n'est effectuée. Ceci réduit les problèmes sémantiques comme les ellipses et les anaphores[5], même s'il arrive bien sûr que de tels phénomènes adviennent à l'intérieur d'une même requête.

Courtoisie de l'utilisateur. Nous estimons également que l'utilisateur qui écrit sa requête, s'il a certes la possibilité de la formuler en langage naturel, n'en profitera pas pour dévier de son but. En quelque sorte, on attend de lui qu'il respecte les maximes de quantité (donner la quantité d'information nécessaire), de pertinence (rester en relation avec le sujet) et de manière (être clair) de *Grice* [97]. En pratique, ceci ne semble pas être un problème majeur ni une restriction très contraignante. Le respect de ces maximes permet surtout d'éviter le bruit provoqué par la présence de mots n'ayant aucun rapport avec le sujet.

[1] Dans notre cas, l'interface ne dépend pas du tout des domaines traités par les documents, mais contient un certain nombre de paramètres concernant leur structure.

[2] Section 3.7.2, page 85.

[3] En effet, plus l'analyse sémantique nécessitée est fine, plus les connaissances apportées doivent être importantes, et donc plus les restrictions concernant les constructions employées sont fortes.

[4] Section 3.4.3.3, page 71.

[5] Voir la section 3.4.3, page 68.

5.1.3 L'heure du choix

Les observations des sections précédentes montrent que des liens *sémantiques* entre les éléments structurels et le contenu doivent être déterminés. Des constructions particulières, des termes particuliers ont une *signification* spécifique que la simple structure syntaxique ne suffit pas à déterminer.

Pourtant il n'est pas nécessaire d'obtenir une représentation sémantique évoluée pour analyser correctement une requête. De nombreux termes ne participent pas aux constructions dont nous parlons, et leur sens a somme toute peu d'importance pour construire la requête formelle[1]. L'apport sémantique est localisé mais néanmoins très important.

Nous avons montré au chapitre 3 [2] qu'il existait plusieurs modes de représentation et d'analyse sémantique. Nous y revenons maintenant pour choisir notre propre piste de travail.

5.1.3.1 Les accessits

L'analyse sémantique fine[3], avec pour résultat une représentation logique complète du sens de l'énoncé, ne correspond ni aux besoins ni aux contraintes de notre problématique. Peu robuste, nécessitant un apport de connaissances lexicales important, elle apporte de plus un niveau de détail global qui ne nous est utile qu'en certains points.

grammaires sémantiques

Les *grammaires sémantiques*[4], très utilisées par les interfaces pour l'interrogation des bases de données[5], sont très appropriées pour des domaines particuliers et pour des formes de requêtes prévues à l'avance. L'application à un nouveau domaine demande de concevoir entièrement une nouvelle grammaire [9, p. 334]. Là encore, cette technique s'adapte donc mal à nos contraintes spécifiques.

template matching

La méthode utilisée par les autres interfaces de RI semi-structurée de la littérature est celles des *patrons sémantiques* (ou template matching[6]) en général plutôt appliquée pour l'extraction d'information spécifique dans un texte. L'utilisation des patrons, éventuellement associée à une analyse syntaxique superficielle, permet de se concentrer sur les parties de la requête pour lesquelles une analyse sémantique est jugée utile. L'autre avantage est une extrême robustesse ; les parties de texte non concernées par les patrons peuvent être considérées comme une liste de mots-clés, par exemple, et le travail consiste alors à éliminer les mots "vides", de liaison, non significatifs pour la requête.

Le template matching semble donc intéressant, mais produit une analyse sémantique très superficielle. Dans notre cas particulier, de nombreuses erreurs sont susceptibles d'être provoquées par la difficulté de relier les parties qui s'accordent avec un patron et celles qui ne s'accordent avec aucun, et donc ne sont pas analysées. De plus, si les patrons sont très appropriés pour ce que nous avons appelé les références "explicites", ils le sont moins pour les références "implicites", qui peuvent être exprimées par plusieurs formes linguistiques différentes dans la requête. D'ailleurs, les interfaces que nous avons décrites à la fin du chapitre 3 [7] ne semblent pas traiter des constructions telles que

[1] Rappelons que nous ne cherchons pas à effectuer des traitements sémantiques usités en recherche d'information, comme la désambiguïsation ou l'expansion de requête.

[2] Section 3.4, page 61.

[3] Section 3.4.2.1, page 64.

[4] Section 3.4.2.3, page 67.

[5] Section 3.6, page 76.

[6] Section 3.4.2.4, page 68.

[7] Section 3.7.4, page 88.

"un article écrit par quelqu'un" (référence à un auteur). Enfin, on peut s'interroger sur les capacités d'une telle technique lorsque les constructions à détecter deviennent très nombreuses, plus fines [9, p. 340], ou plus génériques (voir par exemple notre utilisation de la syntaxe des groupes nominaux au chapitre 6).

5.1.3.2 Le premier prix

Il est possible de conserver bon nombre des avantages des patrons sémantiques tout en réduisant les problèmes qu'ils posent, en s'appuyant sur une analyse syntaxique profonde et en appliquant des règles de conversion des *relations grammaticales* en relations sémantiques, comme la section 3.4.2.2 (page 66) le décrit.

relations grammaticales

L'utilisation des règles sémantiques à un certain niveau d'abstraction autorise la mise en place de règles plus conceptuelles, plus faciles à rédiger, moins dépendantes des constructions linguistiques utilisées dans la requête.

Ainsi, si l'on prend l'exemple de *"l'article écrit par quelqu'un"*, en sortie de l'analyse syntaxique, nous savons que *"article"* est l'objet du verbe *"écrire"*, et que *"quelqu'un"* en est le sujet (sujet réel, inversé par la voix passive), et ce, quelle que soit la forme utilisée (voix active ou passive, adjectifs, adverbes, proposition relative s'insérant dans la phrase, etc.[1]). Une seule règle est alors nécessaire pour prendre en compte ce type de références implicites à un *auteur*.

De plus, les portions de texte qui ne sont concernées par aucune règle sémantique sont toujours liées aux autres par les dépendances grammaticales, ce qui permet de faciliter l'établissement de relations supplémentaires.

Enfin, une telle méthode permet de séparer clairement la partie *générique* de l'analyse, qui ne dépend pas du tout du type de documents auquel on a affaire, de la partie *spécifique*. Nous verrons que, dans ce but, nous mettons en œuvre plusieurs niveaux de règles sémantiques. Ainsi, l'effort et le temps nécessaires pour adapter le système à une nouvelle structuration sont fortement réduits. De plus, nous avons mis en place un cadre facilitant la création de nouvelles règles ou l'insertion de nouvelles données lexicales.

En contrepartie de ces avantages, les enjeux d'efficacité et de robustesse, ainsi que les problèmes d'ambiguïté, sont repoussés vers le niveau syntaxique de l'analyse.

En ce qui concerne la robustesse, nous proposons une analyse à plusieurs niveaux : une analyse profonde tout d'abord, mais néanmoins relativement flexible au niveau de la grammaticalité des constructions ; en cas d'échec, une analyse plus restreinte, ne concernant que les formes les plus importantes, notamment les groupes nominaux. L'inconvénient majeur de ce choix est la performance (en termes de temps d'analyse). Cependant, nous ne souhaitons pas analyser des textes entiers, mais seulement des requêtes de quelques phrases au maximum, ce qui réduit fortement les conséquences de ce problème.

Quant aux ambiguïtés syntaxiques, nous verrons que les plus importantes d'entre elles (c'est-à-dire celles pour lesquelles un choix erroné serait néfaste à la pertinence de la requête finale) peuvent être résolues grâce aux règles sémantiques. Pour les autres, un algorithme "naïf" de résolution s'avère relativement satisfaisant, car les erreurs qui en découlent ont peu d'influence sur le résultat final.

[1] Exemple de variations : *"l'article que quelqu'un a écrit"*, *"l'article long soigneusement écrit par quelqu'un"*, *"Quelqu'un a écrit un article"*.

5.2 Analyse syntaxique

5.2.1 Les besoins

Les réflexions et les choix décrits à la section précédente nous conduisent à énumérer les caractéristiques attendues de l'analyse syntaxique :

Robustesse. Nous avons déjà détaillé cet aspect. Une analyse à plusieurs niveaux peut permettre d'obtenir un système qui retourne toujours une solution, même imparfaite.

Relations grammaticales. Les principales relations grammaticales que nous souhaitons reconnaître sont le *sujet*, l'*objet direct* ou *indirect* d'un verbe et les compléments prépositionnels. Les groupes nominaux simples sont soumis à un traitement particulier avant l'analyse globale.

Représentation. Nous souhaitons conserver le même type de formalisme pour la représentation syntaxique et la représentation sémantique. Les règles sémantiques que nous mettrons en place devront s'intégrer simplement dans ce formalisme.

Simplicité. Si l'analyse syntaxique elle-même est destinée à être relativement immuable, les règles sémantiques, elles, doivent être aussi faciles à gérer que possible (création et mise à jour). Il est donc préférable d'utiliser un formalisme simple à comprendre, nécessitant peu de connaissances linguistiques.

Connaissances lexicales. Pour que l'application soit large, les connaissances lexicales nécessaires au fonctionnement de l'analyseur doivent être limitées aux apports des règles sémantiques[1]. Toute connaissance demandant l'utilisation d'un lexique à large couverture doit être évitée (par exemple, la valence des verbes – transitif direct, indirect, intransitif...).

Ambiguïtés. Nous avons annoncé que les ambiguïtés les plus importantes (pour notre application) pourraient être résolues par les règles sémantiques. Dans le cas où ces règles ne sont pas suffisantes, les erreurs sont acceptables. Cependant, une seule analyse doit être sélectionnée pour la construction de la requête ; il est donc nécessaire de désambiguïser du mieux possible, tout en évitant l'intégration excessive de connaissances, lexicales ou autres.

Nous ne nous basons sur aucune théorie syntaxique particulière pour notre analyse. Nous utilisons, selon nos besoins, un certain nombre de caractéristiques couramment employées dans le domaine, pour effectuer une analyse par constituants, à l'aide d'une grammaire à clauses définies écrite en Prolog[2]. Ceci nous permet de considérer et de propager tous les types d'information utiles, et également d'effectuer toutes les actions que nous jugeons nécessaires au fil de l'analyse. Quant à l'écriture des règles de repérage des relations syntaxiques, elle ne nécessite pas particulièrement de recours aux théories [31].

5.2.2 La représentation

Comme nous l'avons indiqué, le choix de notre représentation est guidé par un besoin de simplicité et de compatibilité avec les règles sémantiques. Ces règles utilisent les relations grammaticales d'une part et les informations lexicales d'autre part. Ces dernières ne sont pas seulement le lemme ou la catégorie grammaticale, mais d'autres traits peuvent également être associés à un mot. De plus, il sera nécessaire de détecter

[1] Exception faite des ressources exploitées par l'étiqueteur morphosyntaxique.
[2] Voir la section 3.3.3.5, page 59.

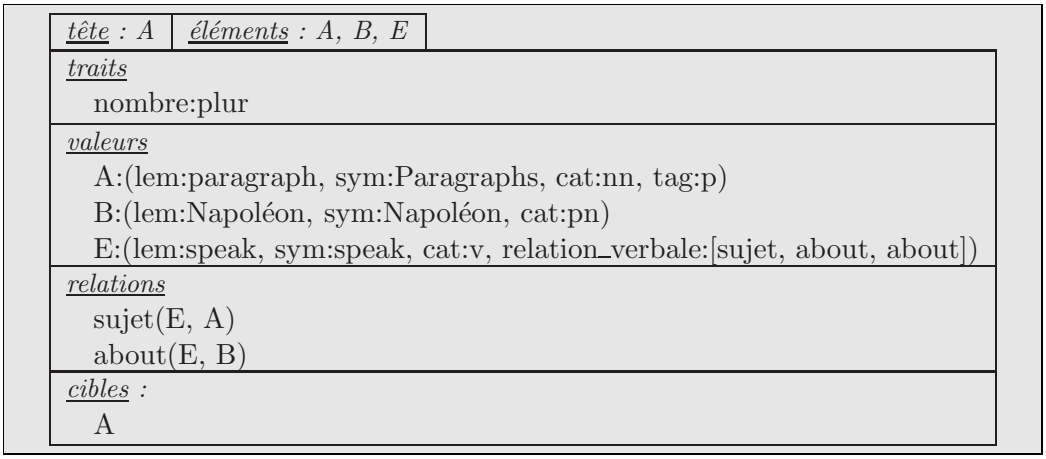


FIG. 5.1 – Représentation de l'énoncé : *"Paragraphs that speak about Napoléon."*

la cible* de la requête, c'est-à-dire l'élément que l'utilisateur souhaite voir retourner en réponse à sa demande.

Ces trois éléments (*relations*, différentes *valeurs* caractérisant les unités lexicales, *cibles*) doivent donc être isolés dans la représentation. C'est pourquoi nous choisissons, comme dans les représentations logiques, de désigner les mots par des variables, auxquelles on peut donc faire référence à chaque niveau.

D'autres composants sont utiles à la progression de l'analyse : la *tête** du constituant, pour construire les relations entre les bons termes, et certains traits de base permettant une analyse plus précise (genre, nombre, indicateur de négation, etc.).

Nous nous inspirons du mode de représentation de la DRT[1] en mettant en place une structure à plusieurs étages, ce qui permettra notamment de construire des règles sémantiques de façon intuitive. Un exemple simple de cette structure est donné à la figure 5.1, pour la requête suivante :

(5.1) Paragraphs that speak about Napoléon.[2]

Parmi les valeurs, on trouve *lem*, le lemme du terme traité, *sym*, sa valeur initiale dans l'énoncé, *cat*, sa catégorie grammaticale, ainsi que d'autres attributs qui seront utiles pour l'analyse sémantique et que nous décrirons plus tard. La cible désignée est A, le paragraphe[3]. Les *traits* sont ceux de la *tête*, ici A[4].

En pratique, cette représentation est transcrite en un prédicat Prolog reprenant la même structure. La figure 5.2 montre le prédicat correspondant à l'exemple 5.1.

L'analyse permettant d'obtenir ce type de représentation est effectuée en deux temps, en commençant par un traitement particulier de certains groupes nominaux.

5.2.3 Avant l'analyse : pré-traitement des syntagmes nominaux

La plupart des langages de requêtes pour la recherche d'information, structurée ou non, permettent de représenter les groupes de mots indissociables les uns des autres, en

[1] Voir la section 3.4.1.2, page 62.

[2] Nous utilisons désormais des exemples en Anglais, langue d'application de notre interface.

[3] En pratique, cette information est obtenue par l'analyse sémantique. La représentation de la figure 5.1 n'est qu'un exemple permettant de comprendre l'utilité de chaque champ de la structure.

[4] Il aurait d'ailleurs été possible de considérer la tête comme un trait.

```
repr(A, [A, B, E],                                          %% tête, éléments
     [nombre:plur],                                         %% traits
     [A:(lem:paragraph, sym:paragraphs, cat:nn, tag:p),     %% valeurs
      B:(lem:Napoléon, sym:Napoléon, cat:pn),
      E:(lem:speak, sym:speak, cat:v,
                 relation_verbale:[sujet, about, about])],
     [sujet(E, A),                                          %% relations
      about(E, B)],
     [A]).                                                  %% cibles
```

FIG. 5.2 – Représentation de l'énoncé : *"Paragraphs that speak about Napoléon"* par un prédicat Prolog.

général avec des guillemets. Ainsi, il est possible d'effectuer, par exemple, une recherche sur les *"pommes de terre"* sans distinguer les mots *"pommes"* et *"terre"*. A l'exception de requêtes extrêmement précises, les termes de ce type sont des syntagmes nominaux simples, de trois ou quatre mots maximum.

En Anglais, et du point de vue de la recherche d'information, les groupes nominaux ont la forme générale suivante [13] :

$$NP \rightarrow det^* \; pre^* \; head \; post^*$$

Où *det* est un déterminant, *pre* (*premodifier*) un adjectif, un nom ou un ensemble des deux (coordonné ou non), *head* est un nom, la tête du groupe et *post* (*postmodifier*) un groupe prépositionnel ou une proposition relative.

Toutes ces formes sont considérées pour l'analyse des requêtes en langages naturels, mais deux comportements bien distincts sont adoptés.

5.2.3.1 Les syntagmes nominaux simples

En Anglais, les syntagmes nominaux les plus simples consistent en une succession d'adjectifs ou de noms, suivie par le nom de tête :

$$NP \rightarrow (JJ \mid NN)+ \; NN \tag{5.2}$$

Ces termes, comme nous l'avons vu au chapitre 3 [1], sont moins l'objet d'ambiguïtés sémantiques que les noms simples, et font en général référence à un domaine particulier [167]. Contrairement aux termes plus complexes que nous abordons plus loin, ils sont peu sujets aux variations syntaxiques[2] ; il est ainsi relativement probable que la forme d'un tel terme représentant un concept donné ne change pas entre la requête et la plupart des occurrences de la collection.

Pour ces raisons, ces NP simples sont très intéressants en recherche d'information. L'ensemble des séquences de mots obéissant à la règle 5.2 sont donc regroupés et consi-

[1] Section 3.5.2, page 75.
[2] Ceci étant plus valable pour l'Anglais que pour le Français.

Topic	Terme
204	"semantic networks" (JJ NN)
231	"graph theory" (NN NN)
210	"multimedia document models" (NN NN NN)
211	"global positioning systems" (JJ NN NN)
204	"Dan Moldovan" (PN PN)
204	"Federal Bureau of Investigation" (PN PN IN PN)

TAB. 5.1 – Exemples de syntagmes nominaux simples.

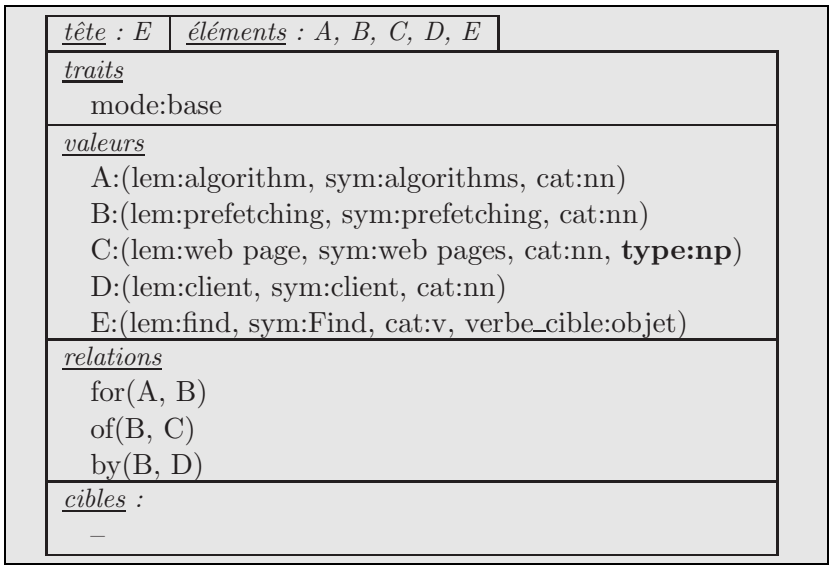


FIG. 5.3 – Représentation de l'énoncé : *"Find algorithms for prefetching of web pages by the client"*.

dérés avant le début de l'analyse comme des noms communs simples (*nn*)[1].

Des exemples (issus des topics d'INEX 2005) sont donnés à la table 5.1. Deux autres phénomènes sont considérés en pré-traitement : la suite de noms propres, éventuellement entrecoupés de prépositions (règle 5.3 et table 5.1), est considérée comme un nom propre seul (*pn*) ; enfin, les mots d'une séquence encadrée par des guillemets dans la requête de l'utilisateur sont également regroupés en un seul nom.

$$NP \rightarrow PN\ (IN?\ PN)+ \tag{5.3}$$

Dans tous les cas, un attribut supplémentaire (`type:np` pour les règles 5.2 et 5.3, `type:quot` pour les guillemets) est ajouté à la valeur concernée. Un exemple est donné à la figure 5.3.

5.2.3.2 Les syntagmes nominaux complexes

Par ailleurs, d'autres groupes nominaux sont plus complexes, même s'ils ne sont guère plus longs (nous ne parlons pas ici des groupes nominaux composés de propositions relatives ou de groupes verbaux) ; il s'agit des noms ou syntagmes reliés par des

[1]Notons que le problème de la reconnaissance locale des syntagmes, posé à la section 3.3.1.1 (page 51), se pose moins dans notre cas, puisqu'aucune préposition n'est concernée.

Topic	Groupe nominal
208	history of Artificial Intelligence
216	the architecture of a multimedia retrieval system
217	user-centered design for web sites
219	the granularity of learning objects
220	annotations in image retrieval
233	development of synthesizers for music creation
276	evaluation measure for clustering

TAB. 5.2 – Exemples de syntagmes nominaux complexes.

prépositions :

$$NP \rightarrow NP\ (IN\ NP)+ \tag{5.4}$$

Eux aussi sont sémantiquement très significatifs [231, 14] et apparaissent aussi fréquemment que les constructions vues précédemment (voir la table 5.2). Pourtant il serait tout à fait hasardeux de les considérer comme un terme unique. La raison principale est qu'ils sont sujets à de nombreuses variations syntaxiques[1].

Nous verrons néanmoins que ces groupes de mots auront une grande importance pour une partie de notre travail.

5.2.4 L'analyse

Les caractéristiques de notre application nous incitent à mettre en œuvre un mode d'analyse simple. Rappelons quelques besoins de notre application :

- véhiculer des informations sémantiques bien précises (éléments structurels, verbes spéciaux, etc.[2]) ;
- connaître le lemme des mots de la requête, sur lesquels les règles sémantiques s'appliqueront ;
- reporter les réductions des ambiguïtés à la phase sémantique, et donc générer toutes les interprétations possibles pour une requête ;
- pouvoir analyser des phrases complètes, mais aussi des énoncés sans verbe ou incomplets ;
- renvoyer un résultat, quel que soit l'énoncé ;
- analyser certains constituants, notamment les groupes de mots entre guillemets, de façon particulière ;
- associer aux constituants les relations grammaticales de base ;
- ne nécessiter aucune connaissance lexicale, sur l'aspect ou la valence des verbes par exemple.

Les analyseurs disponibles[3] sont souvent très efficaces, mais aucun ne combine toutes ces contraintes. Il serait complexe, voire impossible, de les adapter à nos besoins. Nous

[1] Voir la section 3.5.2 page 75. Avec des exemples en Anglais, on peut imaginer que le syntagme *"annotation in image retrieval"* du topic 220 puisse se trouver, avec peu ou pas de modification du sens, sous la forme *"annotate images for retrieval"*, *"retrieve annotated images"*, *"annotated image retrieval"*, *"retrieval of annotated images"*, *"images have been annoted for retrieval"*, etc.

[2] Voir la section 5.3.

[3] Citons par exemple les systèmes Link Grammar (`http://hyper.link.cs.cmu.edu/link/`) et ERG (`http://www.delph-in.net/erg/`).

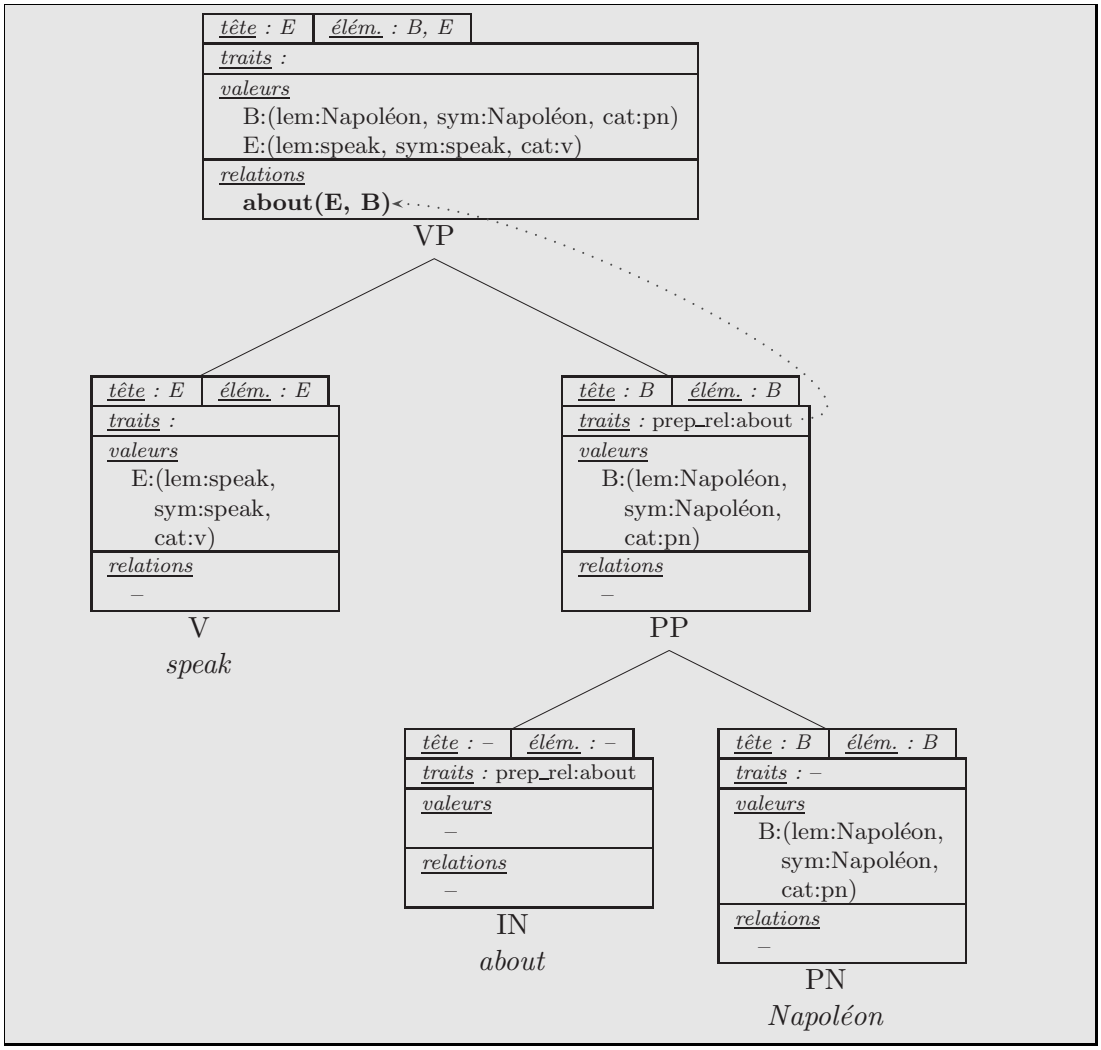


FIG. 5.4 – Analyse syntaxique de *"speak about Napoléon"*.

avons donc décidé de mettre en place notre propre analyseur ; néanmoins nous nous inspirons bien entendu très fortement des systèmes existants, à tous les niveaux. C'est pourquoi décrire l'analyse mise en œuvre dans tous ses détails serait long et sans grand intérêt, celle-ci ne comportant pas d'innovations particulières. Nous nous contentons donc d'aborder les aspects spécifiques de notre système.

5.2.4.1 Le fonctionnement général

Nous utilisons une grammaire à clauses définies, prenant appui sur la grammaire traditionnelle, sans autre apport lexical que les informations utilisées par l'étiqueteur TreeTagger et les ajouts sémantiques propres aux règles que nous aborderons plus tard.

Les traits servent à éviter des attachements abusifs, mais avec des contraintes peu strictes. Ainsi, les contraintes d'accord en nombre sont désactivées si l'analyse ne parvient pas à une solution en les activant. D'autres traits véhiculent des informations concernant les relations grammaticales à mettre en place, notamment le rôle que devra jouer l'élément auquel des propositions prépositionnelles ou relatives seront attachées (voir les traits du PP de la figure 5.4 et de la proposition relative de la figure 5.5. Ces

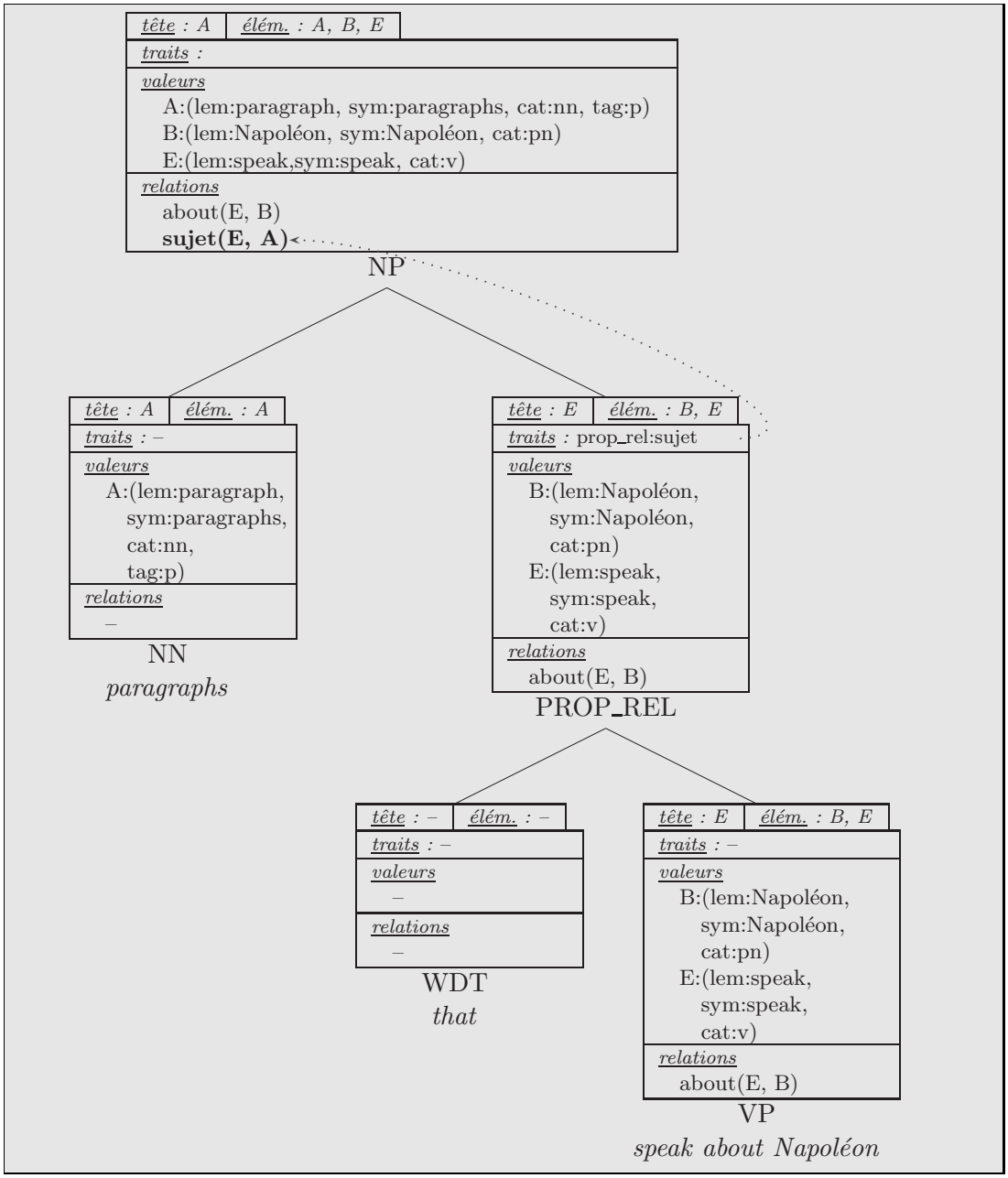


FIG. 5.5 – Analyse syntaxique de *"pragraphs that speak about Napoléon"*.

deux figures illustrent, en deux parties, l'analyse de l'exemple 5.1 – *"Paragraphs that speak about Napoléon"*).

L'ensemble des arbres possibles est généré. Les requêtes n'étant composées que de quelques phrases au maximum, nous pouvons nous permettre cette perte d'efficacité. En l'absence de toute autre information sémantique, l'arbre qui est choisi est sélectionné selon la règle de l'attachement minimal[1] puis, en cas de persistance d'une ambiguïté, par celle de l'association droite[2]. La section 5.3.6 donne les détails concernant la désa-

[1]Cette règle stipule qu'en cas d'ambiguïté, les constructions arborescentes possédant le moins de nœuds doivent être préférées. Cette idée est illustrée à la figure 5.6, page 125, avec la phrase *"Les gendarmes interpellent un conducteur en état d'ivresse"*.

[2]Cette règle s'applique lorsque plusieurs constructions ont le même nombre de nœuds, et précise

mbiguïsation au niveau sémantique.

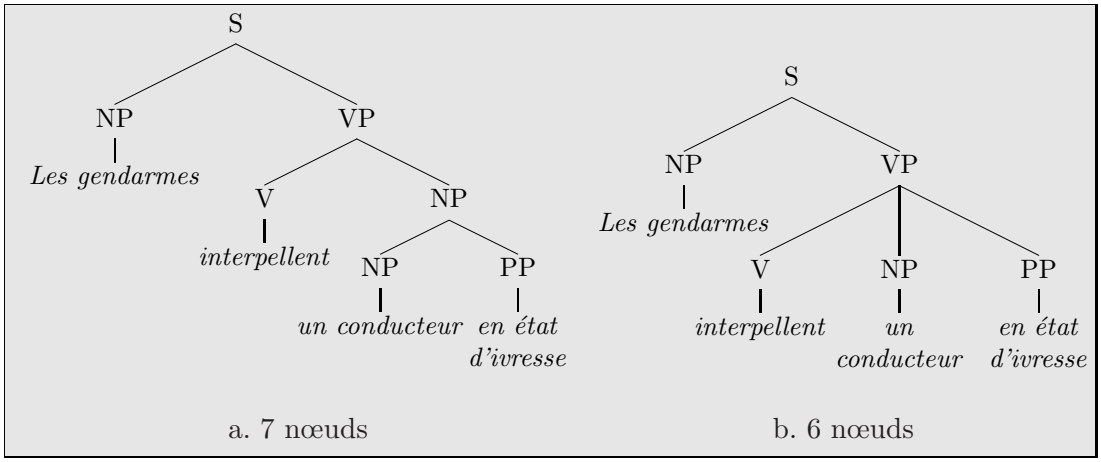


FIG. 5.6 – Règle de l'attachement minimal.

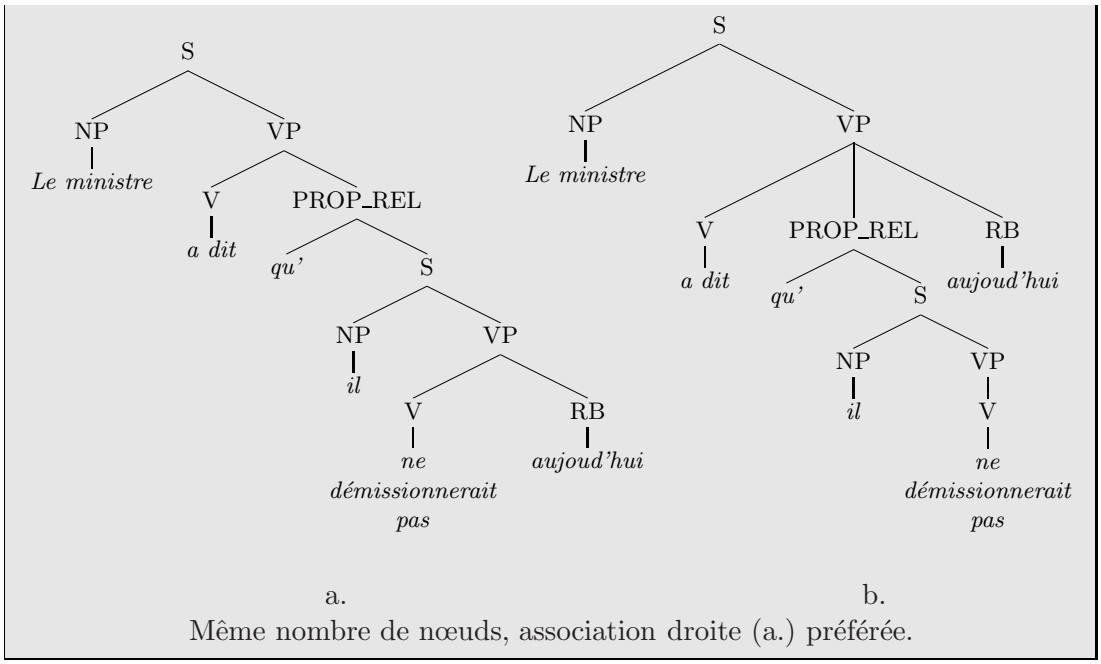


FIG. 5.7 – Règle de l'association droite.

5.2.4.2 L'attribution des relations grammaticales

Une représentation de base est associée à chaque mot de l'énoncé, en fonction de leur catégorie grammaticale, déterminée par TreeTagger. A chaque application d'une règle syntaxique, les représentations des composants concernés sont fusionnées et une ou plusieurs relations grammaticales sont ajoutées (voir les figures 5.4 et 5.5). Le nom de ces relations dépendent des constituants en présence : les liens avec des PP prennent le nom de la préposition concernée (*of*, *in*, *about*...) ; à la voix active, le sujet est avant le verbe, l'objet direct après. A la voix passive, c'est l'inverse. Pour les propositions relatives, le lien dépend de la forme de cette proposition (phrase entière ou VP) et du pronom relatif.

que les attachements de constituants doivent se faire avec le constituant le plus proche (le plus à droite). Voir la figure 5.7, page 125, pour la phrase *"Le ministre a dit qu'il ne démissionnerait pas aujourd'hui"*.

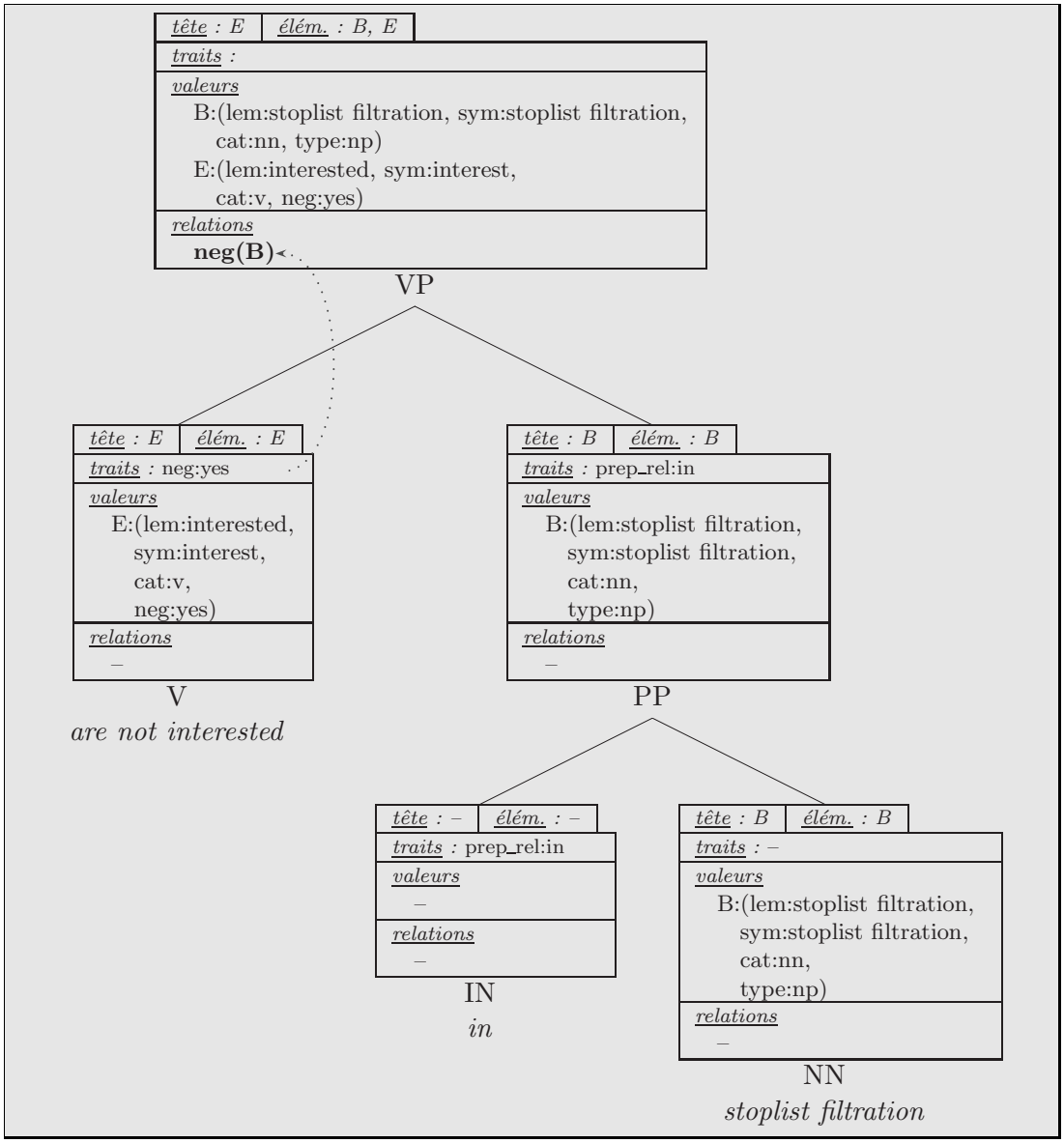


FIG. 5.8 – Traitement syntaxique de la négation, analyse de *"we are not interested in stoplist filtration"*.

5.2.4.3 La négation

En pratique, nous n'utilisons la négation que pour déterminer les termes qui ne doivent pas apparaître dans les documents pour que ceux-ci soient pertinents par rapport à la requête. Ceci correspond au signe '-' de NEXI[1]. Les indicateurs de négation sont en général liés aux verbes, alors que les termes d'une requête sont plutôt dans les syntagmes nominaux. Nous propageons donc la négation du verbe vers ses compléments. Ceci a souvent peu de valeur sémantique, mais nous verrons à la section 5.4.2.5 comment cette approche nous permet de traiter les cas qui nous intéressent. Un exemple de traitement de la négation est donné à la figure 5.8 pour l'exemple suivant :

(5.5) We are not interested in stoplist filtration.[2]

[1] Voir la section 2.4.2.2, page 31.
[2] Extrait du topic 187, INEX 2004.

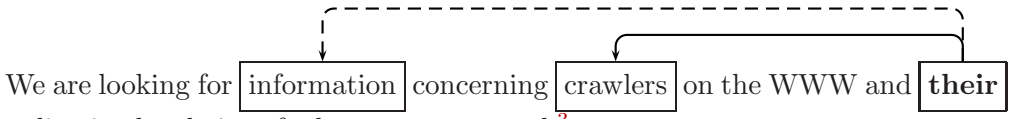
5.2.4.4 Les anaphores

Nous avons insisté sur l'importance des phénomènes de l'anaphore et de l'ellipse pour les interfaces d'interrogation des bases de données[1]. Dans notre cas, le système ne prévoyant pas le dialogue avec l'humain, les problèmes qu'ils posent sont moins prégnants. Les ellipses sont très rares dans les requêtes, et concernent le contenu ; les résoudre n'apporterait que peu d'avantages, d'autant plus que ce serait fait avec beaucoup d'imprécision. Nous ne nous en occupons donc pas.

anaphore

En revanche, les anaphores revêtent une importance un peu supérieure, puisqu'elles instaurent des relations entre les éléments ou entre les termes qui nous intéressent particulièrement :

(5.6) [2]



(5.7) [3]

Dans la requête 5.6, l'adjectif possessif *their* introduit une relation entre *"title"* et *"articles"*. Dans ce cas, il est même possible de nommer cette relation (title *of* article). De la même façon, nous avons la relation *of* entre *"policy"* et *"crawlers"* dans l'exemple 5.7. Ces informations ont beaucoup d'intérêt pour l'analyse sémantique et la construction de la requête.

Notre politique aux sujets des anaphores est de générer tous les attachements possibles (en tenant compte des restrictions de nombre), comme le montrent les flèches des exemples, dans lesquels tous les termes au pluriel sont proposés. Pour déterminer une préférence, l'algorithme naïf, décrit à la section 3.4.3.1, est utilisé. Il conduit à proposer en premier lieu les attachements indiqués en traits continus, ce qui est incorrect dans le premier cas.

Nous avons ainsi de nouvelles relations qui, exceptée leur origine, n'ont pas de différence avec les relations prépositionnelles classiques (voir l'illustration de la figure 5.9, avec deux solutions possibles). La couche sémantique les traitera donc de la même façon (elle permettra notamment de rétablir l'attachement avec *"articles"* dans l'exemple 5.6, pour ne conserver que la forme b. de la figure 5.9).

5.2.4.5 Robustesse

Pour avoir l'assurance de renvoyer un résultat quel que soit l'énoncé, nous effectuons une seconde analyse n'analysant que les syntagmes nominaux, en cas d'échec de la première (celle-ci se plaçant au niveau de la phrase). Ceci permet d'une part d'analyser des énoncés qui ne sont pas des phrases complètes, comme dans l'exemple 5.8, mais aussi de collecter autant d'informations que possible concernant les requêtes pour lesquelles l'analyse complète échoue, comme c'est le cas pour la phrase 5.9.

(5.8) Vita and details about authors that are phD students.[4]

[1] Section 3.7, page 83.
[2] Topic 267, INEX 2005.
[3] Topic 195, INEX 2004.
[4] Topic 153, INEX 2004

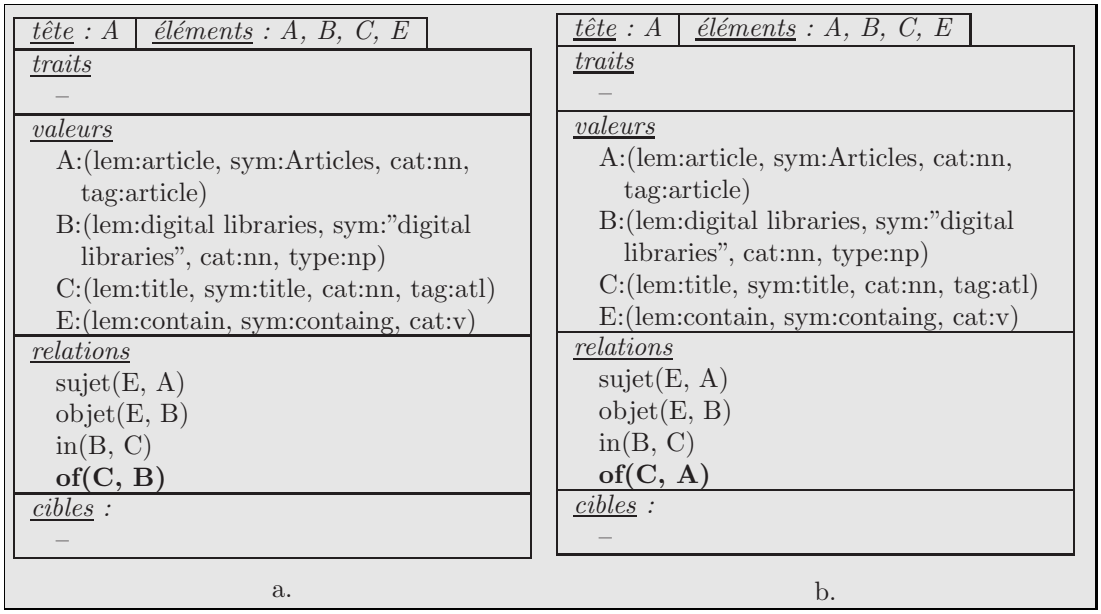


FIG. 5.9 – Traitement de l'anaphore, pour la phrase *"Articles containing "digital libraries" in their title"* (exemple 5.6).

(5.9) The basic idea is to know fields or problems where this integration could be very useful to overcome problems that with both technologies separately couldn't be solved and how they have been solved with that integration, i.e. the solution given integrating.[1]

Notons qu'effectuer une analyse des syntagmes nominaux *avant* de lancer la reconnaissance au niveau de la phrase aurait été source d'erreurs, comme nous l'avons noté au chapitre 3 [2].

En pratique, lorsque la première analyse, recherchant des phrases, ne parvient à englober qu'une partie, voire aucun mot, de l'énoncé, les segments non analysés sont repris et, progressivement, les règles de reconnaissance des syntagmes nominaux leur sont appliquées. Ce processus est illustré par la figure 5.10.

5.3 Règles sémantiques

Pour traduire la suite de référents et de relations diverses décrites à la section précédente en une représentation faisant du sens *du point de vue de la recherche d'information semi-structurée*, il est nécessaire de mettre en place un certain nombre de règles de transformation. Celles-ci peuvent être de plusieurs types :

- Les règles fixes, indépendantes en tous points du type de corpus utilisé. Elles concernent surtout toutes les constructions typiques d'une requête, les relations entre les éléments (quels que soient ces éléments), les relations entre les éléments et le texte (section 5.3.3).
- Les règles propres à un type de structure particulier, et donc dépendantes de cette structure. Il s'agit en général de formes de phrases qui font allusion de façon implicite à des éléments structurant (section 5.3.4).

[1] Extrait du topic 171, INEX 2004
[2] Section 3.3.1.1, page 51.

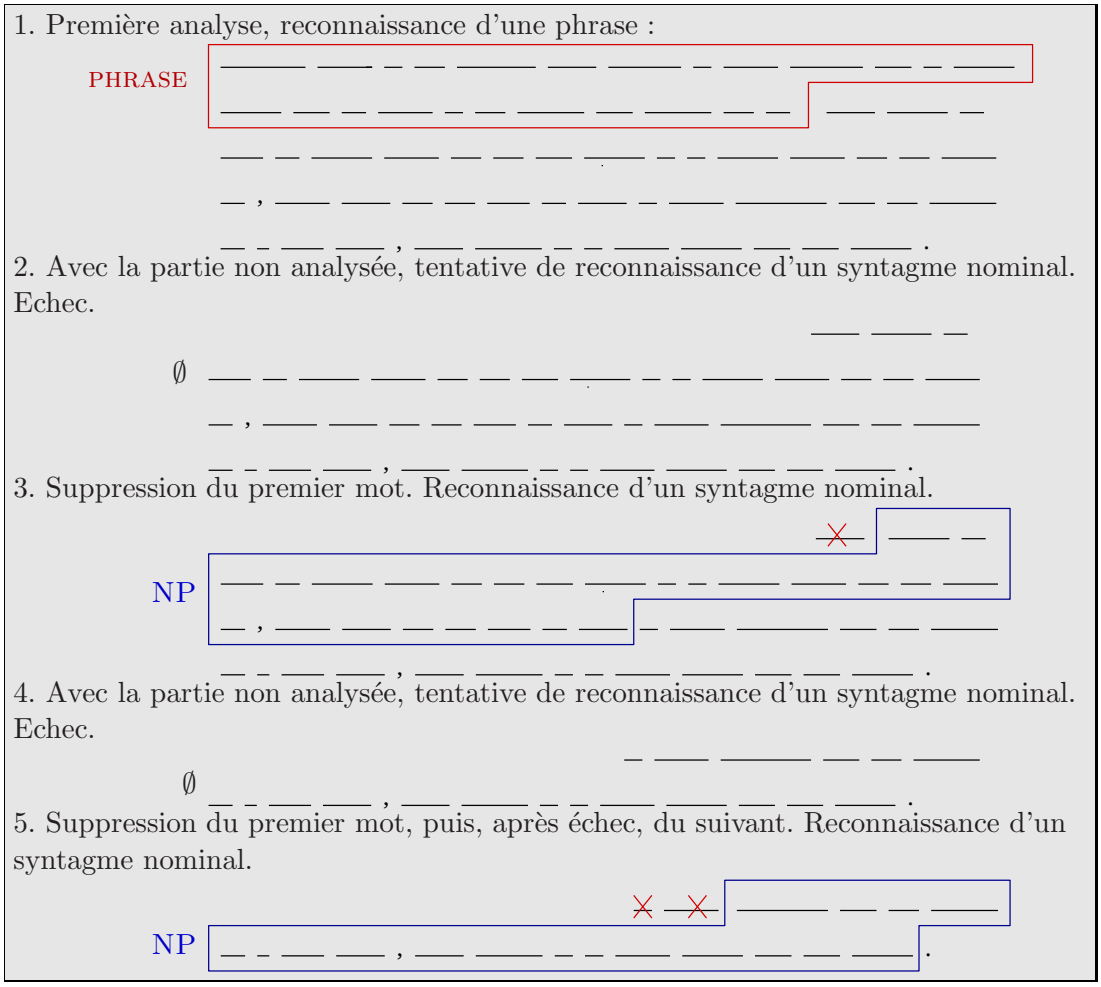


FIG. 5.10 – Analyse progressive des énoncés complexes.

– Les règles propres à un sujet ou un domaine particulier abordé par la requête et/ou par le corpus. Pour des raisons déjà exprimées, nous avons choisi de ne pas utiliser ce type de règles.

Avant cela il faut déterminer quels mots ou groupes de mots constituent une référence explicite à la structure. Ceci est bien entendu dépendant de la DTD (ou des DTD) de la collection. La section 5.3.1 détaille la procédure suivie.

5.3.1 Références explicites à la structure

Nous regroupons désormais sous le vocable "référence explicite" ce que nous avons appelé références "explicites" et "quasi-explicites" à la section 5.1.1, c'est-à-dire l'emploi de mots précis pour désigner des éléments structurels.

La plupart des noms de balises dans les DTD sont des abréviations plus ou moins obscures. Pourtant, l'utilisateur d'une interface en langage naturel est en droit d'attendre que celle-ci lui donne la possibilité d'employer des termes de la langue courante pour y faire allusion. De plus, cet utilisateur ignore probablement les noms des balises utilisés dans la DTD. Enfin, il souhaite peut-être interroger plusieurs collections, obéissant à des DTD différentes. Pour toutes ces raisons, il est nécessaire de distinguer les mots ou les groupes de mots qui désignent des balises du corpus dans une requête.

```
article : article, paper, publication, research, work, document, survey.
    atl : title, article title.
     bb : citation, bibliographic reference, bibliographic entry.
    fgc : figure caption, caption of figure.
      p : paragraph.
     yr : year.
      * : fragment, component, description, explanation, detail, information,
          discussion, tag, text, ...
```

FIG. 5.11 – Extrait du dictionnaire des synonymes utilisé pour la collection INEX 2005 (articles scientifiques).

Détecter les références explicites à des éléments structurels pose *a priori* peu de problèmes. Hélas, les balises et les concepts qu'elles représentent sont, comme toute production du langage, soumis aux problèmes de synonymie et de polysémie dont nous avons déjà tant parlé[1]. Nous verrons qu'une autre relation s'ajoute dans notre cas : la *conceptémie*.

5.3.1.1 Synonymie

Que les noms de balises soient de "vrais" mots de la langue considérée ou des chaînes de caractères apparemment dénués de sens, le problème est somme toute le même : il existe plusieurs façons de désigner le concept qu'une balise représente.

Un dictionnaire reliant un nom de balise à un ou plusieurs termes correspondant dans la langue considérée est donc indispensable. Ce dictionnaire de synonymes est très dépendant de la structure, non seulement à cause des noms de balises fantaisistes, mais également parce que deux termes *a priori* éloignés peuvent être synonymes dans un contexte très précis. Ainsi, pour les articles scientifiques de la collection INEX 2005, les termes *"papier"*, *"article"* et *"publication"* sont synonymes (et désignent la balise `'article'`), ce qui n'est pas le cas dans le langage courant.

La figure 5.11 donne un extrait du dictionnaire des synonymes valable pour des articles scientifiques. Notons que certains synonymes sont des groupes de mots, et que parfois un de ces mots est également le synonyme d'une autre balise (voir *"article"* dans *"article title"*, désignant la balise `'atl'`). Pour limiter les entrées manuelles de synonymes, nous avons mis en place un dispositif de propagation automatique, comme le montre la figure 5.12[2].

5.3.1.2 Polysémie et conceptémie

La polysémie* a deux effets négatifs distincts sur la reconnaissance des références explicites à la structure dans la requête. Le premier produit une ambiguïté entre plusieurs éléments structurels, et le second entre un élément structurel et du texte classique.

[1] Voir la section 3.4.4, page 72.

[2] Notons que ce mécanisme de propagation généralisée peut provoquer des formes très incorrectes, mais qui n'ont aucune chance d'apparaître dans une phrase. De plus, pour éviter une expansion contre-productive, voire des bouclages, nous nous limitons à une seule itération, c'est-à-dire que les synonymes de synonymes ne seront pas propagés.

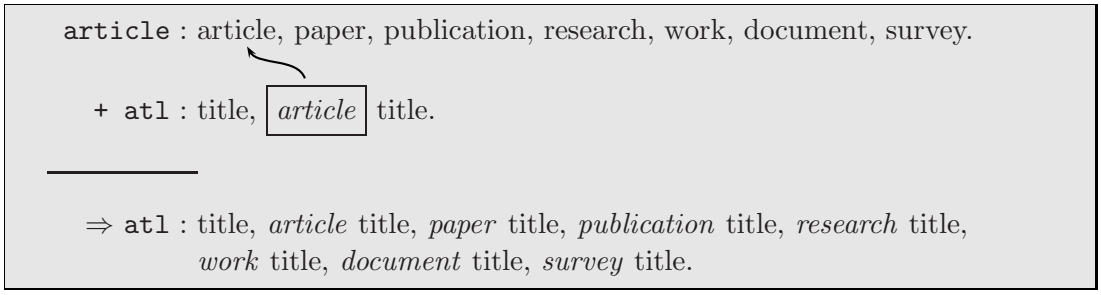


FIG. 5.12 – Propagation des relations de synonymie.

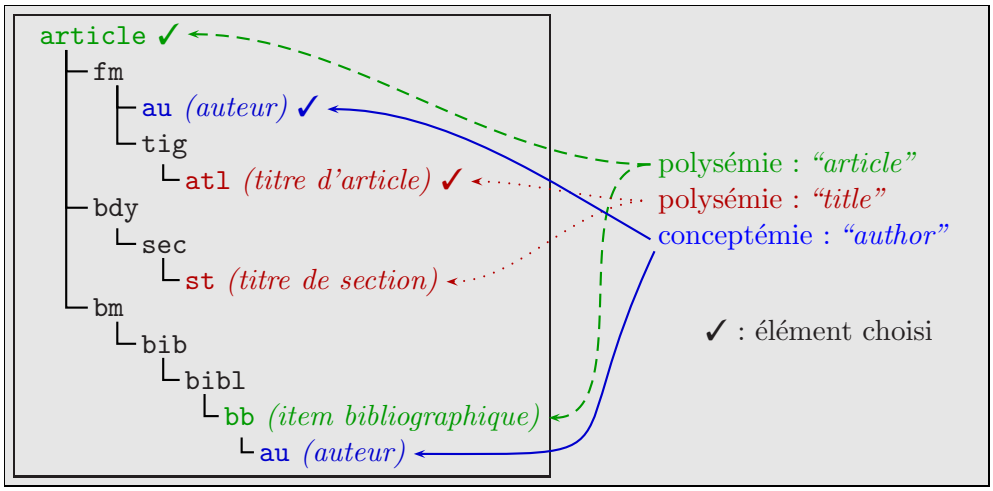


FIG. 5.13 – Polysémie et conceptémie.

Structure contre structure. Il arrive qu'un même terme de la langue naturelle désigne deux balises de noms différents. C'est par exemple le cas pour *"title"*, qui peut, selon le contexte, indiquer un titre d'article (`'atl'`) ou un titre de section (`'st'`). Le mot *"article"*, qui caractérise en général un `'article'`, peut être utilisé pour symboliser un élément bibliographique `'bb'` (comme dans *"un auteur qui cite un* article *de..."*).

Ce phénomène est "enrichi" par un second, spécifique aux documents structurés : la *conceptémie* [242]. Il désigne l'utilisation d'un même nom de balise pour représenter un même concept[1], mais en plusieurs endroits distincts de la structure. Ainsi, la balise `'au'`, caractérisant un auteur, est utilisée dans la collection INEX 2005 pour désigner l'auteur de l'article, mais aussi l'auteur d'une publication citée dans une référence bibliographique.

conceptémie

Ces deux problèmes sont illustrés à la figure 5.13 avec la DTD d'INEX 2005.

Dans les deux cas, la méthode employée pour désambiguïser est la même : par défaut, la balise choisie est *la plus proche* du début de l'article, c'est-à-dire, dans la figure 5.13, la plus haute dans la hiérarchie indiquée par la DTD. Nous considèrons en effet que cet élément est le plus susceptible d'être désigné sans autre indication. Le contexte permettra de déterminer si une autre possibilité doit être retenue, grâce aux règles sémantiques. Par exemple, si l'utilisateur écrit *"title"*, nous choisirons le titre d'article (`'atl'`) par défaut. S'il précise *"section title"* ou *"a section with a title..."*, il s'agira bien sûr du titre de section (`'st'`). En cas d'incertitude (ordre des balises non indiqué par la DTD), on préfère l'élément dont le chemin d'accès est le plus court ; ainsi, parmi

[1] Il n'y a donc ni synonymie ni polysémie.

les chemins d'accès à la balise `'au'` (`/article/fm/au` et `/article/bm/bib/bibl/au`), le premier est préféré car plus court.

Structure contre contenu. Tout terme désignant un élément structurel de la DTD est également susceptible d'être utilisé comme indication de contenu. Ainsi, le mot *"document"*, qui est souvent utilisé dans les requêtes comme désignant un document XML (un article par exemple), est parfois intégré au contenu, comme dans la requête :

(5.10) I'm searching abstracts that deal with multimedia *document* models.[1]

De même, le mot *"text"*, en général synonyme d'*élément* (*"Find texts about..."*), peut faillir à cette règle :

(5.11) Find documents about spatial or text databases.[2]

La première réponse à ce problème est de ne considérer comme élément structurel potentiel que les termes désignant un nom de balise qui sont la *tête** du groupe nominal qui les contient. Ainsi, dans les exemples précédents, les éléments de tête sont respectivement *"models"* (pour *"multimedia document models"*) et *"databases"* (pour *"spatial ou text databases"*), qui ne sont pas répertoriés comme synonymes de balises. L'ensemble du syntagme nominal est alors considéré comme du contenu.

La seconde technique de désambiguïsation, appliquée lorsque le terme est la tête du NP, consiste à envisager les deux possibilités (élément structurel ou contenu) dans un premier temps, et à attendre le verdict des règles sémantiques. Nous détaillerons le principe de ces règles dans les sections suivantes, mais nous pouvons donner un exemple clair concernant le mot *"document"* :

(5.12) We are looking for *documents* about multimedia *documents*.

Ici les deux occurrences de *"documents"* sont en position de tête. Pourtant la première représente une balise, car c'est la cible de la requête (matérialisée par l'expression *"We are looking for"*), tandis que la seconde est plutôt du contenu, comme l'indique la construction *"*<élément> *about"*.

Les choses ne sont bien sûr pas toujours aussi simples. Lorsque le doute persiste on fait le choix de l'élément structurel. Heureusement, dans la majorité des cas, ces termes ambigus ont rarement un poids* très discriminant (au sens de la pondération effectuée lors de l'indexation).

5.3.1.3 Représentation des éléments structurels

Lorsqu'un terme est considéré comme un élément structurel, un nouvel attribut est ajouté à sa représentation, sous la forme `tag:T`, où `T` est le nom de la balise qu'il représente.

5.3.2 Représentation des règles

Nos règles sémantiques sont des règles de transformation d'une représentation logique en une autre, plus concise ou contenant des prédicats plus conformes aux besoins d'un moteur de recherche.

[1] D'après le topic 210, INEX 2005.
[2] D'après le topic 161, INEX 2004.

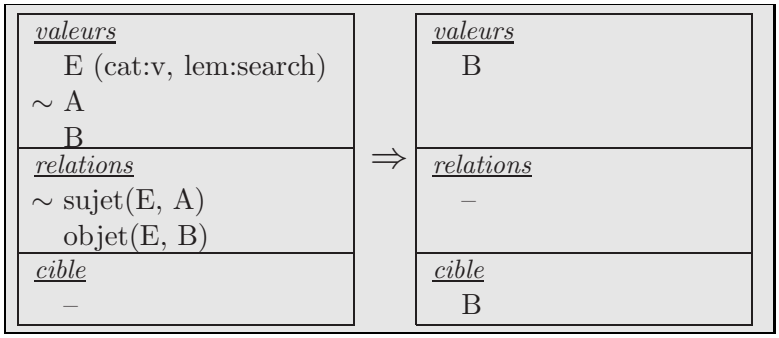


FIG. 5.14 – Exemple de règle sémantique : *to search <objet>*.

En effet, l'analyse syntaxique génère de nombreuses relations linguistiques qu'il faut remplacer par un ensemble d'autres liens, plus facilement transformables en une requête formelle, dont la forme est imposée par le langage final.

Les éléments impliqués dans ces transformations sont :

- les variables et les différents traits qui les caractérisent ;
- les relations entre ces variables ;
- l'indicateur de la cible de la requête.

Nous représentons les règles par deux ensembles qui sont illustrés par l'exemple de la figure 5.14. Le premier ensemble (à gauche) contient les valeurs, relations et cibles nécessaires au déclenchement de la transformation, et le second contient les valeurs, relations et cibles destinées à remplacer celles du premier ensemble. Ainsi, on peut lire la règle de la figure 5.14 de la façon suivante : si le verbe (*"cat:v"*) *"to search"* (*"lem:search"*) est présent dans l'énoncé, et entretient les relations de *sujet* et d'*objet* avec deux autres référents (respectivement A et B), alors le référent B devient la cible de la requête. Les relations sont supprimées et seule la valeur B est conservée. La présence de A et la relation *sujet* ne sont pas indispensables, comme l'indiquent le signe $\sim$[1]. Il est également possible de faire précéder une variable ou une relation du signe $\neg$, pour indiquer qu'elle *ne doit pas* être présente dans la représentation pour que la règle s'applique. Ce dernier signe peut enfin se trouver dans la liste des traits, pour indiquer que tel ou tel trait ne doit pas être présent dans la liste.

5.3.3 Règles fixes

Les règles sémantiques fixes sont celles qui ne dépendent pas du type du corpus interrogé. Elles ne sont pas pour autant valables pour n'importe quel énoncé de la langue, mais sont plutôt orientées vers le traitement des constructions spécifiques des requêtes pour la recherche d'information. Elles gèrent surtout la détection de la cible de la requête et les relations entre les éléments et le contenu, à travers deux types de liens, verbaux d'une part, et prépositionnels d'autre part.

[1] Si une relation n'est pas indispensable au déclenchement de la règle, mais peut y participer malgré tout, il est intéressant de l'indiquer tout de même, surtout lorsqu'elle est supprimée par cette règle : cela permet de "nettoyer" la représentation en ôtant les éléments inutiles.

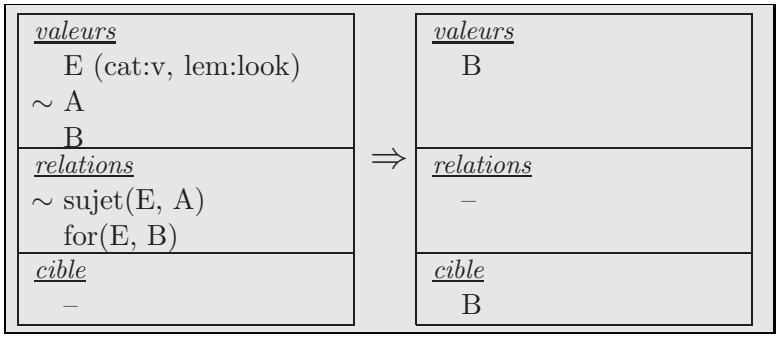


FIG. 5.15 – Exemple de règle sémantique : *to look <for>*.

5.3.3.1 Détection de la cible de la requête

La détection de la cible de la requête est extrêmement simple lorsqu'elle est indiquée explicitement par l'utilisateur. Il utilise alors des constructions comme *"We are looking for..."*, *"Find..."*, *"I want..."*. La règle de la figure 5.14, par exemple, traite l'expression *"to search* <objet>*"*; celle de la figure 5.15, *"to look* <for>*"*. Le sujet est théoriquement présent, mais la règle reste valable en son absence (au cas où l'analyse syntaxique serait imparfaite par exemple), comme l'indique le signe ∼.

En pratique, nous distinguons deux cas : celui où la cible est un élément structurel, et celui où il s'agit de texte simple. Quand il s'agit de texte, un nouvel élément structurel est ajouté, de type indéterminé (`'*'`), contenant le texte[1]. Ainsi la règle 5.15 est en fait doublée, pour mener aux deux règles de la figure 5.16.

La désignation de la cible de façon verbale obéit à un cadre tout simple qui permet de généraliser ces règles. L'identification du verbe concerné et de la relation grammaticale qu'il peut entretenir avec la cible suffit pour générer toutes les composantes de ces règles. Ceci est rendu possible par l'utilisation du prédicat

verbe_cible(VERBE, RELATION).

qui permet d'entrer ou de mettre à jour des règles de façon beaucoup plus simple. La figure 5.17 liste quelques-uns des verbes concernés, et donne la nouvelle règle générique de détection de la cible pour un élément structurel.

Lorsque la cible n'est pas indiquée de façon explicite, le choix se fait de manière différente. Si un ou plusieurs éléments structurels sont présents dans l'énoncé, celui qui apparaît le premier en position de tête d'un groupe nominal est choisi comme cible[2]. En l'absence d'élément structurel répondant à ce critère, un nouvel élément est créé, de type `'*'`, qui contient tous les autres (éléments structurels et textuels) et qui devient la cible.

5.3.3.2 Autres liens verbaux

Certaines relations verbales permettent de repérer des liens entre les différents éléments d'une requête. Les plus évidentes sont les formules comme *"to speak about..."*, *"to*

[1] Par exemple : *"to look for a definition"* est traduit par un élément cible contenant le mot *"définition"* (voir figure 5.16b.).

[2] Ceci permet notamment de traiter des requêtes composées d'un simple groupe nominal, comme *"Paragraphs about..."*.

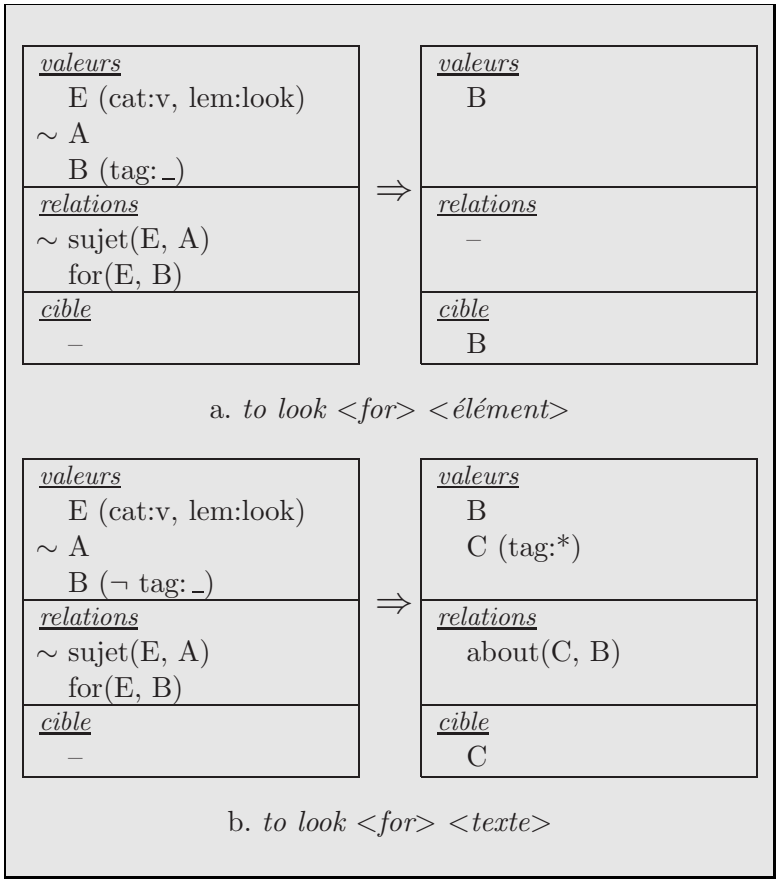


FIG. 5.16 – Interprétation de la cible d'une requête : *to look <for>* suivi d'un élément structurel (a.) ou de texte simple (b.).

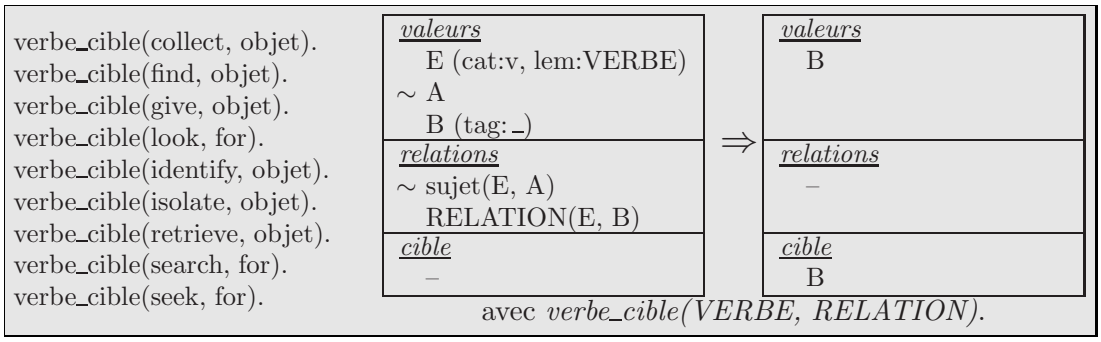


FIG. 5.17 – Quelques prédicats concernant les verbes désignant la cible de la requête, et règle générique les utilisant.

describe", qui indiquent, du point de vue de la recherche d'information, une inclusion d'un élément dans un autre[1]. D'autres peuvent indiquer des relations de précédence, ou d'autres liens. La précision nécessaire dépend des relations que le langage final est capable de traiter.

La figure 5.18 montre un exemple de règle transformant une relation verbale grammaticale (ici *"<sujet> deal <with>"*) en relation sémantique (*about*).

[1]Si un élément A *décrit* un élément B, on en conclut que, dans le document, A doit *contenir* B, moyennant tous les traitements possibles concernant A et B, propres à chaque système.

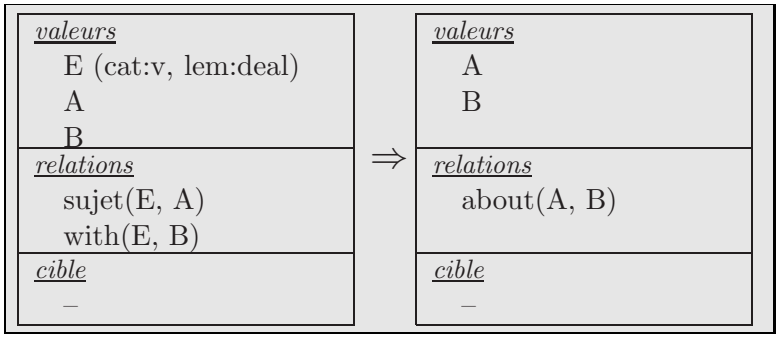


FIG. 5.18 – Règle sémantique : *<sujet> deal <with>*.

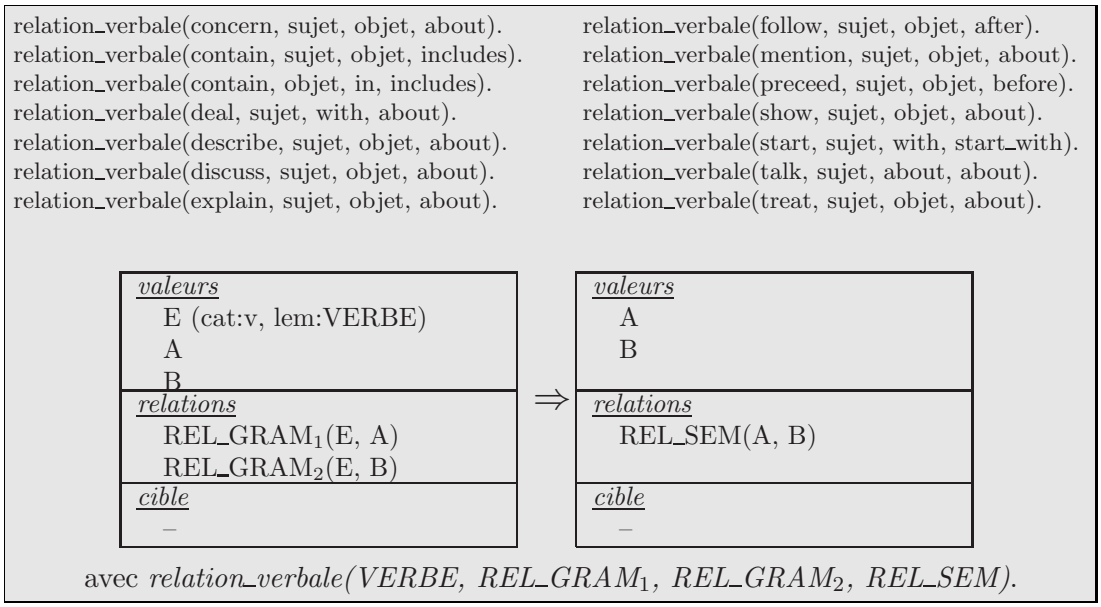


FIG. 5.19 – Quelques prédicats concernant les relations verbales, et règle générique les utilisant.

Encore une fois, il est possible de simplifier la gestion de ces règles en mettant en place un cadre pour ces relations verbales. Le prédicat

relation_verbale(VERBE, REL_GRAM$_1$, REL_GRAM$_2$, REL_SEM).

doit être lu ainsi : si le verbe *VERBE* entretient la relation *REL_GRAM*$_1$ avec un élément *A* et la relation *REL_GRAM*$_2$ avec un élément *B*, alors la relation *REL_SEM* entre *A* et *B* est inférée. Le figure 5.19 propose des exemples de telles relations, ainsi que la règle sémantique générique qui en découle.

5.3.3.3 Liens prépositionnels

Tout comme les constructions verbales vues à la section précédente, les prépositions indiquent certaines relations particulières entre les éléments de la requête. Par exemple, *"<élément$_1$> with <élément$_2$>"* ou *"<élément$_1$> in <élément$_2$>"* signalent des liens hiérarchiques entre deux éléments structurels[1] ; *"<élément> about <texte>"* annonce que l'élément doit traiter le sujet évoqué par le texte.

[1] Comme dans *"a section* with *a paragraph"* ou *"a paragraph* in *a section"*.

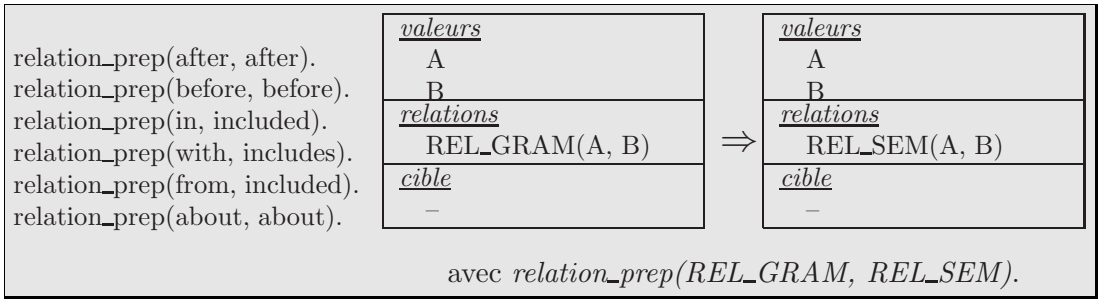


FIG. 5.20 – Quelques prédicats concernant les relations prépositionnelles, et règle générique les utilisant.

Là encore, des règles sémantiques sont mises en place, et là encore, une généralisation est possible à l'aide d'une simple liste de prédicats :

relation_prep(REL_GRAM, REL_SEM).

Cette procédure est illustrée à la figure 5.20. Certains des liens obtenus, comme ceux indiquant l'inclusion entre deux éléments, servent directement à la création de la structure de la requête. Pour les autres, il est parfois difficile de proposer un traitement fin. Pourtant, ces relations nous indiquent que les termes reliés sont indissociables, ce qui nous sera utile pour générer la requête finale.

5.3.3.4 Exemples d'application

Nous illustrons, avec les figures 5.21, 5.22 et 5.23, certaines des règles que nous venons de décrire, à travers la transformation de la représentation de la requête :

(5.13) We are searching paragraphs dealing with version management in articles containing a paragraph about object databases[1].

De nombreux champs de la représentation n'étant utiles que pour la progression de l'analyse syntaxique, nous ne conservons, pour éviter une surcharge inutile et pour améliorer la clarté du propos, que les trois étages (valeurs, relations et cible) qui participent aux règles et à la représentation sémantique.

La première règle appliquée concerne la détection de la cible, avec le traitement de l'expression *"We are searching paragraphs"* (figure 5.21). La seconde prend en compte la relation entre *"paragraphs"* et *"version management"* (figure 5.22). Les autres règles permettent d'aboutir à une représentation dont les prédicats sont plus intéressants que les relations grammaticales du départ (figure 5.23).

Remarquons que la représentation grammaticale du départ (figure 5.21) est celle qui correspond effectivement à la bonne façon de comprendre la requête, mais qu'il ne s'agit pas de la première possibilité générée par l'analyse syntaxique. En effet, la loi de l'attachement minimal fait en sorte que la relation *in(C, D)* est proposée avant *in(B, D)* (en italiques dans la figure). Cette phrase comporte par ailleurs énormément d'ambiguïtés. Nous verrons à la section 5.3.6 comment nous parvenons à sélectionner finalement la représentation illustrée ici.

[1] Topic 130, INEX 2004.

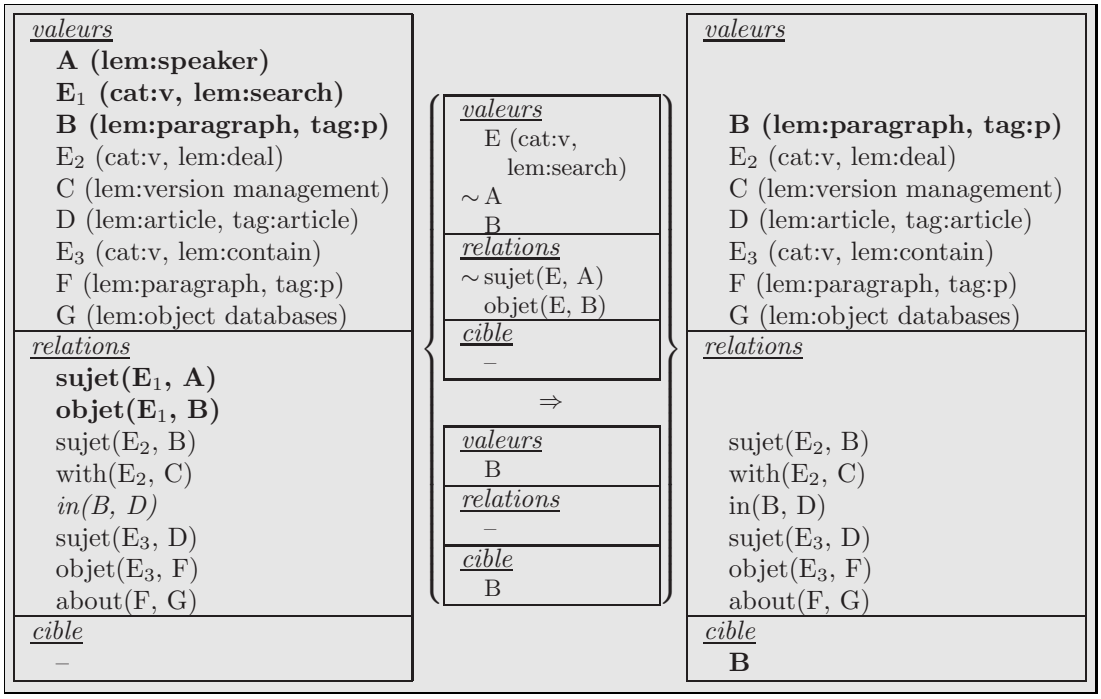


FIG. 5.21 – Détection de la cible dans une requête.

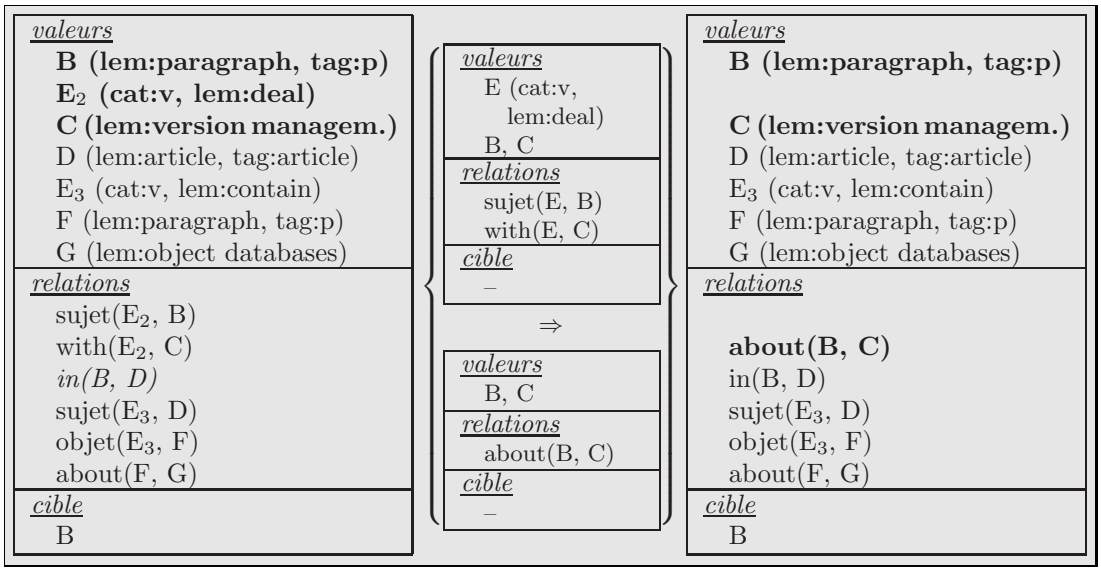


FIG. 5.22 – Application de la règle sémantique <*sujet*> *deal* <*with*>.

5.3.4 Règles dépendant de la structure

Les règles que nous traitons dans cette section sont celles qui dépendent de la structure des documents interrogés. Il est donc potentiellement nécessaire de les changer lorsque l'on souhaite adapter le système à une nouvelle collection, même si certaines d'entre elles sont très courantes.

Remarquons que l'utilisateur ne sait pas nécessairement qu'il fait une référence à la structure lorsque ces règles sont déclenchées. Ainsi, nul n'empêche quelqu'un de demander un *"article écrit par"* quelqu'un, que l'auteur du document soit explicitement balisé ou non.

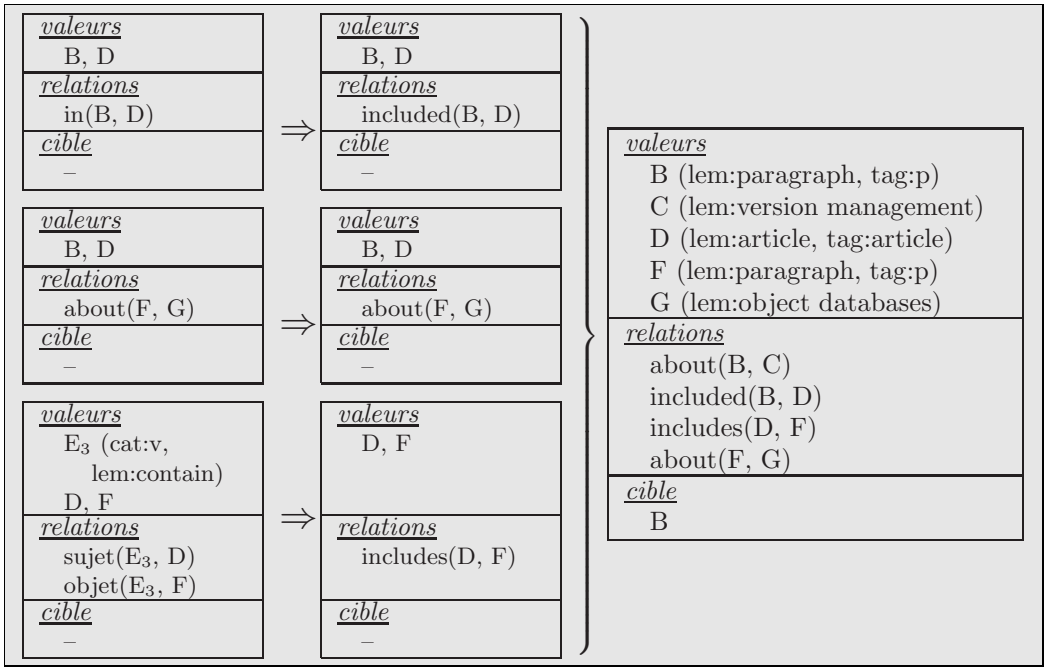


FIG. 5.23 – Exemples d'application de diverses règles sémantiques.

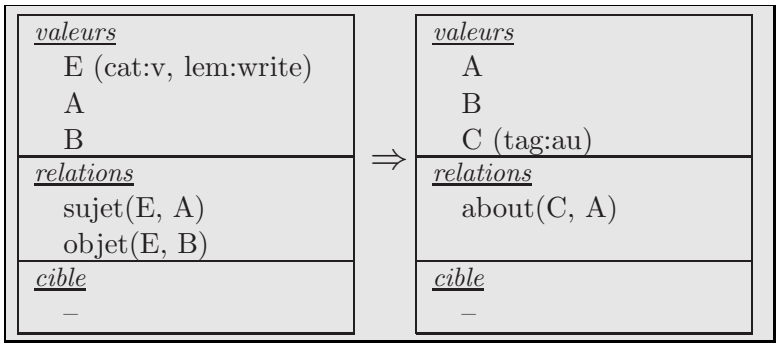


FIG. 5.24 – Exemple de règle sémantique : <*sujet*> *write* <*objet*>.

5.3.4.1 Liens verbaux

Nous appelons *liens verbaux* les constructions impliquant un verbe et la structure des documents d'une façon ou d'une autre. Ces liens ont un caractère beaucoup moins générique que ceux exprimés par les règles vues précédemment.

Nous en énumérons quelques-uns s'appliquant au corpus INEX 2005 des articles scientifiques d'IEEE, décrit au chapitre 2 [1], en précisant leur raison d'être.

"<sujet> write <objet>". Figure 5.24. Des constructions telles que *"an article written by Somebody"* (où *"article"* est un élément structurel, objet du verbe *"to write"* – voix passive – et où *"Somebody"* est le sujet réel) permettent d'introduire une balise `'au'` (*auteur*) contenant *"Somebody"*.

[1] Section 2.5.4.1, page 39.

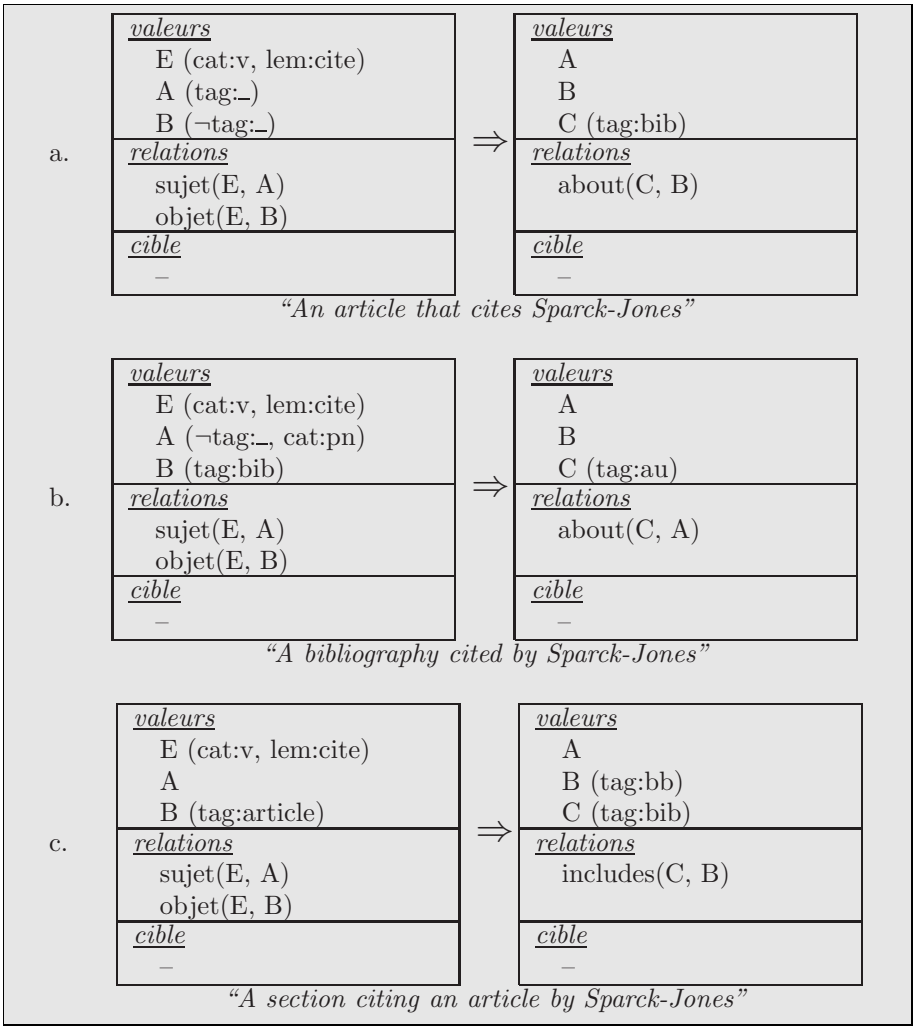


FIG. 5.25 – Exemples de règles sémantiques : <*sujet*> *cite* <*objet*>.

"<sujet> cite <objet>". Figure 5.25. Exemple : *"an article that cites Somebody"* (*"article"* est sujet, *"Somebody"* objet du verbe) est une allusion à la partie des références bibliographiques (`'bib'`). Il est possible de distinguer plusieurs tournures possibles. Comme l'indique la figure, on peut citer un *auteur* (a) ou un *article* (c), et un élément bibliographique peut être cité par un *article*, une *section* (c), mais aussi un *auteur* (b), etc. La catégorie grammaticale peut permettre de distinguer un nom d'auteur (ou au moins un nom propre).

"publish <in> <année>". Figure 5.26. Exemple : *"an article published in 2000"* (*"article"* est objet, *"2000"* est préalablement reconnu comme un indicateur d'année) provoque l'introduction d'une balise `'yr'` (year). Lorsque le texte lié au verbe par la préposition *"in"* n'est pas une année mais du texte normal, on remplace l'élément `'yr'` par l'élément `'ti'` (nom du journal). Notons qu'ici c'est le prédicat *contains* qui est utilisé, et non *about* comme le plus souvent. En effet, nous utilisons *about* comme un prédicat vague, autorisant les manipulations comme la lemmatisation* ou l'expansion. Ici, *contains* indique que c'est la chaîne de caractères que nous recherchons, pas un concept à signifiant variable.

Remarquons que l'avantage d'avoir effectué une analyse syntaxique avant d'appli-

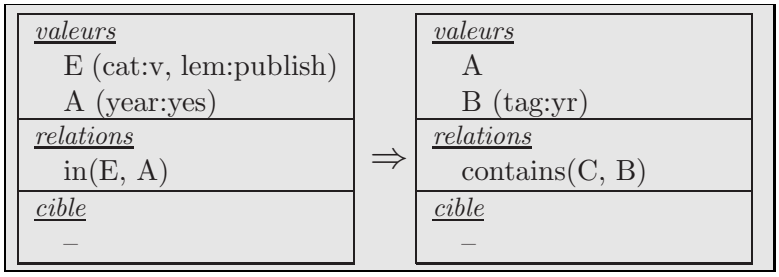


FIG. 5.26 – Exemple de règle sémantique : *publish <in> <année>*.

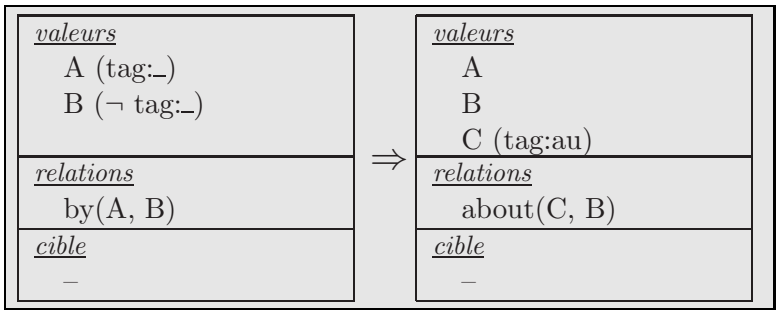


FIG. 5.27 – Exemple de règle sémantique : *<élément> by <texte>*.

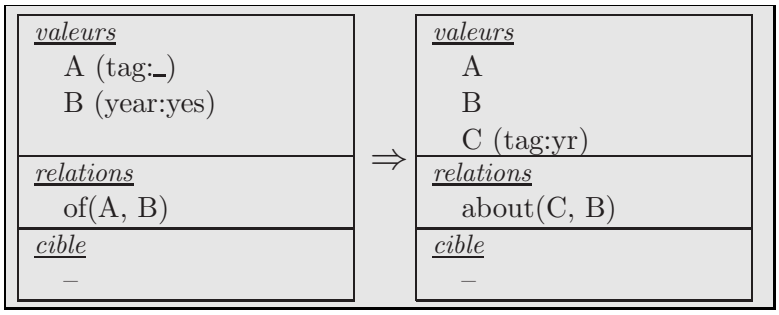


FIG. 5.28 – Exemple de règle sémantique : *<élément> of <année>*.

quer des règles sémantiques apparaît pleinement ici, puisque nous pouvons nous affranchir de la forme initiale des requêtes. Ainsi, les constructions *"an article citing Somebody"*, *"an article that cites Somebody"*, *"an article in which Somebody is cited"*... sont traitées indifféremment par la même règle sémantique.

5.3.4.2 Liens prépositionnels

Les règles basées sur les liens prépositionnels sont basées sur le même principe que les précédentes : une relation grammaticale spécifique induit l'introduction d'un nouvel élément structurel et de nouvelles relations, comme le montrent les deux exemples suivants, toujours concernant la collection INEX 2005.

"<élément> by <texte>". Figure 5.27. Exemple : *"article by Somebody"*.

"<élément> of <année>". Figure 5.28. Exemple : *"article of 2000"*.

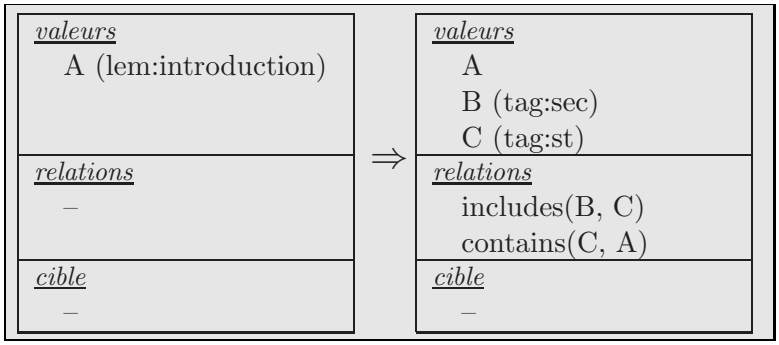


FIG. 5.29 – Exemple de règle sémantique : *introduction*.

5.3.4.3 Autres règles

D'autres règles, encore plus spécifiques, peuvent être imaginées grâce au cadre mis en place. Nous en avons très peu pour la collection INEX 2005, mais elles peuvent prendre beaucoup d'importance pour d'autres types de documents. Pourtant, ce sont des règles plus *ad hoc*, et donc plus difficiles à adapter si besoin est.

Dans notre cas, nous avons créé trois règles prenant en compte les termes "summary", "introduction" et "conclusion", qui ont la particularité de faire référence à la structure de façon indirecte, puisqu'il s'agit de sections particulières (dans les volumes IEEE pour "summary", dans les articles en général pour les deux autres) mais non distinguées par le balisage. C'est le titre de ces sections (`'st'` pour *section title*) qui contient le mot discriminant.

Nous donnons à la figure 5.29 l'exemple de l'introduction, les deux autres règles sont construites de la même façon.

5.3.5 Etablissement d'un modèle de la collection

Il est possible de construire un modèle de collection résumant les relations *linguistiques* susceptibles d'exister entre les différents éléments de la DTD. Un tel modèle permet de faciliter fortement la conception de la plupart (mais non la totalité) des règles sémantiques dépendant de la structure (voire de générer automatiquement ces règles), mais offre également un moyen de détection des ambiguïtés susceptibles de dégrader la qualité des requêtes.

Une utilité essentielle de ce type de modèle est de permettre d'identifier les constructions qui risquent de provoquer des ambiguïtés.

5.3.5.1 Le modèle

Le modèle, pour la collection INEX 2005, est présenté en partie à la figure 5.30 par un graphe orienté. Il est très simple à concevoir et à comprendre, et les conventions en sont les suivantes :

- Un nœud représente une balise de la DTD. Entre parenthèses est indiquée, si elle existe, la catégorie générale du contenu de la balise (par exemple, nom propre – *pn* – pour un auteur – `'au'`).
- Un arc représente une relation linguistique entre deux balises.
  - Quand la relation est verbale, le verbe est indiqué puis, avant et après, entre

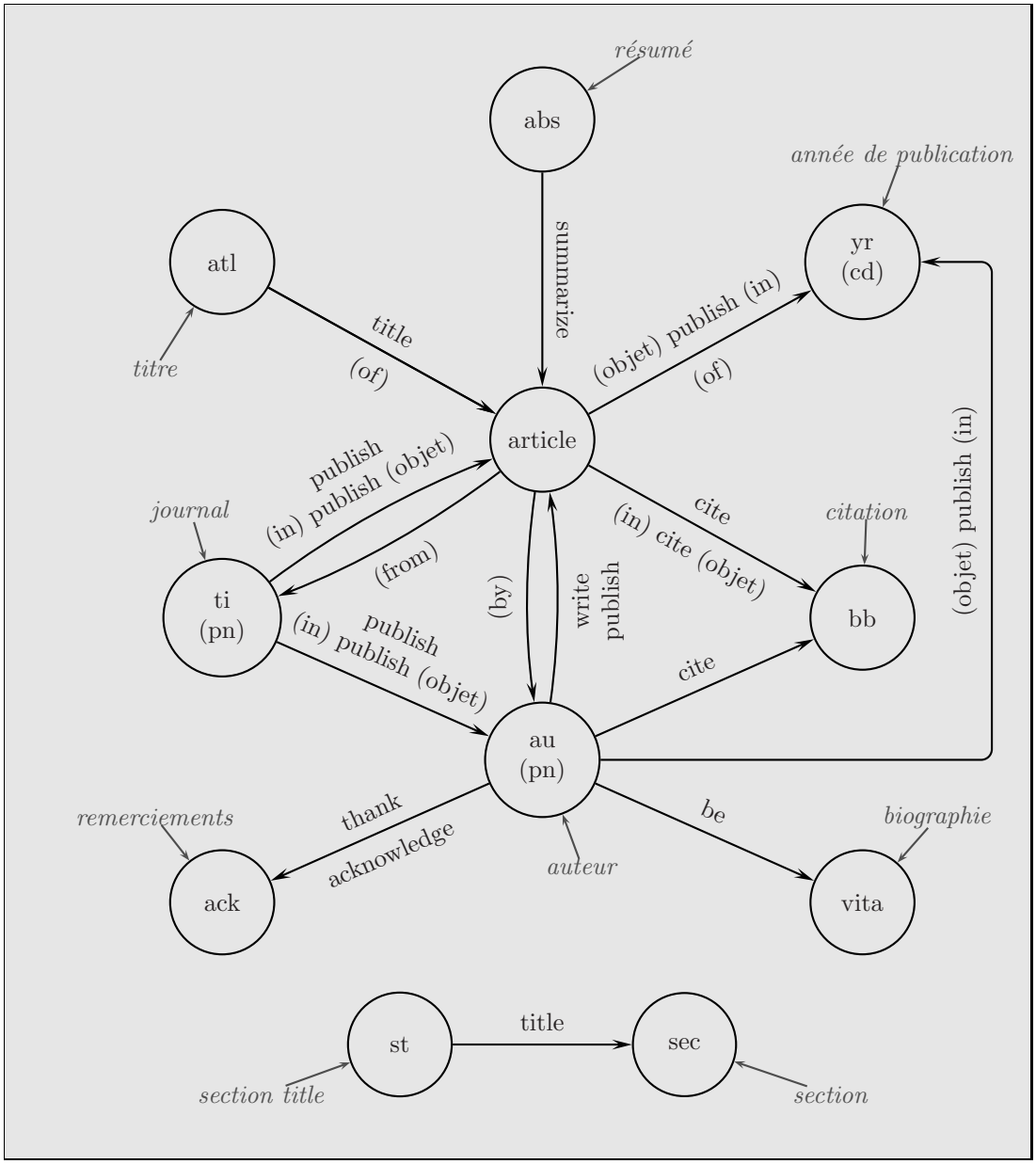


FIG. 5.30 – Extrait du modèle de la collection INEX 2005.

parenthèses, est précisée la relation que ce verbe possède avec les éléments. Si ces relations ne sont pas inscrites, il s'agit par défaut de *sujet* (à gauche) et d'*objet* (à droite). Ainsi, dans le modèle :
– un auteur `'au'` (*sujet*) *"écrit"* (*"writes"*) un `'article'` (*objet*).
– un `'article'` (*objet*) *"est publié"* (*"is published"*) *dans* (*in*) un journal `'ti'`.
Certaines relations peuvent surprendre, comme la relation *be* entre un auteur (`'au'`) et un résumé biographique (`'vita'`). Elles sont dues au fait que ce sont également *les contenus* des balises qui sont en relation. Ainsi un auteur *est* un professeur, ou un doctorant, et cette information est contenue dans le résumé biographique.

– Quand la relation est prépositionnelle, la préposition concernée est indiquée entre parenthèses. Ces liens sont plus classiques, comme par exemple : `'article'` *by* `'au'` (auteur).

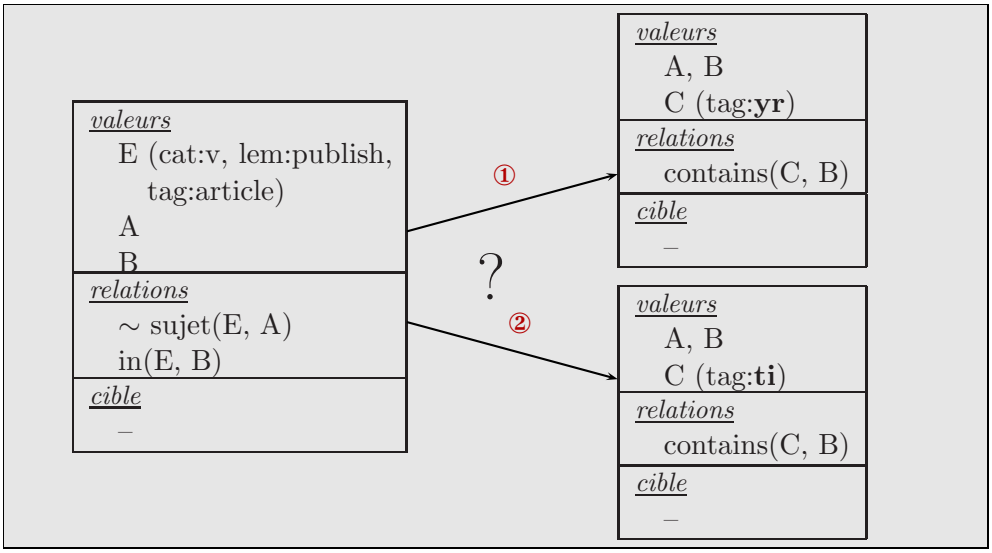


FIG. 5.31 – Ambiguïté entre deux règles sémantiques.

N.B. : Remarquons la relation <*title*> *of* <*article*>, qui permet de corriger l'analyse de l'exemple 5.6, donné à la section 5.2.4.4 sur les anaphores[1] :

(5.6) Articles containing "digital libraries" in their title.

La représentation (correcte) de la figure 5.9.b[2] est donc finalement préférée.

5.3.5.2 Les ambiguïtés du modèle

Remarquons tout d'abord que les énoncés établissant une relation entre deux éléments de façon totalement implicite, c'est-à-dire ne faisant référence qu'aux contenus, et pas aux noms de balises, sont très rares dans notre cas. En effet, nous avons affaire à des demandes d'information, et si la structure est impliquée, c'est que l'utilisateur ignore quelque chose. Nous aurons donc affaire à des références explicites ou des pronoms interrogatifs[3].

Grâce à cette restriction, le risque d'ambiguïté n'existe pas entre chaque paire d'arcs partageant la même étiquette. Ainsi, l'utilisation du verbe *"title"*, possible entre deux paires de nœuds (`'article'` et `'atl'` d'une part, `'sec'` et `'st'` d'autre part) n'est pas ambiguë, puisque l'on attend au moins une référence explicite (par exemple à un article) qui précise de quel arc il s'agit.

Il existe donc ambiguïté entre deux arcs partant du même nœud, ou conduisant au même nœud, et partageant bien sûr la même étiquette. Ainsi, le modèle de la figure 5.30 peut comporter une ambiguïté entre une année de publication (`'yr'`) et un journal (`'ti'`) à travers la relation *(in) publish* (figure 5.31). C'est le cas pour des énoncés comme :

(5.14) an article *published in* 2002. (①)

(5.15) an article *published in* IEEE Multimedia. (②)

[1] Page 127.

[2] Page 128.

[3] On ne dira pas par exemple *"Frank Dupont remercie son directeur de thèse"*, faisant ainsi allusion à l'auteur et aux remerciements, puisqu'il ne s'agit pas d'une demande d'information. En revanche, nous

En pratique, cette ambiguïté, comme toutes celles concernant le modèle d'INEX 2005, est résolue par le fait que la catégorie attendue n'est pas la même pour une année (`'yr'`, un nombre – *cd*) et pour le nom du journal (`'ti'`).

Mais si le modèle décrit ici n'est pas ambigu, un autre peut l'être ; nous verrons au chapitre 6 [1] la solution que nous apportons.

5.3.6 La désambiguïsation par la sémantique

Nous avons annoncé dans les pages précédentes que l'application des règles sémantiques permettrait une réduction des ambiguïtés provoquées à deux niveaux : au niveau de l'analyse syntaxique d'une part, et au niveau de la détection des références explicites à la structure des documents d'autre part. Nous décrivons ici comment cette désambiguïsation est rendue possible.

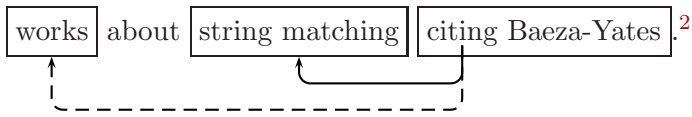
L'idée est que, si les règles sémantiques sont correctement formulées, les constructions qu'elles détectent ont une signification forte pour la requête. En substance, plus une représentation est propice à l'application de règles sémantiques, plus elle a de chances d'être la forme la plus appropriée pour la traduction de l'énoncé.

Pour matérialiser cette idée, nous mettons en place des "points de sémantique", que nous attribuons à une représentation en fonction du nombre et du type de règles qui s'y associent, selon les critères suivants.

5.3.6.1 Nombre de règles utilisées

Le premier critère est le nombre de règles sémantiques déclenchées. Prenons l'exemple de la requête suivante :

(5.16) Find works about string matching citing Baeza-Yates. [2]



Deux représentations sont possibles, illustrant le fait que le syntagme *"citing Beaza-Yates"* peut être attaché à *"string matching"* ou à *"works"*. Selon la règle de l'association droite*, la première solution devrait être choisie.

A chaque déclenchement d'une règle sémantique, on attribue un point à la structure concernée. Comme le montre la figure 5.32, la seconde forme (attachement à *"works"*) met en œuvre une règle de plus que la première. En effet, *"work"* est un synonyme répertorié d'*article*, qui est une balise. La règle <*article*> *cite* <*bb*> s'applique donc (voir le modèle de la figure 5.30). C'est donc finalement cette solution qui est préférée.

5.3.6.2 Elément ou texte ?

Certaines règles s'appliquent de préférence à des éléments, d'autres à du contenu. Par exemple, les règles de détection de la cible (section 5.3.3.1) sont plutôt destinées à des éléments structurels. Dans le même ordre d'idée, on peut considérer qu'une règle introduisant la relation *about* (voir figure 5.18, page 136) concerne plus probablement un élément pour sa partie gauche (premier paramètre du prédicat) et du texte simple

pourrons dire *"Qui remercie son directeur de thèse"*, ou *"Quel auteur remercie son directeur de thèse"*, ou *"Qui Franck Dupont remercie-t-il"*.

[1] Section 6.5.2.4, page 169.

[2] Extrait du topic 280, INEX 2005.

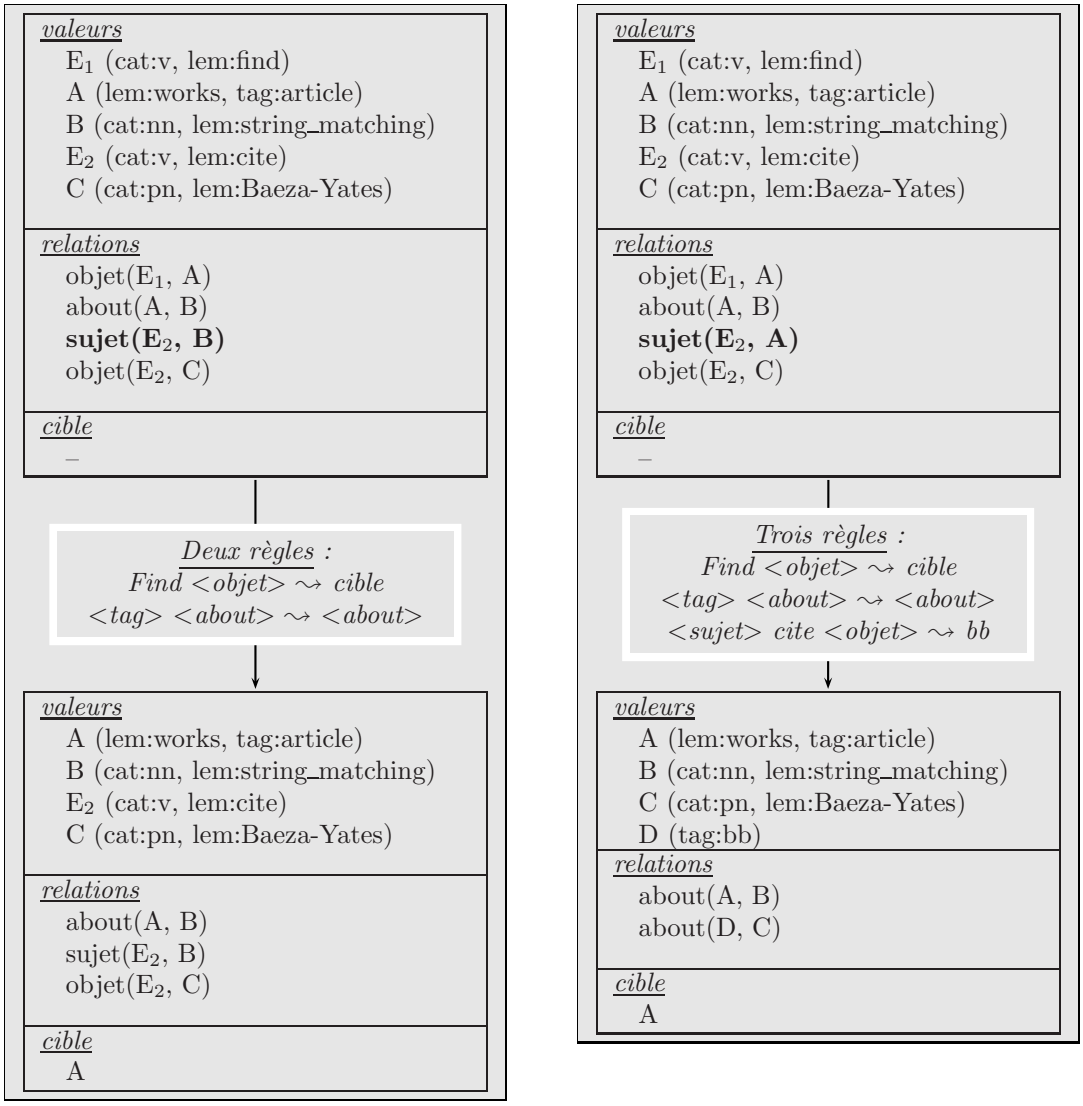
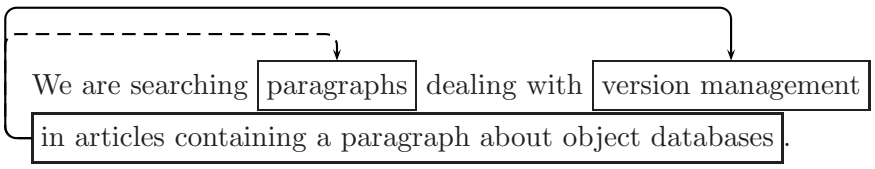


FIG. 5.32 – Désambiguïsation par le nombre de règles déclenchées (la représentation de droite, qui permet le déclechement de trois règles, est préférée à celle de gauche).

pour sa partie droite (second paramètre)[1]. Pour d'autres relations (*contains*, *included*, *before*, *after*, etc.), on préfèrera l'élément structurel.

Ces constats permettent tout d'abord de désambiguïser encore certaines constructions. Ainsi, l'exemple suivant est ambigu à de nombreux endroits, mais notamment en ce qui concerne l'attachement de la préposition *"in"* (voir la représentation de cet énoncé et l'application des règles correspondantes à la section 5.3.3.4 [2]) :

(5.13) 

[1]Par exemple, on trouvera plutôt *"a document about information retrieval"*, et non *"a document about a section"*.

[2]Page 137. La relation en gras *in(B, D)* est en concurrence notamment avec *in(C, D)*.

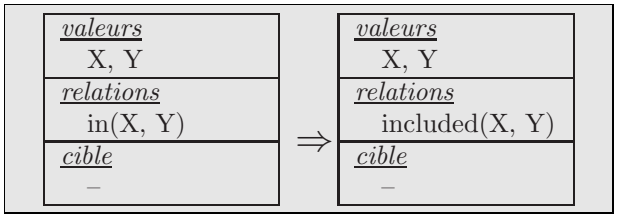


FIG. 5.33 – Règle pour la préposition *"in"*.

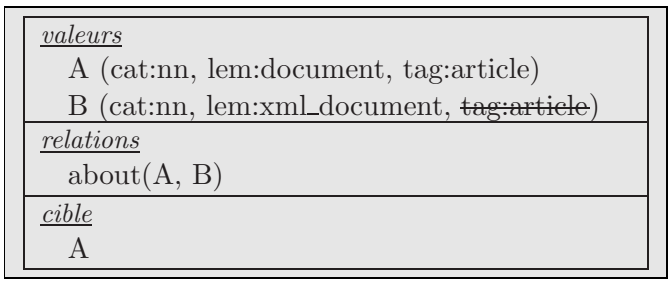


FIG. 5.34 – Choix entre élément textuel et élément structurels.

La règle concernant cette préposition, donnée de nouveau à la figure 5.33, s'applique de préférence entre deux éléments. Un point de plus est attribué si c'est le cas, ce qui permet de sélectionner la bonne représentation (celle où *"in articles"* est attaché à *"paragraphs"*).

Par ailleurs, nous avons décrit à la section 5.3.1 les problèmes posés lors de la détection des références explicites à des noms de balises contenus dans les documents interrogés. Les problèmes de polysémie des termes employés conduisent à l'utilisation d'un même terme, pour désigner tantôt des éléments structurels, tantôt du contenu textuel.

Là encore, la distinction entre règles s'appliquant à des éléments ou à du texte nous permet de réduire cette ambiguïté.

Ainsi, si un terme est reconnu comme pouvant faire référence à une balise, et s'il est concerné par une règle qui favorise les éléments, alors l'hypothèse de l'élément structurel sera validée. En revanche, si la structure considérée obtient plus de points en considérant ce terme comme du texte simple, alors l'élément est rejeté.

Ainsi, dans l'exemple suivant, le mot *"document"*, synonyme potentiel d'*article*, est une fois un élément structurel et la seconde fois du texte simple. On aboutit à la représentation finale de la figure 5.34.

(5.17) Find a document about XML documents.
+ élément *+ élém.* *+ texte*

Pour résumer :

- Les règles de détection de la cible sont utilisées de préférence pour des éléments.
- Parmi les autres règles, celles qui génèrent la relation *about* sont appliquées à des éléments (à gauche) sur du contenu (à droite) : *about(élément, texte)*.
- Les autres règles (relations *contains*, *includes*, etc.) concernent une paire d'éléments.
- Lorsqu'une règle respectant ces contraintes s'applique à une représentation, un point de plus est attribué à celle-ci.

valeurs A (lem:paragraph, tag:p) B (lem:article, tag:article) C (lem:information_retrieval) D (lem:relevance_feedback)	*valeurs* A (lem:paragraph, tag:p) B (lem:article, tag:article) C (lem:information_retrieval) D (lem:relevance_feedback)
relations about(B, C) **about(B, D)** $\rightsquigarrow$ **-1 pt.**	*relations* about(B, C) **about(A, D)**
cible A	*cible* A

FIG. 5.35 – Règle d'association de contenu textuel à un élément structurel. L'élément *A* n'est associé à aucun contenu dans la représentation de gauche, d'où le choix de celle de droite.

5.3.6.3 Elément et contenu

Une dernière règle stipule que, dans la mesure du possible, un élément structurel doit être associé avec du contenu textuel. Un point est retiré si ce n'est pas le cas. Cette règle permet de traiter des énoncés complexes comme le suivant :

(5.18) We are looking for paragraphs in articles about information retrieval dealing with relevance feedback.[1]

Le syntagme *"dealing with relevance feedback"* est d'abord attaché en priorité à *"articles"* ou *"paragraphs"* plutôt qu'à *"information retrieval"*, car la règle <*sujet*> *deal* <*with*> accepte de préférence des éléments à sa gauche.

Mais, comme le montre la figure 5.35 (à gauche), le choix de *"articles"* ferait en sorte que *"paragraphs"* ne serait associé à aucun contenu. La structure de droite (dans la figure) est donc préférée.

5.3.6.4 Les autres ambiguïtés

Les ambiguïtés non traitées par les techniques décrites ci-dessus restent bien entendu nombreuses, et les méthodes de résolution par l'attachement minimal* ou l'association droite* sont trop naïves pour avoir une véritable efficacité sémantique. Cependant le traitement que nous mettons en place pour les groupes de mots nous permet de nous affranchir des ambiguïtés restantes. C'est ce qu'illustre l'exemple suivant :

(5.19) Find text describing algorithms for prefetching of web pages by the client.[2]

L'attachement de *"by the client"* à *"web pages"* est erroné. Pourtant, la recherche finale n'en souffrira pas, puisque les relations prépositionnelles fines de ce type (<*nom*> <*for*> <*nom*>, <*nom*> <*by*> <*nom*>) ne sont pas traitées. Les termes *"prefetching"*,

[1] Topic 145, INEX 2004.
[2] Topic 174, INEX 2004.

"web pages" et *"client"* seront recherchés dans le même contexte[1].

5.4 Obtention d'une requête formelle

L'obtention de la requête formelle à partir de la représentation sémantique obéit à des règles simples qui diffèrent selon le langage. Nous décrivons ici la génération d'une requête en NEXI, dont la description est donnée à la section 2.4.2.2, page 31.

5.4.1 Disposition des éléments structurels

Les éléments structurels peuvent être séparés en trois catégories : la cible, le support et les éléments subordonnés (inclus dans la cible ou le support). Les explications qui suivent sont illustrées par deux exemples, l'un réel, à la figure 5.36, l'autre volontairement simplifié, à la figure 5.37.

5.4.1.1 Cible

La représentation sémantique indique quelle est la cible* de la requête. Plusieurs configurations sont possibles :

La cible est un élément structurel. Dans ce cas, le nom de l'élément désigné est placé à l'extrême droite de la requête, à la place habituelle de la cible.

La cible est du texte simple. La cible devient alors un élément de type indifférent (`'*'`), et le texte est inséré dans le filtre de cet élément, avec la relation *about* : `//*[about(., texte)]`

Il existe plusieurs cibles. Même si elle est rare, cette configuration doit être considérée. On crée un nouvel élément de type `'*'` qui contient les cibles. Ces nouvelles relations hiérarchiques sont traitées comme les autres (voir plus loin).

5.4.1.2 Support

Le support* de la requête est composé de tous les éléments qui subordonnent la cible, c'est-à-dire qui la contiennent[2]. Les éléments qui contribuent au support sont ordonnés (du plus loin de la cible dans la DTD au plus proche) et sont placés dans le chemin principal, de gauche à droite, avant la cible (voir la figure 5.37).

5.4.1.3 Eléments subordonnés

Les éléments restants ne font pas partie du chemin principal de la requête, ils sont insérés dans les filtres de l'élément auquel ils sont subordonnés. C'est le cas du paragraphe F à la figure 5.36 et des éléments D et E à la figure 5.37.

5.4.2 Le contenu

Le texte simple est disposé dans la requête selon ses relations avec les éléments structurels. La méthode décrite ci-dessous est illustrée aux figures 5.38 et 5.39, qui

[1] Le chapitre 6 montre une utilisation de ces relations prépositionnelles, mais sans contredire ce que nous venons de décrire.

[2] Dans le cas de NEXI, qui ne connaît que les liens hiérarchiques (ancêtre ou descendant), on dit qu'un élément est subordonné à un autre s'il est contenu par un autre (relations *includes* ou *included*).

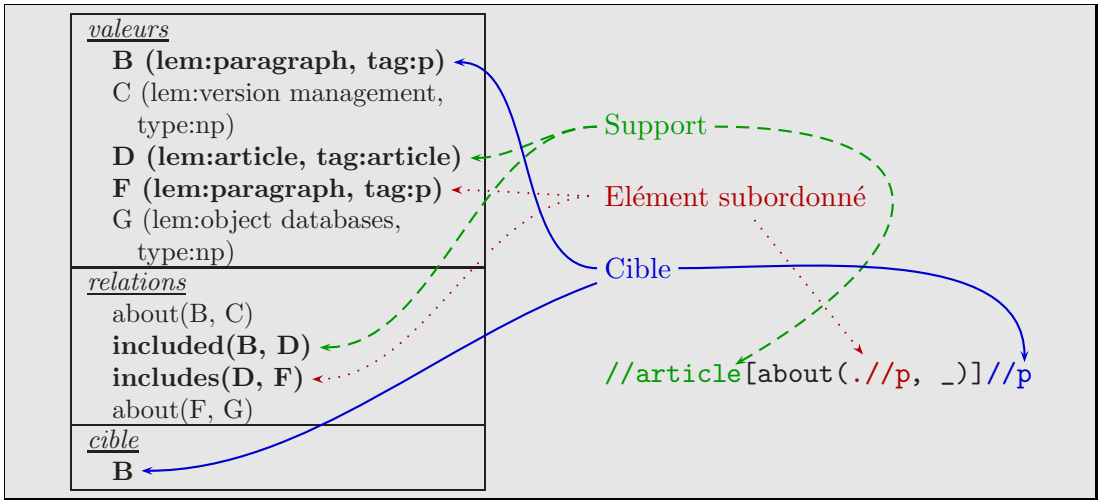


FIG. 5.36 – Distinction entre cible, support et autres éléments, pour la requête *"We are searching paragraphs dealing with version management in articles containing a paragraph about object databases."*

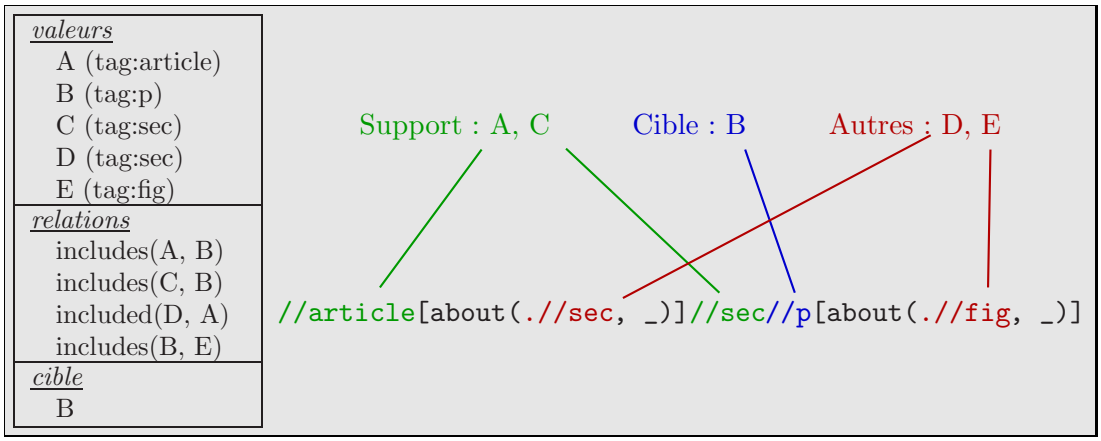


FIG. 5.37 – Distinction entre cible, support et autres éléments.

poursuivent les deux exemples de la section précédente.

5.4.2.1 Le cas général

Les termes liés à un élément structurel par un prédicat *about* sont insérés dans les filtres *about* de NEXI à la suite de la référence à cet élément. Si celui-ci fait partie du support ou de la cible, le chemin relatif, premier paramètre de la relation *about*, sera un simple point `'.'` (*"object databases"* à la figure 5.38, *"term3"* à la figure 5.39). Sinon, le chemin est le nom de l'élément, comme indiqué précédemment (*"version management"* à la figure 5.38, *"term1"* et *"multiterm2"* à la figure 5.39).

Lorsque deux filtres s'appliquent au même élément, ils sont séparés par l'opérateur `'AND'`.

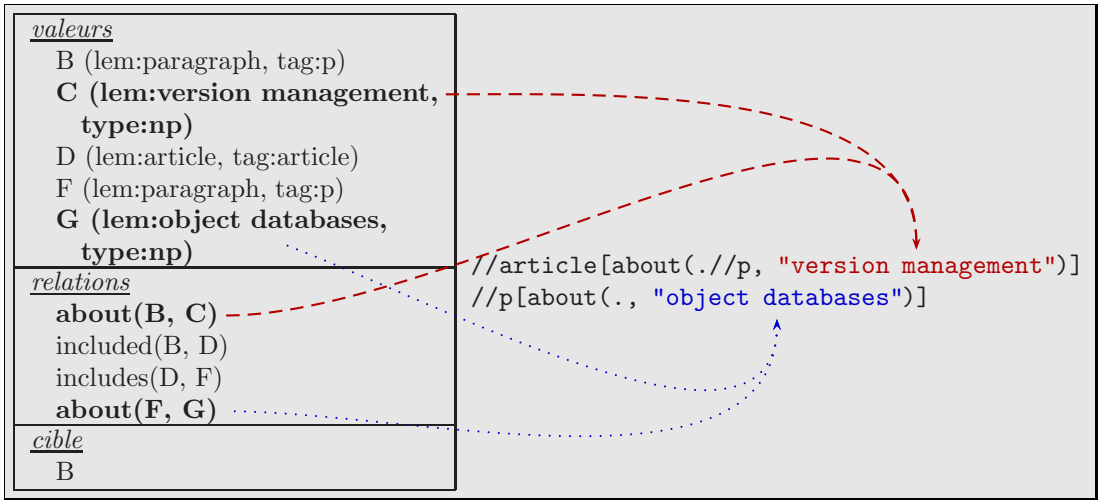


FIG. 5.38 – Insertion du contenu dans la requête *"We are searching paragraphs dealing with version management in articles containing a paragraph about object databases."*

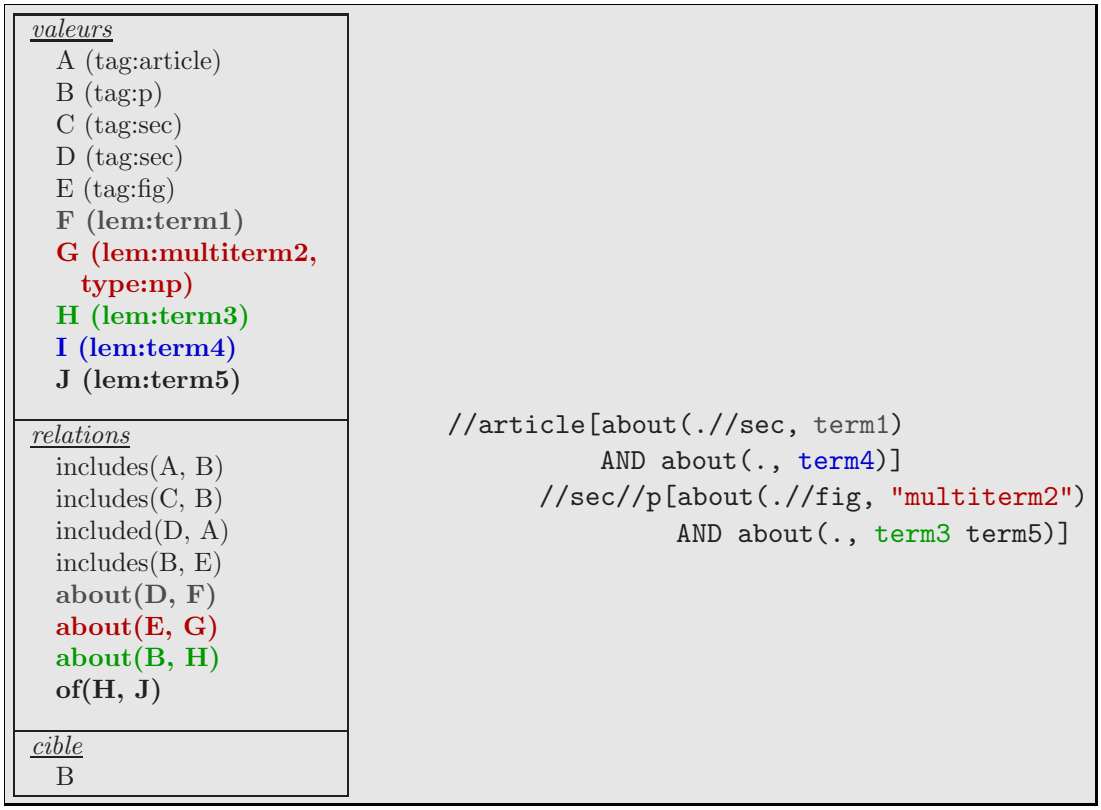


FIG. 5.39 – Insertion du contenu dans la requête.

5.4.2.2 Les syntagmes nominaux simples

Les termes reconnus lors du pré-traitement comme des syntagmes nominaux simples[1], et donc marqués par un trait `'type:np'`, sont entourés de guillemets dans la requête NEXI, comme on le voit à la figure 5.38 et pour *"multiterm2"* à la figure 5.39.

Les moteurs de recherche, en présence des guillemets, ont des comportements très

[1] Section 5.2.3, page 119.

variés, de l'ignorance totale à la considération stricte (un seul mot du groupe n'est pas valable), en passant par une simple valorisation du score en présence de tout ou partie du groupe nominal.

5.4.2.3 Traitement des relations

Les relations entre termes non structurels sont utilisées pour disposer de façon appropriée certains termes qui ne sont pas directement liés à des éléments. Ainsi, dans l'exemple de la figure 5.39, *"term5"* (J) est indirectement relié à l'élément B par les relations *about(B, H)* et *of(H, J)*. Ainsi, *"term3"* (H) et *"term5"* (J) sont tous deux contenus par la même balise B. Un autre exemple : pour la requête *"a paragraph about prefetching of web sites"*, les termes *"prefetching"* et *"web sites"* sont insérés dans l'élément *"paragraph"* (`'p'`).

5.4.2.4 Le texte non relié à un élément

Tout texte non relié, directement ou indirectement, à un élément structurel, est inséré comme filtre de la racine des documents (ici, `'article'`). C'est le cas de I (*"term4"*) à la figure 5.39.

5.4.2.5 Les disjonctions, conjonctions et négations

La coordination se traite de façon très simple. En ce qui concerne les éléments, la disjonction et la conjonction ont le même résultat, qui est la référence à un élément pouvant avoir plusieurs noms. NEXI autorise ceci avec le recours au caractère `'|'`. Ainsi, *"paragraphs or figure-captions"*, dans l'exemple suivant, sont traduits en NEXI par `'(p|fgc)'` (voir figure 5.40).

(5.20) Find paragraphs or figure-captions containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications.[1]

La négation d'un élément est reportée en négation de tous les termes qu'il contient (voir ci-dessous).

En ce qui concerne la coordination des contenus textuels, elle est transcrite par plusieurs clauses *about*, liées par les opérateurs adéquats. C'est ce que montre également la figure 5.40 pour l'exemple 5.20 ; l'opérateur `'AND'` est alors utilisé pour faire le lien avec les termes associés par des prépositions, comme ici avec *"definition"*.

Enfin, la négation est matérialisée en NEXI par un signe `'-'` avant les mots-clés concernés. Nous avons vu[2] que les négations, quelles qu'elles soient, étaient toutes propagées vers les syntagmes nominaux grâce à la relation *neg*. La traduction en NEXI est donc immédiate, comme le montre l'exemple suivant et la figure 5.41.

(5.21) I am interested in articles discussing machine translation but not programming languages translation.[3]

Dans le cas où le prédicat *neg* concerne du texte qui n'est relié à aucun élément ou contenu, ce texte n'est pas du tout inséré dans la requête finale, que ce soit en positif ou en négatif. Ainsi, dans l'exemple 5.22, le mot *"distances"* sera tout simplement ignoré. En effet, sa relation avec le verbe *"modify"* n'entre pas dans un cadre sémantique

[1] Topic 127, INEX 2004.
[2] Section 5.2.4.3, page 126.
[3] Topic 236, INEX 2005.

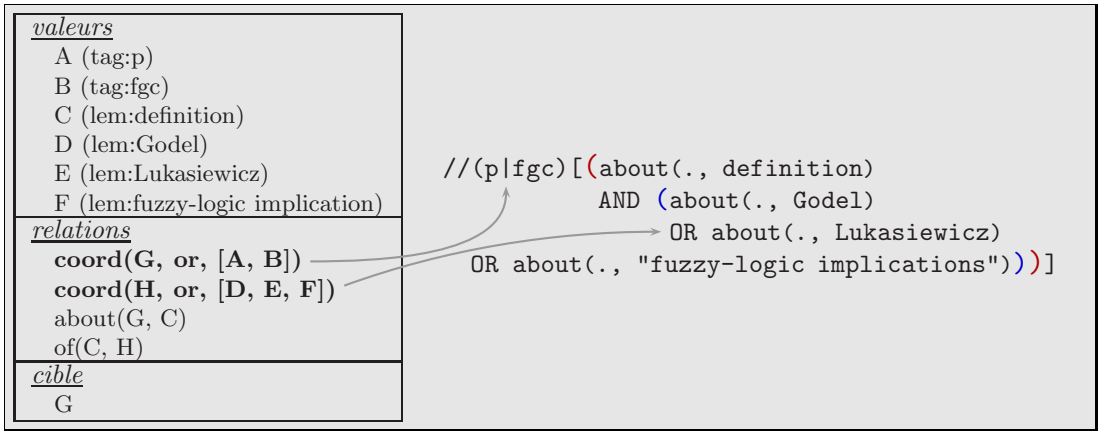


FIG. 5.40 – Traitement des coordinations.

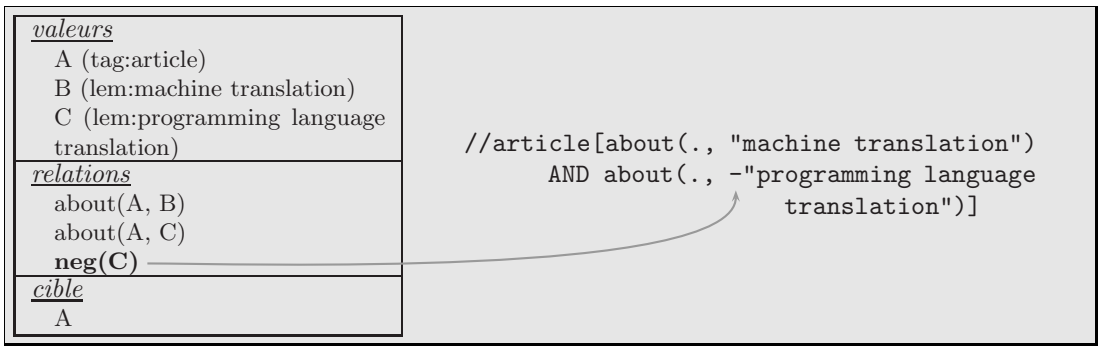


FIG. 5.41 – Traitement de la négation.

existant, et il est impossible de savoir sans apport supplémentaire si la présence du terme *"distances"* contribue à la pertinence d'un élément ou, au contraire, nuit à cette pertinence. Or une erreur pourra avoir des conséquences importantes ; nous préférons donc ne pas considérer le mot, par précaution.

En revanche, *"stoplist filtration"* sera accompagné du signe `'-'`, puisque l'expression *to be interested in* génère la relation *about*.

(5.22) ... Examples of this method are well-known latent semantic indexing (LSI) which improves recall of the retrieval system and random projection which **does not modify distances** too much. We **are not interested in stoplist filtration**...[1]

5.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode de traduction des requêtes en langage naturel vers un langage formel de type NEXI, s'appliquant à la recherche d'information dans des documents XML. Cette traduction s'appuie sur une analyse syntaxique traditionnelle à laquelle nous associons un ensemble de règles sémantiques simples et une procédure de génération de la requête adaptée à la problématique de la recherche d'information semi-structurée. Les règles sémantiques ne sont pas pour la

[1] Extrait du topic 187, INEX 2004.

plupart des heuristiques *ad hoc*, mais des règles générales ne dépendant pas de la forme linguistique utilisée.

Le fait de produire un langage reconnu et largement employé, mais aussi relativement simple, est un avantage indéniable, puisque cela permet à l'interface en langage naturel d'être utilisée en complément facultatif de tout moteur de recherche. De plus, la partie générant la requête finale étant relativement indépendante, il est possible d'adapter le système à un autre langage.

Jusqu'ici, nous nous sommes concentré sur une transcription aussi fidèle que possible de la description du besoin d'information. Ceci ne peut permettre, au mieux, que d'approcher les performances des requêtes effectuées "à la main" en NEXI. Il est temps d'orienter le travail vers une utilisation plus fine de la structure des documents, permettant potentiellement de générer des expressions plus efficaces (qualitativement parlant) que celles d'un utilisateur humain.

Cependant, les langages existant, et NEXI en particulier, ont des limites quant à la faculté d'expression de certains types de requêtes. L'utilisation de certaines techniques d'analyse du langage naturel pourrait permettre d'affiner la requête, mais le "carcan" des langages non conçus pour cet usage empêche de donner libre cours à son imagination. Le chapitre suivant étudie les tenants et les aboutissants de ce nouvel enjeu.

Chapitre 6

Utilisation de la structure pour l'amélioration de la requête

Le chapitre précédent a présenté les mécanismes d'analyse d'une requête en langage naturel, en vue de l'obtention d'une représentation intermédiaire, puis de la construction d'une requête formelle dans un langage orienté vers la recherche d'information, tel que NEXI.

A l'issue de ce travail, on peut dire que l'on obtient, dans le principe, des requêtes comparables à celles que construisent les utilisateurs humains : mêmes mots-clés (moyennant bien sûr les erreurs de l'interface) et même degré de mise à profit "bas-niveau" de la structure. Par "bas-niveau", nous entendons une considération purement "utilitaire" : si j'ai besoin d'une figure, je demande une figure, si je souhaite qu'un paragraphe traite d'un sujet, je formule une expression de type `//paragraphe[about(., "ce sujet")]`.

Nous avons déjà expliqué que ce type de requêtes était le maximum que l'on pouvait raisonnablement espérer d'un utilisateur humain, même expert, pour un usage régulier d'un moteur de recherche.

Si une interface offrant cette fonctionnalité présente déjà un avantage certain en termes de confort (à condition bien sûr que les requêtes générées offrent des performances satisfaisantes), il est possible d'aller au-delà avec un processus automatique, en étudiant d'une part les constructions de la requête en langage naturel, et d'autre part la structure spécifique des documents recherchés.

Les idées (et leurs applications) présentées dans ce chapitre impliquent trois facteurs spécifiques des documents semi-structurés dans la recherche d'information. Ces facteurs sont :

- l'imbrication des éléments, qui bouleverse le principe d'indépendance des unités de recherche entre elles, principe admis en RI traditionnelle (section 6.1) ;
- les liens explicites entre les documents (section 6.2) ;
- la taille variable de l'unité d'information (section 6.3).

Il s'avère que le langage NEXI ne permet pas d'exprimer des requêtes évoluées comme celles que nous proposons. Par ailleurs, de nombreuses caractéristiques collectées par une analyse linguistique peuvent trouver une utilité pour la recherche, et NEXI n'est pas conçu pour les traiter.

C'est pourquoi nous introduisons le langage XOR (XML Oriented Retrieval Language), une représentation extensible et compatible avec NEXI. La description de ce langage est présentée à la section 6.4, et l'adaptation du système GPX pour l'intégrer y est rapidement abordée.

Enfin nous exposons les améliorations que nous apportons à notre interface dans le but de prendre en compte les diverses propositions évoquées et les traduire dans le langage XOR (section 6.5).

6.1 Recherche contextuelle

6.1.1 Problématique

Nous avons déjà abordé à plusieurs reprises, et en particulier à la section 2.2.1 [1], la notion d'auto-explicativité d'une unité d'information.

Cette notion permet de postuler que les unités de recherche sont indépendantes les unes des autres. Ainsi, le degré de pertinence d'un document par rapport à une requête n'a aucune influence sur le degré de pertinence d'un autre document.

L'idée simple qui découle de cette absence de lien entre les documents et qui gouverne les méthodes de recherche d'information, quelles qu'elles soient, est que plus un document contient de mots de la requête (ou de mots apparentés, morphologiquement ou sémantiquement), plus il a de chances d'être pertinent. Le corollaire sous-jacent est qu'un document contenant peu de mots de la requête a moins de chances d'être pertinent que celui qui en comprend beaucoup.

Cette hypothèse est discutable sur les documents plats, puisque des documents peuvent faire référence à d'autres, mais devient très certainement fausse en présence d'une structure autorisant l'imbrication des éléments. Il n'est bien entendu plus possible de supposer l'indépendance des unités lorsque celles-ci peuvent se contenir mutuellement. La plupart des modèles de recherche d'information structurée prennent d'ailleurs en compte ce phénomène, puisque les scores de pertinence des doxels dépendent de ceux de leurs enfants (soit grâce à une indexation par sous-arbres imbriqués – voir section 2.3.2, soit par une propagation des scores – section 2.4.3.1). Cette transmission des scores "de bas en haut" est intuitive : un terme contenu dans un paragraphe, par exemple, fait également partie de la section contenant ce paragraphe, et donc participe à son sens.

Nous pensons cependant que ce problème a des conséquences négatives importantes sur les performances globales des systèmes de recherche. En effet, même en considérant la propagation de bas en haut, les techniques employées partent toujours du principe qu'un élément sera pertinent s'il contient les mots de la requête, et ne le sera pas dans le cas contraire. Or, cette dernière idée (peu de mots implique faible pertinence) ne rend plus compte de la réalité d'un document XML.

En effet, des textes formant un tout, explicitement structuré ou non, sont faits pour être lus en entier. En conséquence, au niveau global, ces textes introduisent d'abord un contexte, puis développent différents points autour de ce contexte, qui n'est bien entendu pas rappelé à chaque paragraphe. De plus, à un niveau plus local, différents phénomènes linguistiques, comme les anaphores et les ellipses[2], permettent de ne pas repéter explicitement les concepts dont il est question d'une phrase à une autre, d'un paragraphe à un autre. En particulier, s'agissant d'une personne, il n'est pas rare de voir des références pronominales (*"il"* ou *"elle"*) la désigner, plusieurs pages durant, si elle est le sujet central du texte.

La figure 6.1 illustre ce problème avec l'exemple d'une requête concernant les *contrats de travail pour les jeunes*, où l'on voit schématiquement que les sections pertinentes par

[1] Page 20.
[2] Voir la section 3.4.3, page 68.

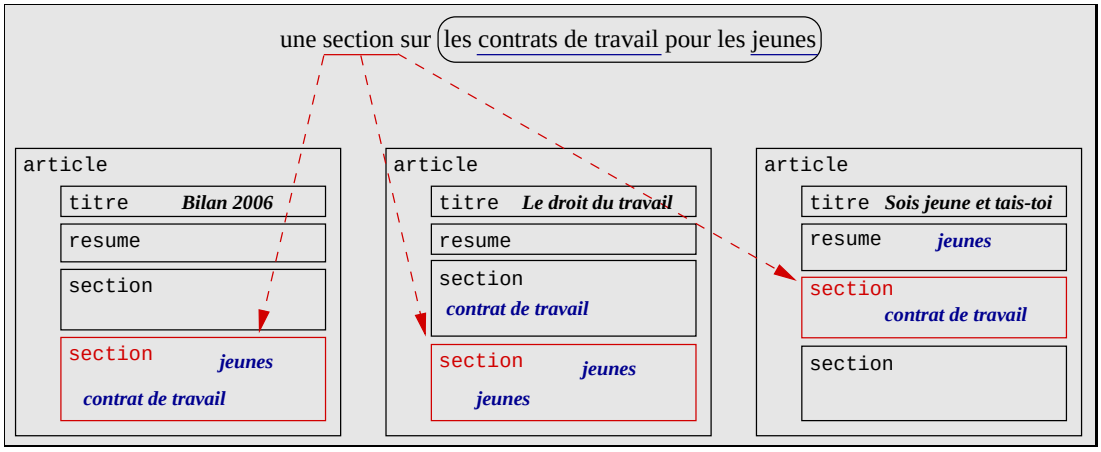


FIG. 6.1 – La pertinence d'un élément dépend également dans son contexte. Ici la requête concernant les *contrats de travail pour les jeunes* peut trouver trois types de sections pertinentes, dont seulement un contient à la fois les termes *"contrats de travail"* et *"jeunes"*.

rapport à cette requête peuvent ne pas contenir l'ensemble des termes demandés, tout simplement parce que certains d'entre eux ont été introduits auparavant[1].

Très peu de modèles utilisent le *contexte* des éléments, c'est-à-dire le contenu qui les entoure ou qui est situé au-dessus d'eux. Les seuls systèmes proposant une méthode s'en approchant effectuent une *rétro-propagation*, initialement proposée par *Sigurbjörnsson et al.* [214], c'est-à-dire qu'ils pondèrent le score d'un élément avec le score du document global qui le contient pour la même requête. Cette technique semble améliorer les résultats mais n'est somme toute qu'un pis-aller.

Une analyse attentive, à l'occasion des évaluations de la campagne INEX, des résultats retournés par les systèmes de recherche d'information, montre pourtant que le phénomène décrit ci-dessus est suivi d'effets négatifs non négligeables sur la précision de ces systèmes. Ainsi, une recherche concernant les *systèmes de navigation pour les automobiles*[2] a vu beaucoup d'éléments concernant les systèmes de navigation dans les avions ou les bateaux retournés dans les premiers rangs. Cela était dû au fait que les éléments contenant l'ensemble des termes *"systèmes de navigation"* et *"automobiles"* étaient relativement rares, et que, au-delà de ces éléments, les moteurs de recherche n'étaient pas capables de faire la distinction entre les allusions aux *systèmes de navigation* dans un contexte automobile, aérien ou maritime.

6.1.2 Modification des requêtes pour la recherche contextuelle

Notre cadre d'analyse des requêtes peut permettre de prendre en compte cette problématique et de modifier le mécanisme de génération des requêtes formelles. Avant d'aborder les aspects pratiques de la détermination du contexte et de la construction de la requête, étudions les possibilités de modifications qui se présentent.

[1] Rappelons que nous n'adressons pas ici les problèmes des mots apparentés tels que les synonymes, ni celui de la polysémie, abordés à la section 3.5. Ces problèmes ne sont en effet pas spécifiques au cas des documents semi-structurés.

[2] "navigation systems for automobiles", Topic 128, INEX 2004.

6.1.2.1 Aspects théoriques

Des requêtes répondant de façon plus appropriée au problème du contexte seraient celles mettant en œuvre :

recherche contextuelle

- une *recherche contextuelle*, c'est-à-dire une recherche considérant le contexte dans lequel un élément pertinent doit apparaître. Par exemple, pour la requête concernant *des sections sur les systèmes de navigation pour automobiles*, on recherche des sections contenant les mots *"systèmes de navigation"* contenus dans des articles sur les *"automobiles"* (ou le contraire)[1]. *A priori*, une technique de ce type améliorerait le rappel* (plus de documents pertinents retournés) mais pourrait éventuellement diminuer la précision* (plus de documents non pertinents retournés).

recherche conditionnelle

- une *recherche conditionnelle* à l'intérieur d'un doxel recherché. Par exemple, pour éviter le retour d'éléments concernant les systèmes de navigation pour les avions ou les navires, on demanderait des doxels sur les *"systèmes de navigation"* **si, et seulement si** ces doxels traitent des *"automobiles"*. On imagine que ceci augmenterait la précision (les éléments retournés seraient très probablement pertinents) mais diminuerait fortement le rappel, le problème du contexte exposé plus haut n'étant pas réglé.

Malheureusement, ces deux caractéristiques peuvent difficilement être représentées avec une simple requête NEXI. De plus, la difficulté est de décider quels mots-clés peuvent être séparés et utilisés dans la recherche contextuelle.

6.1.2.2 Quelle recherche, quel contexte ?

La définition de ce qui peut appartenir au contexte, d'une part, mais aussi le choix des termes qui peuvent être séparés et de ceux qui doivent rester regroupés, d'autre part, est délicate. Le risque de faire chuter la précision en produisant des requêtes ayant peu de rapport avec le besoin initial est fort. Ainsi, dans notre exemple des *systèmes de navigation pour les automobiles*, la distinction entre *"systèmes de navigation"* d'une part, *"automobiles"* de l'autre semble pertinente, tandis que rechercher les termes *"systèmes"* et *"navigation"* séparément et à des endroits différents serait clairement contre-productif.

Il arrive que le rédacteur de la requête en langage naturel donne des indications explicites. Il utilise alors des expressions comme *"dans le contexte de"*, *"dans le domaine de"*, etc. Cela ne signifie pas pour autant que le contexte sera le même dans tous les documents pertinents. Ainsi, une requête sur *"les systèmes de navigation dans le domaine automobile"* ne fournirait que peu d'information supplémentaire ; dans l'esprit de l'utilisateur, l'automobile est le contexte, mais un document peut traiter des systèmes de navigation en général et contenir une section (pertinente) sur leurs applications dans le domaine automobile. Cependant ces tournures de phrases sont importantes car elles suggèrent la nécessité d'un traitement différent.

D'autres indicateurs implicites sont les liens entre les syntagmes nominaux. Nous avons présenté à la section 5.4.2.2 [2] les suites de mots que nous traitons comme des groupes indissociables dans la requête (et donc représentés entre guillemets dans la requête NEXI). En suivant cette logique, nous considérons que ces mêmes mots ne peuvent pas être dissociés et dispersés entre la cible et le contexte. En revanche, nous

[1] Pour distinguer les deux aspects, nous conservons la terminologie habituelle en nommant *cible* l'élément à retourner. Nous appellerons *support* ou *contexte*, selon les cas, les parties de la requêtes ne concernant pas directement la cible.

[2] Page 152.

avons montré que les noms ou groupes nominaux reliés par des *prépositions* (syntagmes de la forme *"NP_1 $PREP$ NP_2"*, règle 5.4, voir des exemples à la figure 5.2[1]) ne peuvent être traités comme des termes uniques. Qui plus est, on remarque que pour de nombreuses requêtes contenant des constructions de ce type, les groupes NP_1 et NP_2 apparaissent rarement ensemble dans des petits éléments, même pertinents. Ce sont essentiellement ces constructions qui vont nous servir de base à la mise en place d'une recherche contextuelle ou conditionnelle.

6.1.2.3 Premières réponses pratiques

Cette section décrit une première approche concernant les recherches contextuelle et conditionnelle, approche s'inscrivant dans le cadre de NEXI. Nous nous focalisons ici sur les groupes nominaux situés dans la partie cible de la requête et comportant des prépositions (NP_1 $PREP$ NP_2).

L'idée est donc de concentrer la cible de la requête sur une des parties du groupe nominal (NP_1, NP_2 ou d'autres, dans le cas de tournures plus complexes) et de détourner les autres éléments vers le support.

Comme nous l'avons indiqué, NEXI se prête relativement mal aux manipulations trop ambitieuses des requêtes. En particulier, le fait qu'il interdise la combinaison de plusieurs requêtes (et donc de plusieurs cibles) est très problématique. De plus, seul le prédicat `about` est autorisé.

Nous avons cependant essayé de simuler les deux types de recherches (contextuelle et conditionnelle) avec les moyens dont nous disposons. Considérons donc la requête demandant des éléments parlant de NP_1 $PREP$ NP_2. La traduction directe en NEXI est :

`//*[about(., NP`$_1$`) AND about(., NP`$_2$`)]`

Par exemple :

```
//*[about(., "navigation systems") AND about(., automobiles)]
```

Une transformation bilatérale (*"navigation systems"* dans le contexte de *"automobiles" OU "automobiles"* dans le contexte de *"navigation systems"*), qui nécessiterait une requête multiple ou de nouveaux prédicats, étant impossible, il faut choisir un sens. Au niveau théorique, il ne semble pas qu'il existe une présomption particulière. Pourtant l'inspection approfondie des documents d'INEX semble montrer que la disposition la plus courante correspond à la configuration "NP_1 *dans le contexte de* NP_2". Que cette observation soit exacte ou pas, devant la nécessité de faire un choix, nous avons décidé de modifier les requêtes de la façon suivante :

- pour la recherche contextuelle : ajout de NP_2 dans la partie support, concernant la racine du document (ici l'élément `'article'`) :
  `//article[about(., NP`$_2$`)]//*[about(., NP`$_1$`)]`
  `//article[about(., automobiles)]//*[about(., "navigation systems")]`

- pour la recherche conditionnelle : ajout du signe '+' avant NP_2 dans la cible :
  `//*[about(., NP`$_1$`) AND about(., +NP`$_2$`)]`
  `//*[about(., "navigation systems") AND about(., +automobiles)]`

  Rappelons que le signe '+' indique l'importance particulière du terme qu'il qualifie.

[1] Page 122.

Pour obtenir les effets bénéfiques attendus des deux méthodes et pour tenter d'atténuer leurs effets néfastes (respectivement sur la précision et sur le rappel), nous les avons en pratique utilisées de façon conjointe, toujours dans la même requête :

```
/article[about(., NP2)]//p[about(., NP1) AND about(., +NP2)]

/article[about(., automobiles)]//p[about(., "navigation systems")
                                   AND about(., +automobiles)]
```

Remarques.

- Dans le cas de groupes plus complexes ou reliés par d'autres relations, c'est toujours le NP situé le plus à droite qui devient le contexte.
- Pour de nombreux types de documents (notamment les articles d'INEX), une recherche contextuelle plus fine pourrait être effectuée, en recherchant par exemple des mots-clés dans le titre ou le résumé du document, dans le cas fréquent où ces champs sont balisés. Nous ne souhaitons pas expérimenter ceci, pour les raisons suivantes :
  - Nous désirons mettre en place un cadre le plus générique possible, et ces informations ne sont pas toujours disponibles.
  - Avec l'utilisation des pondérations de type *tf.ief*[1] tenant compte de la taille des éléments, les petites parties comme le titre, les mots-clés ou le résumé ont déjà plus d'influence que les autres dans le score de la racine.

6.2 Les liens entre documents

L'affinement des critères de pertinence d'un document en utilisant les liens inter-documents a été énormément exploré, en particulier dans le cadre de la recherche sur le Web [100], grâce aux liens HTML. Les documents XML contiennent également des références à d'autres documents de la collection, qu'il est possible de matérialiser en utilisant XLink[2].

Ainsi, dans la collection des articles scientifiques de l'IEEE, de nombreuses références bibliographiques sont disponibles (même si les articles cités n'appartiennent pas nécessairement à la collection). Dans la collection Wikipedia [61], utilisée pour INEX 2006, les liens inter-documents sont encore plus nombreux. De plus, chaque article de l'encyclopédie contient une liste d'autres articles de la collection qui le citent.

Nous ne nous intéressons pas aux moyens de déterminer la pertinence d'un élément en fonction des documents auxquels il fait référence, mais nous proposons plus simplement l'ajout d'une fonctionnalité de requête structurée permettant de formuler des demandes se propageant sur plusieurs documents. Ceci est rendu possible par l'ajout ou la modification de règles sémantiques.

6.2.1 De nouvelles règles sémantiques

L'ensemble des règles sémantiques que nous créons ou que nous modifions dépend de la structure. En ce qui concerne les articles scientifiques de l'IEEE, les citations ont une signification bien précise, il s'agit d'allusions dans le corps du texte à un élément contenu dans la bibliographie, à la fin du document. Dans ce cas, sont concernées par nos modifications les références explicites et implicites à la bibliographie, qui doivent

[1] Voir la section 2.3.1, page 24.
[2] Voir l'annexe A.3.

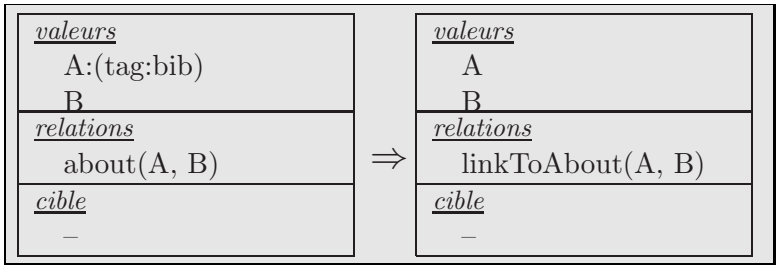


FIG. 6.2 – Nouvelle règle sémantique pour la bibliographie.

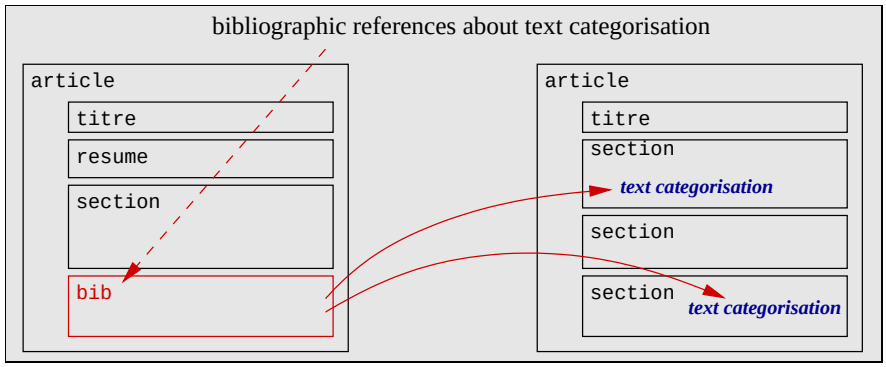


FIG. 6.3 – Recherche utilisant les liens.

conduire à l'exploration des documents cités. Ainsi, pour l'exemple suivant issu des requêtes INEX, il semble logique de consulter les articles cités pour découvrir s'ils traitent ou pas du sujet concerné, comme le montre également la figure 6.3 :

(6.1) Find bibliographic references that are about text categorisation where Support Vector Machines (SVM) categoriser is used.[1]

Nous proposons dans ce cas d'ajouter une règle simple, à appliquer après toutes les autres, qui transforme le lien habituel *about* dans la bibliographie en *linkToAbout*, exprimant l'idée que les liens potentiellement existant doivent être explorés (voir la figure 6.2).

6.2.2 Le rôle du langage formel

Détecter des liens potentiels inter-documents durant l'analyse de la requête ne suffit bien entendu pas, il faut également que le moteur de recherche soit capable d'explorer ces liens et que le langage formel intermédiaire permette de donner les ordres nécessaires au système.

NEXI, notamment, est incapable de formaliser les relations requises, puisqu'il ne peut faire référence qu'à un seul document dans une requête et que son seul prédicat autorisé est *about*.

[1] Topic 136, INEX 2004.

6.3 Etude concernant la taille des éléments à retourner

La granularité* des éléments peut varier énormément entre les documents, et même à l'intérieur d'un document. Par exemple, certaines méta-informations (auteurs, prix, année de publication, numéro ISBN...), plus orientées données*, sont contenues dans un élément qui leur est réservé. En revanche, à d'autres niveaux, le plus petit élément peut être un paragraphe entier.

Nous nous intéressons ici aux cas où les questions de l'utilisateur se font plus ciblées, et nécessitent une réponse brève et précise, contenue dans un ou quelques mots. En cela, nous nous éloignons un peu de la recherche d'information pure, qui adresse plutôt des besoins d'information vagues. Pourtant, le fait que le matériau de recherche soit structuré, parfois de façon très fine, donne envie de mêler la RI à ce qui s'apparente plus à du "question-réponse".

Dans ce domaine, où les requêtes sont le plus souvent formulées sous forme de questions, il est courant d'utiliser les pronoms interrogatifs pour cibler le type de texte que l'on va rechercher. Ainsi, une question commençant par *"combien"* appelle un nombre en retour ; *"qui"*, un nom propre, etc.

Dans notre cas, nous ne possédons pas d'information concernant le type de contenu des éléments. Nous pourrions ajouter cette connaissance pour un petit nombre d'entre eux (comme l'auteur ou l'année), mais la plupart des éléments contiennent du texte "classique". De plus, nous sommes limités par l'unité d'information, qui reste le doxel, et que nous ne pouvons pas dépasser sans sortir complètement du cadre fixé[1]. Nous ne pouvons donc pas rechercher des éléments très précis si la granularité n'est pas suffisante.

Sans nous engager dans des chemins qui nous entraîneraient en dehors de ce cadre, nous pouvons néanmoins proposer un mécanisme limitant la taille des éléments retournés. En effet, une question précise commençant par *"combien"*, *"qui"*, *"à qui"*, *"où"*, etc. n'appelle pas une réponse de la taille d'un document entier, ni même d'une grande partie de document.

La taille des éléments étant très variable, nous pouvons difficilement définir une taille maximum, au risque de refuser des textes pour la seule raison que le niveau de granularité dépasse cette taille maximum. Nous pouvons cependant rechercher *le plus petit élément possible*. Bien entendu, nous ignorons le nom de cet élément ; pour cette raison, le langage NEXI ne nous permet pas de mettre en œuvre cette idée.

6.4 Le langage XOR

Comme nous l'avons dit, les différentes problématiques que nous avons décrites dans les précédentes sections ne peuvent trouver de solutions satisfaisantes avec un langage tel que NEXI. D'autres idées, formulées par des chercheurs dans le domaine, souffrent également des limitations de ce langage. Nous récapitulons d'abord ces limites, avant de proposer un nouveau langage destiné aux concepteurs d'interfaces en langage naturel.

[1] Ce qui est souhaitable, mais un peu prématuré.

6.4.1 Limites de NEXI

NEXI, en tant que langage orienté vers la recherche d'information, empêche de considérer un certain nombre de caractéristiques[1] que nous récapitulons ici :

- NEXI n'accepte que des requêtes simples (ne comprenant qu'un seul élément cible). Ceci est très limitant ; nous l'avons déjà signalé au sujet de la recherche contextuelle, mais d'autres types de requêtes pourraient bénéficier de la mise en place de plusieurs cibles. Par exemple, supposons que nous recherchions des informations au sujet du fameux article d'Einstein sur l'électrodynamique, écrit en 1905. Nous pouvons demander directement cet article[2] :

```
//article[.//année = 1905
        AND about(.//auteur, Einstein)
        AND about(.//*, électrodynamique)]
```

  Mais nous souhaitons également voir quelques-uns des nombreux documents qui expliquent ou commentent cet article :

```
//article[about(., article Einstein 1905)
        AND about(., électrodynamique)]
```

  Ou encore, des articles des années 1900 sur l'électrodynamique, pour se rendre compte de l'état de l'art à cette période :

```
//article[about(.//année, 1900)
        AND about(., électrodynamique)]
```

  Ainsi, un besoin d'information unique (*"Je veux tout comprendre au sujet de ce fameux article d'Einstein, en 1905, sur l'électrodynamique"*) peut être représenté par au moins trois requêtes formelles différentes et complémentaires. Il est bien entendu possible de lancer séparément les trois expressions, mais le rang des éléments retournés est faussé et l'opération de recherche et de "dépouillement" des résultats plus longue.
- De plus, NEXI n'autorise que le prédicat *about*, bien entendu le plus important en RI, mais d'autres peuvent être intéressants. De nombreuses modes de recherche dépendent du paramétrage du moteur de recherche et ne peuvent pas être commandés par la requête. C'est le cas par exemple de l'utilisation de techniques de variations syntaxiques ou sémantiques telles que la racinisation* ou l'expansion de requête.
- Par ailleurs, NEXI ne permet pas de se référer à plus d'un article dans la même expression. Ainsi, des requêtes d'INEX concernent les références bibliographiques, mais les moteurs de recherche ne reçoivent pas explicitement d'instructions pour s'intéresser aux contenus des articles cités (s'ils sont disponibles dans la collection).
  C'est le cas de l'exemple suivant, déjà relevé plus haut :

  (6.2) Find bibliographic references that are about text categorisation where Support Vector Machines (SVM) categoriser is used.[3]

  Cette requête est exprimée en NEXI par son auteur de la façon suivante :

```
//bib[about(., text categorisation)
   AND about(., "Support Vector Machines" SVM)]
```

[1] En plus de cette liste, NEXI n'est pas conçu pour traiter de nombreuses contraintes orientées vers les bases de données (agrégation, construction du résultat, etc.), mais ces problèmes ne nous concernent pas ici.

[2] Nous utilisons ici des requêtes NEXI avec une structure issue de notre imagination pour la clarté du propos.

[3] Topic 136, INEX 2004.

Il s'agit donc d'une recherche dans une zone très limitée (les quelques mots de la référence bibliographique), alors que l'on souhaiterait s'appuyer surtout sur le contenu de l'article cité.

– Enfin, NEXI ne propose pas de moyen d'exprimer la moindre caractéristique supplémentaire concernant les éléments ou les termes recherchés. Parmi celles citées par les chercheurs, nous pouvons citer la taille minimum ou maximum d'un élément, le type d'interprétation des contraintes structurelles (strict ou vague) ou la catégorie grammaticale des mots.

6.4.2 Description du langage XOR

Le nouveau langage que nous proposons avec *Shlomo Geva*, de l'Université Technologique du Queensland (QUT, Australie), et *Marcus Hassler*, de l'Université de Klagenfurt (Autriche), a pour nom XML Oriented Retrieval Language (XOR). Il est conçu de telle façon qu'une requête en NEXI soit compatible avec lui, c'est-à-dire qu'un système lisant des expressions XOR fonctionne sans modification avec NEXI, que ce soit au niveau syntaxique ou au niveau sémantique.

De plus, il s'agit d'un langage générique, qui apporte une syntaxe globale, mais qui autorise l'ajout de toutes sortes de caractéristiques jugées utiles par des concepteurs d'un moteur de recherche, et principalement d'un système se basant sur l'analyse des descriptions en langage naturel du besoin d'information.

L'avantage principal de NEXI, qui est sa simplicité d'utilisation, est quelque peu mis à mal (même si on est loin de la complexité de la plupart des autres langages). Mais XOR est surtout destiné à être généré par une interface en langage naturel, et donc à être traité en entrée comme en sortie par une machine.

La syntaxe générale du langage XOR est la même que celle de NEXI. Nous n'y reviendrons donc pas. Nous décrivons uniquement les nouveautés instaurées par ce langage[1].

6.4.2.1 Opérateur de négation

Le signe `'-'` de NEXI est insuffisant pour exprimer qu'un terme ne doit pas apparaître dans les éléments retournés. Nous proposons la négation du prédicat *about*, sémantiquement plus adéquate, et syntaxiquement plus puissante.

Ainsi, l'énoncé 6.3 ci-dessous ne signifie pas que les termes *"tree structure representation"* et *"images"* ne doivent pas apparaître dans les éléments (ce qui serait exprimé par exemple par `-"tree structure representation" -images`). Au contraire, la requête concerne les arbres, et refuser les éléments contenant le premier terme pourrait être très préjudiciable. En revanche les deux termes ne doivent pas être présents ensemble.

(6.3) We are looking for documents presenting approaches for evaluating the structural similarity between two labeled trees. [...] We are not interested in articles dealing with tree structure representation of images. [...][2]

Le signe `'-'` ne permet pas de demander que deux termes ne soient pas utilisés conjointement. En revanche, une expression comme

```
NOT about(., "tree structure representation" images)
```

correspond plus à la réalité du besoin d'information.

[1] Ce langage a été conçu dans le cadre d'INEX 2006, dont nous étions co-organisateur.
[2] Extrait du topic 166, INEX 2004.

6.4.2.2 Requêtes multiples

Nous avons déjà justifié l'utilité d'autoriser des expressions multiples dans la même requêtes. Ceci est permis dans le langage XOR, avec les opérateurs `'AND'`, `'OR'` et `'NOT'` entre des requêtes de formes classiques.

Ainsi, l'exemple de l'article d'Einstein proposé plus haut sera tout simplement traduit en :

```
//article[.//année = 1905
       AND about(.//auteur, Einstein)
       AND about(.//*, électrodynamique)]
  AND
//article[about(., article Einstein 1905)
       AND about(., électrodynamique)
  AND
//article[about(.//année, 1900)
       AND about(., électrodynamique)
```

En pratique, les opérateurs `'AND'` et `'OR'` possèdent exactement la même sémantique. En effet, un moteur de recherche renvoyant une liste d'éléments indépendants les uns des autres, et non pas des groupes d'éléments, les deux conjonctions ont le même sens[1].

6.4.2.3 Extension des chemins et des mots-clés

Nous ajoutons également à NEXI la possibilité de spécifier des contraintes additionnelles concernant la structure ou le contenu. Ces informations optionnelles sont exprimées par une liste de paires attribut-valeur (de la forme `'attribut:valeur'`), séparées par des virgules et entourées par des accolades. Le choix des attributs à gérer par le moteur est libre.

Extension des chemins. Voici des exemples de contraintes pouvant être appliquées sur les chemins d'accès aux éléments structurels :

- Poids : représente l'importance relative des éléments, comme dans l'exemple suivant (où la section `'sec'` est jugée deux fois plus importante que le paragraphe `'p'`).

  (6.4)
  ```
  //article[about(.//sec{poids:2}, systèmes de navigation)
  AND about(.//p{poids:1}, avions)]
  ```

- Interprétation : indique si les exigences structurelles doivent être strictes ou vagues. Ainsi, l'expression 6.5.a demande des éléments sur le tennis, avec une préférence (un meilleur classement) pour les sections, tandis qu'avec la forme 6.5.b, on ne veut que des sections.

  (6.5) a.
  ```
  //sec{inter:vague}[about(., tennis)]
  ```
  b.
  ```
  //sec{inter:strict}[about(., tennis)]
  ```

- Taille : impose une borne minimum ou maximum à la taille (en mots ou en caractères) du contenu des éléments retournés. Cette borne peut permettre de demander des éléments plus larges ou plus précis, selon le besoin.

  (6.6)
  ```
  //*{taille_min:200}[about(., dopage)]
  ```

[1] Par exemple, demander *"des sections et des paragraphes"* ou *"des sections ou des paragraphes"* conduira dans les deux cas à une liste ordonnée contenant des sections et des paragraphes.

Extension des mots-clés. De la même façon, on peut trouver de multiples contraintes à ajouter aux mots-clés :

– Catégorie grammaticale : permet de réduire l'ambiguïté à ce sujet dans la requête :

(6.7) `//article[about(., collection printemps été{cat:nn})]`

– Poids : même sémantique que pour les éléments structurels.

6.4.2.4 Prédicats additionnels

Le prédicat *about* ne suffit pas à représenter l'ensemble des besoins de recherche d'information. Le langage XOR autorise le recours à d'autres prédicats. Nous suggérons l'implémentation des relations suivantes :

– ***contains(., K) :*** version stricte du prédicat *about* (voir section 6.5.2.1).
– ***linkToAbout(., K) :*** l'élément comporte un lien vers un document concernant les mots-clés *K*.
– ***linkFromAbout(., K) :*** un document concernant les mots-clés *K* comporte un lien vers le document recherché.
– ***gt(., N)* et *lt(., N) :*** pour un élément avec du contenu numérique (une année, par exemple), ces prédicats remplacent les opérateurs 'plus grand' et 'plus petit'.

6.4.3 Implémentation

Nous avons contribué à l'adaptation du moteur de recherche GPX, de l'Université Technologique du Queensland, décrit au chapitre 2 [1], pour qu'il traite les requêtes écrites dans le langage XOR. Ceci était indispensable pour évaluer les modifications de notre interface en langage naturel utilisant ce langage.

Les solutions techniques imaginées pour l'élaboration de ce système n'entrent pas dans la cadre de ce mémoire. Les prédicats implémentés sont ceux décrits ci-dessus, et les extensions concernent, jusqu'à présent, le *poids*, le type d'*interprétation*, la *taille* (minimum ou maximum), la *catégorie grammaticale*, ainsi qu'une distinction entre éléments feuilles* et éléments branches.

Cette dernière fonctionnalité est une version plus pragmatique de la taille maximum. Demander une taille maximum revient à courir le risque de limiter fortement les résultats si le niveau de granularité* des documents n'est pas assez fin. Si l'on souhaite obtenir un résultat précis, mieux vaut imposer un élément feuille, c'est-à-dire le plus petit élément possible, ce qui est variable selon les documents.

6.5 Construire des requêtes avec le langage XOR

Les nouvelles fonctionnalités du langage XOR offrent des opportunités pour la prise en compte des différents facteurs introduits dans les premières sections de ce chapitre. Nous décrivons ici comment nous avons modifié notre interface pour profiter au maximum de ces possibilités.

6.5.1 Recherche contextuelle

Les requêtes multiples apportent une solution élégante et simple au problème de la recherche contextuelle décrit à la section 6.1. Elles rendent possible la génération de

[1] Section 2.4.3.2, page 33.

plusieurs constructions ; dans chacune d'entre elles, des termes différents sont désignés comme faisant partie du contexte.

6.5.1.1 Les relations prépositionnelles

Comme avec le langage NEXI, les relations prépositionnelles sont au centre de la construction de requêtes contextuelles en XOR. Lorsque deux syntagmes nominaux NP_1 et NP_2, contenus par un élément structurel E **autre que la racine**, sont liés par une préposition, trois requêtes sont créées : dans la première, NP_1 prend le rôle du contexte, et le texte correspondant est placé comme support dans la racine du document ; dans la deuxième, c'est NP_2 qui devient le contexte, tandis que NP_1 reste dans l'élément E ; enfin, dans la troisième, les deux syntagmes sont placés dans E, sans contexte (requête "classique").

Ainsi si l'on reprend l'exemple du paragraphe sur les systèmes de navigation dans les automobiles, la requête multiple qui est générée est la suivante :

```
//article[about(., "navigation systems")]//p[about(., automobiles)]
     AND
//article[about(., automobiles)]//p[about(., "navigation systems")]
     AND
//article//p[about(., automobiles) AND about(., "navigation systems")]
```

Ceci est la prise en compte directe du problème illustré à la figure 6.1 (page 157).

Des éléments qui obtiennent un score de pertinence non nul pour plus d'une requête voient leurs scores additionnés. Ainsi, la perte de précision* potentiellement occasionnée par l'ajout de formulations moins précises (les deux premières) doit être compensée par le fait que les éléments répondant à la requête "initiale", celle qui ne considère pas de contexte (la troisième) obtiennent un bon score pour les trois requêtes, et donc sont les mieux classés.

Dans le cas des syntagmes comportant plusieurs prépositions (par exemple NP_1 *PREP* NP_2 *PREP* NP_3), toutes les combinaisons mêlant contexte et cible sont proposées, comme illustré dans la figure 6.4 pour deux prépositions. Pour éviter la multiplication des expressions, nous ne générons pas les formes dans lesquelles il y a plus d'une alternance (une alternance = un changement entre contexte et cible), comme c'est le cas pour les formes ③ et ⑥ de la figure[1].

6.5.1.2 La négation

Dans le cas de la négation, il faut au contraire éviter de séparer les différentes parties du syntagme concerné. En effet, comme nous l'avons montré à la section 6.4.2.1, la négation s'applique sur le tout, et non sur chacune des parties. Ainsi, dans le cas des syntagmes de la forme NP_1 *PREP* NP_2 *PREP* NP_3 dans l'élément E, on obtiendra la requête :

```
//E[NOT about(., NP1 NP2 NP3)]
```
[2]

[1] Ce type de formes n'a en effet plus beaucoup de sens en pratique. De plus, cela permet de limiter fortement le nombre d'expressions générées : 5 pour deux prépositions, 7 pour trois, 9 pour quatre, etc.

[2] Cet exemple est donné pour illustrer le principe de la négation, mais une telle requête n'est pas traitable en pratique. La négation doit être un filtre sur des éléments déjà sélectionnés, il est difficile et d'ailleurs sans intérêt de sélectionner des éléments à partir d'une négation sur l'ensemble de la collection (comme *"Je cherche des articles qui ne parlent pas de la Coupe du Monde de football."*).

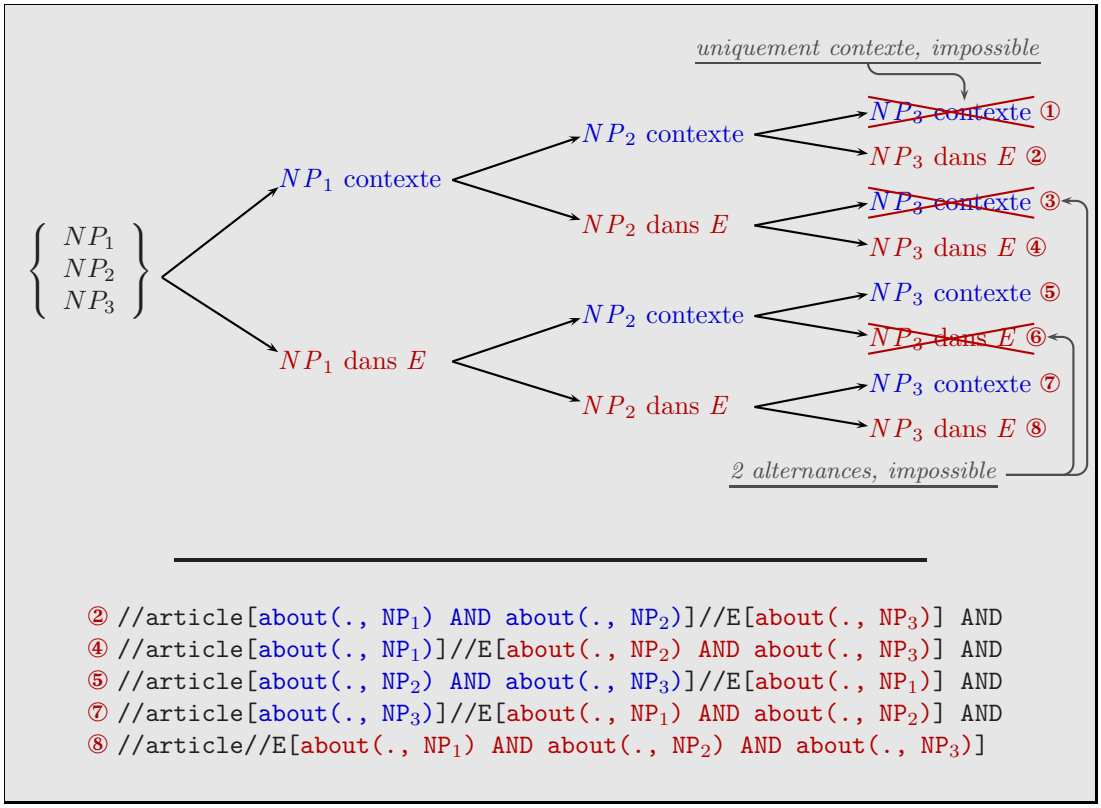


FIG. 6.4 – Recherche contextuelle et relations prépositionnelles multiples.

6.5.2 Autres traitements

6.5.2.1 Le prédicat *contains*

Le prédicat habituel *about* est le prédicat de base de la recherche d'information. Il laisse en fait le champ libre au moteur de recherche pour effectuer toutes sortes de traitement sur les termes. Une instruction plus stricte, demandant une simple recherche exacte de chaînes de caractères, peut être utile dans certains cas. Nous nommons cette relation *contains*. Nous l'avons déjà utilisée à la section 5.3.4.1 [1] ; elle peut s'appliquer par exemple lorsque l'utilisateur a placé des termes entre guillemets[2] :

(6.8) We are looking for articles whose abstracts contain "spatial join".[3]

Le nouvelle règle qui s'applique dans ce cas est celle de la figure 6.5, et la requête en XOR remplace la clause *about* par *contains* :

```
//article[contains(.//abs, "spatial join")]
```

6.5.2.2 Les liens entre documents

Nous avons montré plus haut[4] comment il était possible d'obtenir des relations de type *linkToAbout* avec les règles sémantiques. Il suffit d'utiliser ce même prédicat,

[1] Page 139.
[2] Voir le traitement réservé aux termes entre guillemets à la section 5.2.3.1, page 120.
[3] Extrait du topic 156, INEX 2004.
[4] Section 6.2, page 160.

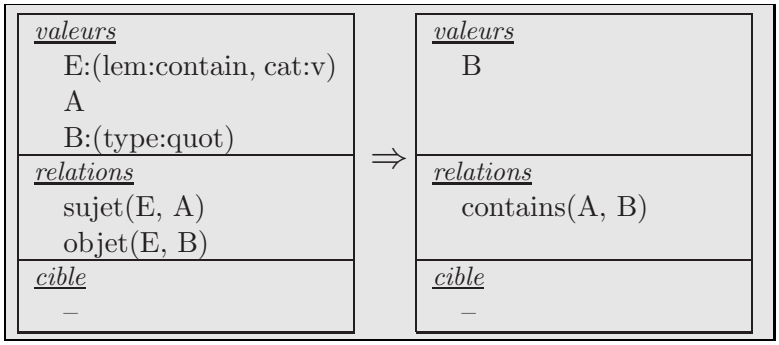


FIG. 6.5 – Introduction du prédicat *contains*.

désormais admis par le moteur de recherche, dans la requête XOR. Ainsi, la requête de l'exemple 6.1 permet d'aboutir à la forme suivante :

```
//bb[linkToAbout(., "text categorisation")
          AND linkToAbout(., Support Vector Machines SVM categoriser)]
```

Notons que le moteur de recherche effectue également une recherche classique (de type *about*) en plus de la recherche des liens.

6.5.2.3 La taille des éléments

Nos idées concernant la limitation de la taille des éléments retournés, exprimées à la section 6.3, peuvent désormais être appliquées grâce à la distinction possible des éléments feuilles. Nous utilisons donc l'attribut `'type:leaf'`, qui demande le renvoi des éléments "les plus petits possibles" (les feuilles), lorsque les requêtes sont de la forme *"who"*, *"whose"*, *"which"*, *"where"*, *"when"*, ainsi que les dérivés de *"how"* : *"how much"*, *"how many"*, *"how old"*, *"how long"*, *"how far"*, etc.[1]

Exemple :

(6.9) Who is candidate to be president in 2007 ?

```
//*{type:leaf}[about(., candidate president 2007)]
```

6.5.2.4 Les ambiguïtés du modèle

Nous avons décrit à la section 5.3.5[2] le problème potentiel des ambiguïtés provoquées par les règles sémantiques de l'interface. La requête multiple nous offre un moyen imparfait mais simple de traiter ces ambiguïtés. En effet, si deux règles s'appliquent à un énoncé précis, provoquant la création de deux éléments différents, il est possible de générer deux constructions différentes, chacune étant le résultat de l'application d'une règle. Nous ne résolvons pas l'ambiguïté, mais proposons les deux possibilités.

[1] Ces requêtes sont souvent des questions, mais les pronoms peuvent être insérés dans le texte avec la même signification (*"I want to know how much..."*).

[2] Page 142.

6.6 Conclusion

Ce chapitre est une étude concernant l'utilisation de notre interface en langage naturel dans le but d'affiner les requêtes générées et d'améliorer la recherche dans la structure des documents.

La première proposition est la mise en œuvre d'une recherche contextuelle qui, en utilisant des indicateurs linguistiques simples, permet de prendre en compte l'imbrication des éléments et la variabilité de l'unité d'information dans les documents XML. Nous avons vu qu'il était possible de réaliser cette recherche contextuelle en utilisant NEXI, mais de façon très imparfaite.

Par ailleurs, deux autres caractéristiques de la recherche semi-structurée ont été étudiés : les liens entre les documents d'une part, et la taille de l'unité d'information recherchée d'autre part. Dans ces cas-là, les limites de NEXI semblent insurmontables.

Au vu de ces difficultés, ainsi que d'autres, exprimées par les chercheurs du domaine, nous avons créé le langage XOR, extension générique de NEXI, rendant possible la spécification de nouvelles fonctionnalités.

Si la recherche contextuelle a pu être testée de façon efficace, l'étude des liens entre documents souffre d'un manque de requêtes évaluées disponibles et pour lesquelles l'utilisation du prédicat *linkToAbout* pourrait apporter une plus-value. En effet, les évaluations d'INEX étant menées avec NEXI, les requêtes que ce langage ne pouvait pas exprimer étaient, *de facto*, évitées. L'encyclopédie Wikipedia, collection unique pour INEX 2006 et comportant beaucoup de liens, devrait permettre des expérimentations plus détaillées.

Enfin, les idées concernant la restriction de la taille des éléments retournés dépassent un peu le cadre de la RI au sens strict du terme, et sont orientées vers des requêtes plus précises, de type question/réponse (domaine dans lequel ces techniques sont d'ailleurs très utilisées, sans les problèmes d'éléments structurels). Nous n'avons donc pas été en en mesure de les tester avec des collections existantes.

Le chapitre suivant détaille l'ensemble des évaluations qui ont pu être menées au sujet de l'interface en langage naturel.

Chapitre 7

Expérimentations et résultats

Et il se fût frotté les mains, s'il eût été dans sa nature de faire un mouvement inutile.
Jules Verne, *Le tour du monde en 80 jours.*

Nous abordons dans ce chapitre l'évaluation des performances de notre système d'analyse des requêtes en langage naturel pour l'interfaçage sur un moteur de recherche d'information dans des documents semi-structurés.

Nos recherches comportent deux buts principaux. Le premier est de montrer qu'il est possible d'utiliser la formulation en langage naturel d'un besoin d'information pour effectuer une recherche dans des documents XML, et ce avec une efficacité comparable à celle obtenue avec une expression formelle de type NEXI. Le second but est d'utiliser la structure des documents pour améliorer encore les requêtes produites, et donc la recherche d'information.

Dans le premier cas, l'évaluation porte sur notre génération de requêtes en langage NEXI, reconnu par de nombreux moteurs de recherche, ceci permettant l'utilisation de notre interface par n'importe quel système ; dans le second, nous utilisons notre langage XOR, implémenté uniquement par le système GPX, mais comportant de nombreuses fonctionnalités supplémentaires.

Ces deux pistes étant nettement distinctes, nous présentons séparément les résultats obtenus dans chacun des cas.

Ce chapitre rappelle brièvement les conditions d'expérimentations proposées par la campagne d'évaluation INEX 2005 (section 7.1) ; il détaille ensuite l'évaluation de notre interface de traduction simple vers NEXI (section 7.2.1), puis décrit l'ajout des techniques de recherche contextuelle avec NEXI (section 7.2.2) et les résultats officiels obtenus à INEX 2005 (section 7.2.3). Enfin, il aborde les performances concernant le langage XOR (section 7.3), avant de donner quelques exemples représentatifs d'analyses complètes (section 7.4).

7.1 Les conditions d'expérimentations

7.1.1 La collection INEX 2005

La collection mise à disposition pour la campagne d'INEX 2005 est composée de :

– 16819 articles concernant tous types de recherches scientifiques en informatique, respectant tous une DTD unique de 192 éléments différents[1].

[1] Un extrait de document est donné à la section 2.5.4.1, page 39.

– 87 requêtes, avec notamment une description en Anglais, un titre composé de mots-clés et un autre titre exprimé en NEXI par l'auteur de chaque requête[1].
– Des jugements de pertinence des éléments des documents par rapport à chacune de ces requêtes. Rappelons que ces jugements sont exprimés par deux dimensions : l'*exhaustivité**, représentée par une valeur de 0, 1 ou 2, et la *spécificité*, valeur réelle entre 0 et 1[2].

De plus, des métriques permettant d'évaluer les performances d'un système et de comparer les systèmes entre eux sont proposées[3].

Les premiers tests et l'"entraînement" de notre système ont été effectués sur la collection 2004 d'INEX, comportant un nombre moins important de documents et 74 requêtes (différentes de celles de 2005). C'est pourquoi nous ne donnons ici que les résultats obtenus avec la collection 2005.

7.1.2 La tâche CO+S

Dans la tâche CO+S, les requêtes, au nombre de 40, ne concernent que le contenu des documents, et pas leur structure. Le but du système est alors de retourner les éléments pertinents tout en déterminant le niveau de granularité approprié.

Trois sous-tâches sont associées à ces requêtes :

– dans la variante *Focussed*, le système ne doit pas retourner d'éléments se chevauchant. Il doit donc déterminer le meilleur élément dans un chemin donné.
– pour la sous-tâche *Thorough*, les chevauchements sont autorisés.
– la stratégie *Fetch and Browse*, plus orientée vers l'utilisateur, consiste à retourner un document pertinent entier, puis à lister les éléments intéressant de cet article (ceci simulant le renvoi d'un article avec des passages "surlignés", mis en évidence).

Un exemple de requête CO+S est donné à la figure 2.12, page 42.

7.1.3 La tâche CAS

La tâche CAS, avec 47 topics, étudie la capacité des systèmes à gérer à la fois le contenu et la structure des documents, à travers des requêtes comportant les deux aspects.

Pour chacune de ces requêtes, quatre évaluations distinctes sont menées, relatives à l'interprétation faite des contraintes structurelles :

– Dans la sous-tâche SSCAS, les indications structurelles concernant la cible* et le support* doivent être interprétées de façon stricte. Ainsi, si le titre en NEXI demande une figure, seule une figure pourra obtenir une score d'exhaustivité non nul.
– Avec SVCAS, le support pourra être interprété de façon vague. Ceci signifie que l'indication structurelle doit être considérée comme un indice, mais qu'un autre type d'éléments pourra être valable également.
– Pour la configuration, VSCAS, c'est la cible qui est vague, et le support strict.
– Enfin, VVCAS autorise une interprétation vague de l'ensemble des contraintes structurelles.

Des exemples de requêtes CAS sont fournis aux figures 2.13 et 2.14, page 43.

[1] Ces requêtes sont décrites à la section 2.5.4.2, page 39.
[2] Voir la section 2.5.4.3, page 43.
[3] Section 2.5.4.4, page 44.

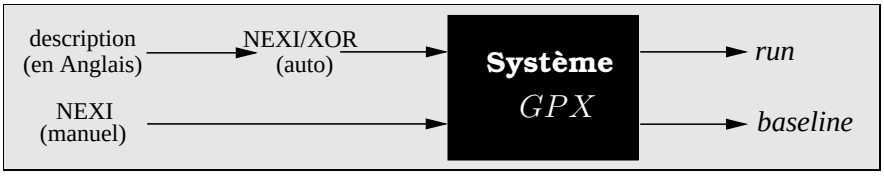


FIG. 7.1 – La méthode d'évaluation.

7.1.4 La méthode d'évaluation

Dans chaque cas, nous comparons les performances du système sur le principe de la tâche NLQ2NEXI, rappelé à la figure 7.1 : la description en Anglais est l'entrée de notre interface, qui génère du NEXI ou du XOR. Celui-ci ainsi que le titre de l'auteur du topic sont appliqués à la collection par le moteur de recherche GPX, pour donner deux listes de doxels*. Ce sont ces deux listes qui font l'objet d'une évaluation et d'une comparaison.

7.1.5 Les métriques d'évaluation

Les métriques d'évaluation utilisées dans l'ensemble des descriptions de nos expérimentations sont les métriques officielles de la campagne INEX 2005, à savoir le gain cumulé étendu normalisé (nxCG) et l'effort-précision/gain-rappel (ep-gr). Ces deux métriques sont décrites à la section 2.5.4.4, page 44.

Les courbes que nous présentons sont celles reportées lors de la présentation officielle des résultats d'INEX, à savoir :

- la courbe obtenue pour l'ensemble des valeurs du gain cumulé, avec le nombre de documents variant de 1 à 1500, agrégation généralisée.
- la courbe ep/gr, agrégation généralisée.

7.2 Traduction en NEXI

Les premières expérimentations présentées concernent la méthode basique de traduction de la requête en NEXI, de la façon décrite au chapitre 5. La comparaison entre la "baseline" et le résultat de l'interface ("NEXI simple") est illustrée aux figures 7.2 à 7.8.

Les courbes concernant la recherche contextuelle sont déjà présentées (courbes "XOR multiples" et "NEXI contexte"). Cela nuit au suspense mais permet une présentation des résultats mieux agrégée. En effet, le moteur de recherche et la "baseline" étant les mêmes, il est intéressant de tout comparer dans la même figure.

7.2.1 Traduction simple

7.2.1.1 Les requêtes CAS

Les performances réalisées par le moteur de recherche GPX pour les requêtes CAS sont données par les figures 7.2 à 7.5. On y constate que les résultats obtenus avec notre interface approchent et souvent dépassent les résultats manuels.

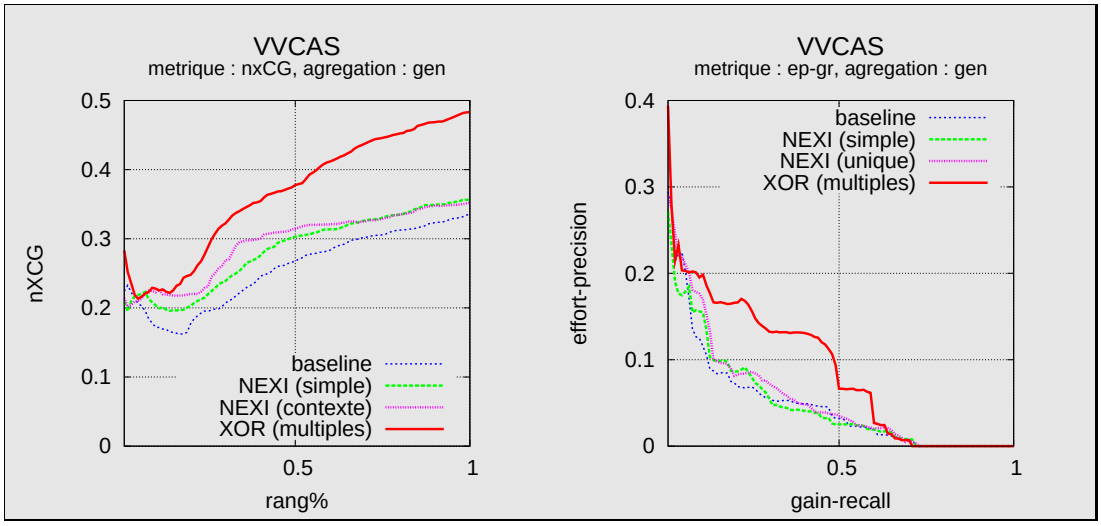


FIG. 7.2 – Courbes pour la tâche VVCAS, agrégation généralisée, requêtes en NEXI et en XOR (expressions multiples).

Aux premiers rangs, les courbes “baseline” et “NEXI simple” s’entrecroisent et conservent un niveau comparable. La courbe de l’interface se détache ensuite relativement nettement au-dessus de la “baseline” et suit une évolution parallèle à celle-ci. L’écart est similaire dans toutes les sous-tâches.

Les graphes ep/gr, en revanche, montrent des courbes globalement superposées (ou sont légèrement en faveur de l’interface).

Le nombre total d’éléments pertinents retournés n’est pas changé (dans chaque cas, les deux courbes d’effort-précision s’éteignent au même niveau de gain-rappel). En revanche, les expressions automatiques permettent de mieux placer les doxels pertinents (courbes d’effort-précision et de gain un peu au-dessus[1]).

Un résultat particulièrement intéressant est celui de la tâche SSCAS (figure 7.5), pour laquelle les contraintes structurelles sont strictes. Les bonnes performances de l’interface prouvent que les aspects structurels des requêtes (relations entre éléments et contenu, allusions implicites) sont bien maîtrisés.

7.2.1.2 Les requêtes CO+S

En ce qui concerne les requêtes C0+S, dont les résultats sont affichés aux figures 7.6 à 7.8, les résultats sont plus mitigés. La courbe de l’interface est toujours proche de la “baseline”, mais reste en général en-dessous, que ce soit pour le gain normalisé ou pour l’effort-précision.

La raison de cette différence est que les requêtes ne concernant pas la structure sont souvent plus longues et contiennent plus de mots n’ayant pas un rapport direct avec le sujet. L’interface ne filtre alors pas assez bien et conserve un certain nombre de mots peu discriminants, qui brouillent la requête. Ce phénomène est bien visible lorsqu’on observe les requêtes[2]. Toutes les approches que nous avons mises en place

[1] Rappelons qu’avec l’agrégation généralisée, la valeur de pertinence est une valeur réelle continue, et non pas une valeur binaire, et que par conséquent le gain augmente plus vite si les éléments retournés sont *plus* pertinents.

[2] cf. Annexe D.

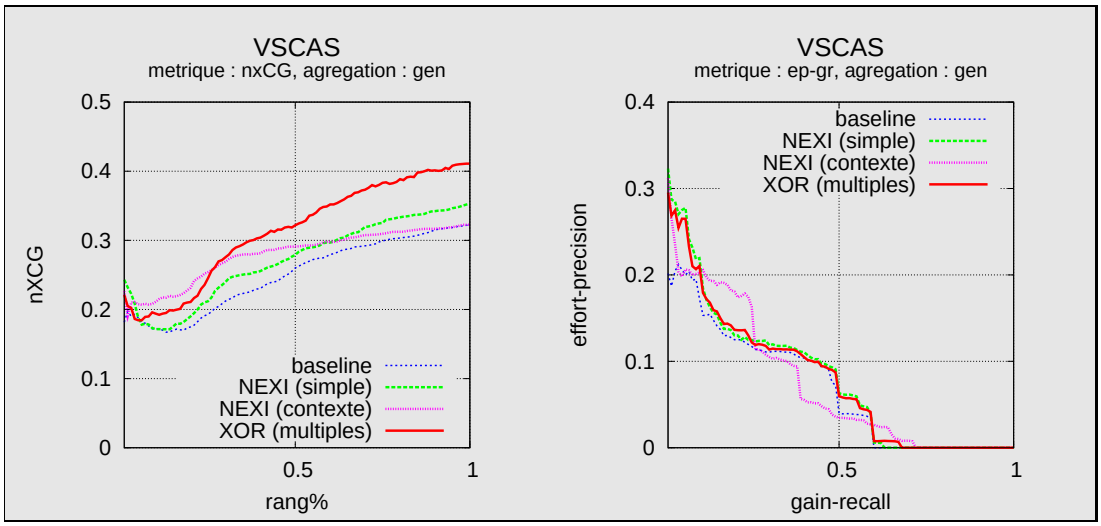


FIG. 7.3 – Courbes pour la tâche VSCAS, agrégation généralisée, requêtes en NEXI et en XOR (expressions multiples).

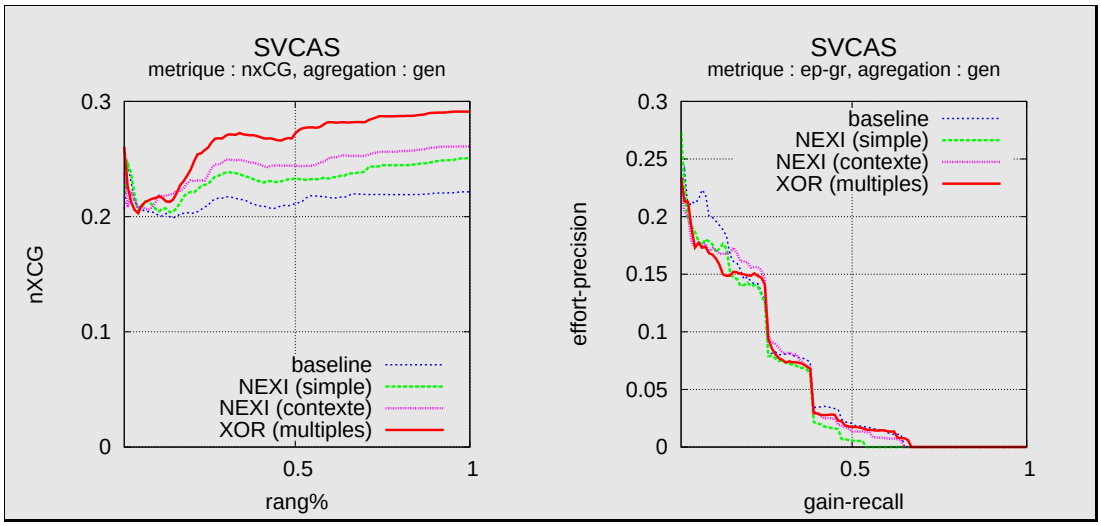


FIG. 7.4 – Courbes pour la tâche SVCAS, agrégation généralisée, requêtes en NEXI et en XOR (expressions multiples).

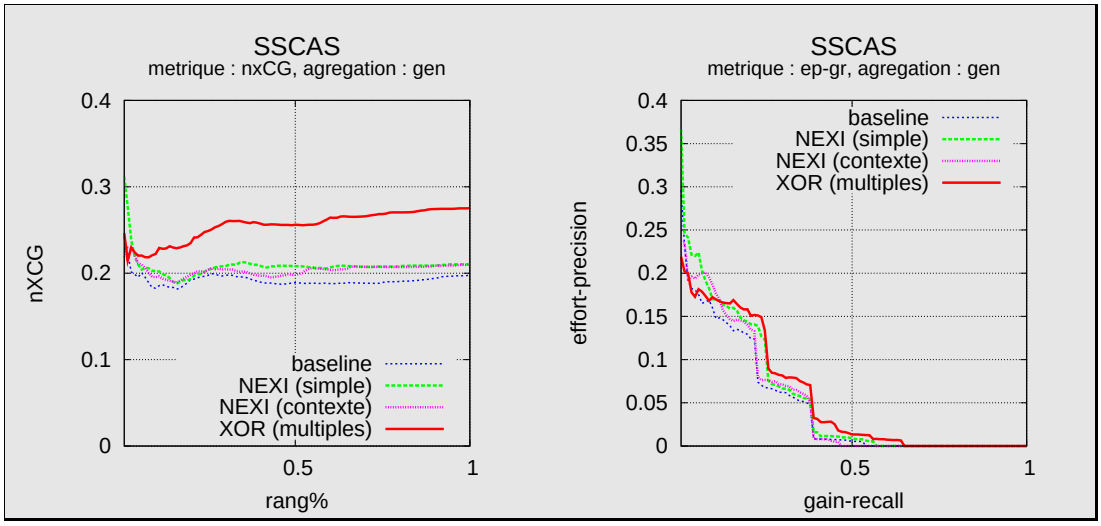


FIG. 7.5 – Courbes pour la tâche SSCAS, agrégation généralisée, requêtes en NEXI et en XOR (expressions multiples).

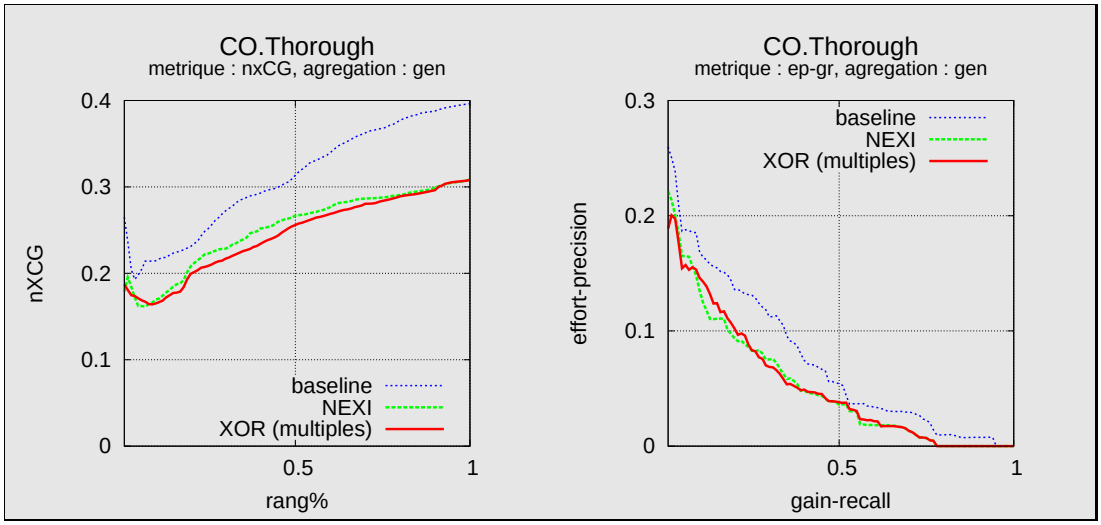


FIG. 7.6 – Courbes pour la tâche CO+S.Thorough, agrégation généralisée, requêtes en NEXI simple et en XOR (expressions multiples).

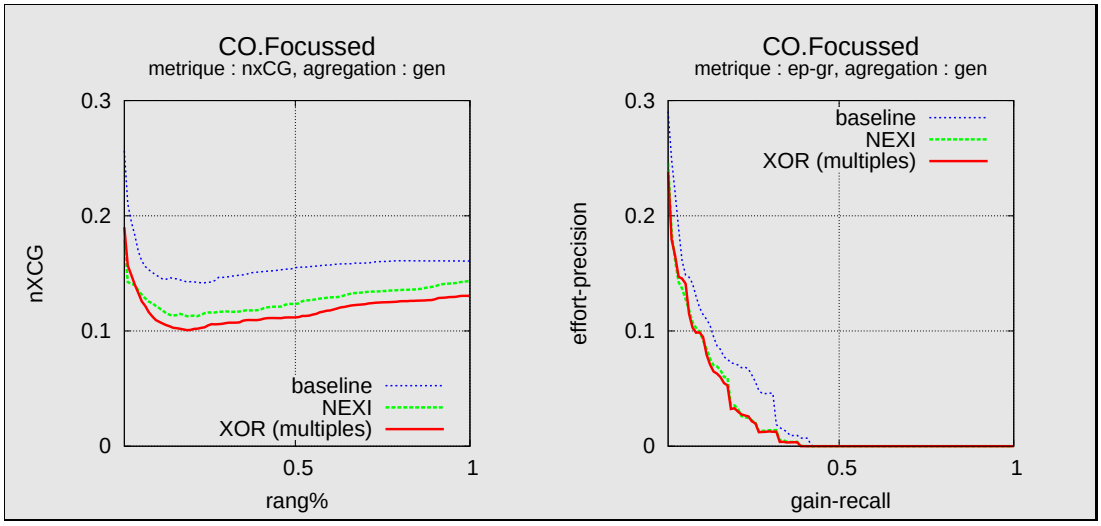


FIG. 7.7 – Courbes pour la tâche CO+S.Focussed, agrégation généralisée, requêtes en NEXI simple et en XOR (expressions multiples).

pour mettre au point notre interface sont orientées vers l'étude des liens structurels, et nous avons mis en œuvre peu de techniques orientées vers le contenu textuel seul. Parmi celles-ci, le regroupement entre guillemets des groupes nominaux simples, exposé à la section 5.2.3.1 [1], est un point important dont l'apport peut être évalué indépendamment.

7.2.1.3 Le regroupement des groupes nominaux simples

Rappelons que certains syntagmes nominaux sont regroupés pour être placés entre guillemets dans les requêtes NEXI ou XOR. Il s'agit des groupes obéissant aux deux règles suivantes :

$$NP \rightarrow (JJ \mid NN)+ \; NN \tag{7.1}$$

[1] Page 120.

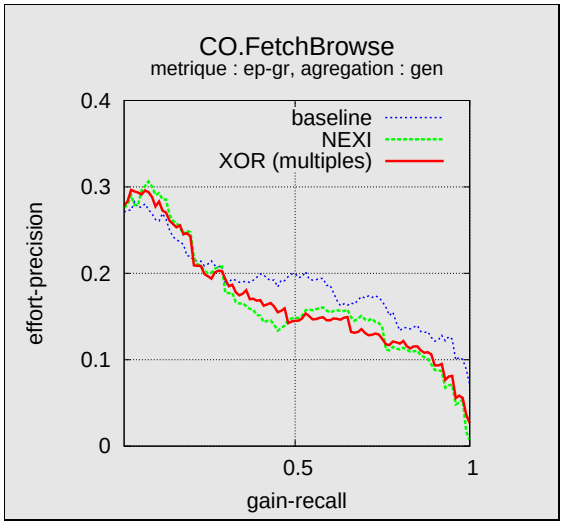


FIG. 7.8 – Courbes pour la tâche CO+S.FetchBrowse, agrégation généralisée, requêtes en NEXI simple et en XOR (expressions multiples).

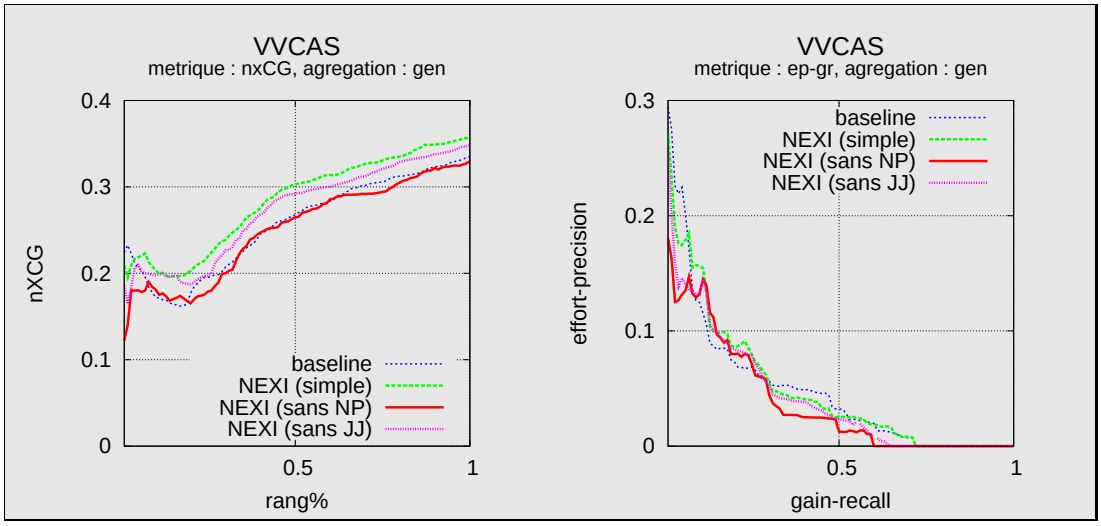


FIG. 7.9 – Tâche VVCAS, étude de l'influence du regroupement en groupes nominaux simples.

$$NP \rightarrow PN\ (IN?\ PN)+ \tag{7.2}$$

Pour connaître la part de cette méthode dans le comportement de l'interface, notamment en comparaison de la *baseline*, nous avons simplement évalué un nouveau *run* en désactivant l'ajout des guillemets. Nous n'avons conservé, par souci de simplicité, qu'une sous-tâche pour les requêtes CAS (figure 7.9) et une autre pour les requêtes CO (figure 7.10). Les résultats pour les autres sous-tâches de chaque type sont similaires.

Pour les requêtes CAS, le bénéfice apporté par le regroupement des syntagmes nominaux est évident. En l'absence de guillemets dans les requêtes automatiques (courbe "NEXI sans NP"), les performances de l'interface retombent environ au même niveau que celles des experts humains, ce qui représente une perte importante.

En revanche, l'avantage est beaucoup moins clair s'agissant des requêtes CO, même

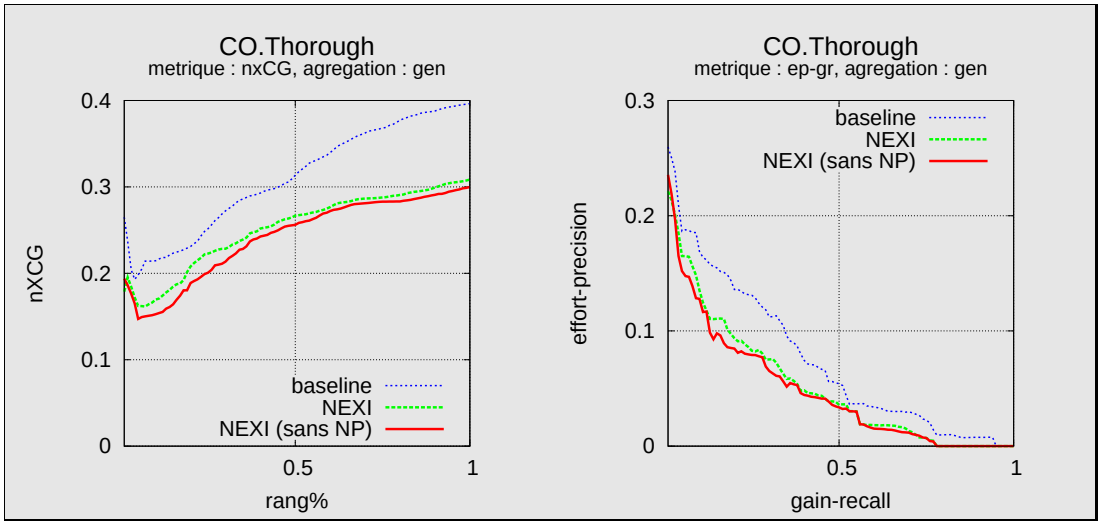


FIG. 7.10 – Tâche COS Thorough, étude de l'influence du regroupement en groupes nominaux simples.

Regroupements pertinents		*Regroupements non pertinents*	
Topic	*Groupe*	*Topic*	*Groupe*
204	"semantic networks"	205	"current digital technologies"
222	"electronic commerce"	206	"physical limits"
229	"latent semantic analysis"	221	"commercial speech
229	"latent semantic indexing"		recognition software"
229	"differential geometry"	232	"graph-related algorithms"
237	"articifial intelligence"	232	"expressive experiment results"
253	"digital libraries"	269	"interconnected networks"
259	"intellectual property"		
275	"frequent itemsets"		

TAB. 7.1 – Exemples de regroupements de syntagmes nominaux avec des adjectifs.

si les courbes avec regroupement sont légèrement au-dessus des courbes sans regroupement.

Par ailleurs, l'intégration des adjectifs dans les règles mérite une analyse plus approfondie. Nous avons donc tenté d'évaluer sa pertinence en supprimant les adjectifs (JJ) de la règle 7.1. Nous obtenons la courbe "NEXI sans JJ" de la figure 7.9, qui montre que la considération des adjectifs a elle aussi un effet positif ; nous n'avons pas reporté cette courbe sur la figure 7.10, car elle se situait de façon peu lisible entre "NEXI" et "NEXI sans NP", avec des différences peu significatives.

Cette dernière expérimentation globale concernant les adjectifs ne doit cependant pas occulter le fait qu'au niveau individuel, le regroupement "JJ NN" n'est pas toujours pertinent, comme l'indiquent quelques exemples choisis dans l'ensemble de requêtes d'INEX 2005 et présentés à la table 7.1 ; certains d'entre eux influent de façon négative sur la recherche. Même si nous n'avons pas poussé l'étude plus loin, il est probable que des techniques sémantiques simples pourraient permettre de sélectionner les groupes nominaux avec plus de discernement.

Précisons enfin que le système GPX ne recherche pas les groupes entre guillemets de façon stricte. Il sépare tout d'abord chaque mot du groupe et attribue un score à

l'élément en fonction de la présence de ces mots. Il ajoute ensuite un bonus si le groupe est présent (avec une certaine distance autorisée entre les mots). Ce traitement ne rend ainsi pas trop restrictive l'utilisation des guillemets ; les conclusions de cette section ne sont donc applicables qu'à ce type de systèmes.

7.2.1.4 La désambiguïsation syntaxique par les règles sémantiques

Le processus de désambiguïsation syntaxique à l'aide des règles sémantiques, décrit à la section 5.3.6 [1], a un poids important dans le choix de la requête finale pour les requêtes CAS. C'est ce que montre l'analyse des 81 requêtes CAS proposées en 2004 et 2005 à INEX. Dans 50 % des cas, la forme syntaxique préférée est modifiée par la contrainte du nombre de règles sémantiques déclenchées (voir section 5.3.6.1). Par ailleurs, les règles de préférence entre élément et contenu (section 5.3.6.2) s'appliquent dans 53 % des cas et celle stipulant qu'un élément doit avoir du contenu (section 5.3.6.3) dans 55,5 % des cas. Souvent ces différentes techniques s'appliquent à la même phrase et avec les mêmes conclusions.

Voici quelques nouveaux exemples typiques des apports de cette désambiguïsation. En ce qui concerne le contenu des éléments, l'exemple 7.3 montre que l'attachement de *"containing"* à *"figure-captions"* est remplacé par un attachement au NP coordonné *"paragraphs or figure-captions"*, faute de quoi les paragraphes resteraient sans contenu.

(7.3) Find [paragraphs] or [figure-captions] [containing] the definition of ... [2]

L'exemple 7.4 est intéressant car le système y tient compte de contraintes de NEXI pour parvenir à une analyse peu intuitive, mais en accord avec l'auteur de la requête. Dans ce cas, l'auteur propose la forme `'//article//p[about(.,watermarking)]'`. L'interface en fait de même à l'issue du processus ; ceci vient du fait que NEXI n'autorise pas de cible sans contenu, ce que nous aurions obtenu en considérant que les *articles*, et non les *paragraphes*, contenaient le mot *"watermarking"*.

(7.4) We are looking for [paragraphs] in [articles] [which contain] watermarking. [3]

En ce qui concerne le nombre de règles utilisées, il permet très souvent, en plus des exemples donnés à la section 5.3.6, d'éviter un attachement au verbe d'un groupe prépositionnel :

(7.5) We are [looking] for [the abstracts] [of the documents] ... [4]

Toutes ces règles de désambiguïsation, si elles ne permettent pas toujours de parvenir

[1] Page 145.
[2] Extrait du topic 127, INEX 2004.
[3] Topic 271, INEX 2005.
[4] Extrait du topic 131, INEX 2004.

à une traduction parfaite de la requête, vont dans le bon sens dans la très grande majorité des cas.

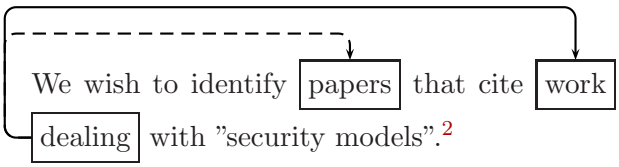
Nous pouvons citer deux contre-exemples. Dans le premier (7.6), le mot *"references"* est reconnu comme une citation bibliographie, ce qui est pertinent dans tous les autres cas. Ici, l'auteur n'utilise pas ce mot dans ce sens, comme le montre sa propre traduction (7.6.b). La reconnaissance erronée d'un élément bibliographique conduit bien entendu à l'emploi des règles sémantiques sur des bases fausses et à une requête relativement éloignée, structurellement parlant, du besoin de l'utilisateur (7.6.c).

(7.6) a. Find references that are about Dempster-Shafer Application, where the application is modeled by Dempster-Shafer theory...[1]

b. `//article[about(.//abs, Dempster-Shafer theory)]//sec[about(., Dempster Shafer database experiment)]`

c. `//article//bb[about(., "Dempster-Shafer Application") AND about(., model Dempster-Shafer theory)]`

Le second exemple d'erreur d'analyse (7.7) vient du fait que l'analyse syntaxique attache *"dealing"* à *"papers"* (7.7.c), tandis que l'auteur voulait signifier un attachement à *"work"* (7.7.b). Les mêmes règles sémantiques s'appliquent dans les deux cas, et ne permettent donc pas de réparer l'erreur.

(7.7) a. 

b. `//article[(about(.//bb//au//snm, Bertino) or about(.//bb//au//snm, Jajodia)) and about(.//bb//atl, security model)]`

c. `//article[about(., "security models") AND (about(.//bb//au, Bertino) OR about(.//bb//au, Jajodia))]`

7.2.1.5 Conclusion

L'évaluation de notre système de génération de requêtes NEXI est très positive, puisqu'elle démontre que l'analyse de la requête en langage naturel est une alternative intéressante à la construction d'expressions formelles complexes.

Elle confirme également une hypothèse qui a dirigé notre travail, à savoir que la structure des documents et leur utilisation dans les requêtes peuvent être un support efficace pour l'analyse des descriptions du besoin d'information. En effet, les résultats concernant les topics faisant allusion à la structure sont excellents, tandis que ceux orientés vers le contenu uniquement sont moins décisifs.

Pour tenter de comprendre pourquoi les résultats obtenus par l'interface simple peuvent être meilleurs que ceux des requêtes manuelles, il est important de remarquer que les utilisateurs effectuent eux-mêmes une traduction mentale de leur besoin "exprimé" en langage naturel vers NEXI. Or, leur traduction peut être imparfaite, avec des changements de mots-clés ou des imprécisions. Nous pouvons prendre l'exemple du topic 132, dont la description est la suivante :

(7.8) We are interested in finding sections about experiment design and experiment results of different classification methods. We hope that the abstract should mention the classification methods.

[1] Extrait du topic 232, INEX 2005.
[2] Extrait du topic 139, INEX 2004.

L'expression NEXI proposée par l'auteur est :

```
//article[about(.//abs, classification)]
            //sec[about(., experiment compare)]
```

où le terme *"compare"*, très fréquent et donc générateur de bruit, en remplace d'autres ; en comparaison, notre interface propose :

```
//article[about(.//abs, classification)]
          //sec[about(., "experiment design" "experiment results")
                AND about(., classification)]
```

Une autre raison est que les humains n'utilisent pas toujours les regroupements de termes entre guillemets à bon escient. Nous avons d'ailleurs montré à la section précédente que l'utilisation automatique des guillemets avait un effet très positif sur les résultats. C'est par exemple le cas d'une requête sur les systèmes d'exploitation et les threads, traduite ainsi par son auteur :

```
//article[about(., operating system)
            AND about(.//sec, thread implementation)]
```

Les termes séparés *"operating"* et *"system"* sont très fréquents, tandis que leur occurrence conjointe a une sémantique très spécifique. Ceci est exprimé par l'interface[1] :

```
//article[about(., "operating systems")
            AND about(.//sec, implement threads)]
```

Le problème est inversé lorsque le besoin est plus vague et la description plus longue (CO) :

(7.9) We are looking for documents presenting approaches for evaluating the structural similarity between two labeled trees. We are mainly interested in approaches that could be applied to XML documents. We are not interested in articles dealing with tree structure representation of images and approaches related to the evaluation of the edit distance between two strings.

Dans ce cas-là l'humain est plus apte que l'interface à sélectionner les mots-clés importants :

```
//*[about(., +"tree edit distance" +XML -image)]
```

tandis que l'interface a tendance à multiplier les mots-clés[2] :

[1] La requête en langage naturel étant : *"We are seeking articles about operating systems with sections discussing how they implement threads."*

[2] Voir à la section 5.4.2.5, page 152, une explication du problème spécifique de la négation d'un groupe nominal complexe posé par cette requête.

```
//*[about(., evaluating "structural similarity")
      AND about(., "labeled trees")
      AND about(., "XML documents")
      AND about(., -"tree structure representation image")
      AND about(., -"evaluation edit distance strings")]
```

Dans ce cas le résultat semble approprié, mais l'auteur humain a utilisé une formulation plus concise et probablement plus proche du besoin réel.

Dans d'autres cas, le système fournit clairement des mots-clés non pertinents, car trop généraux, comme pour la requête suivante :

(7.10) Find references that are about Dempster-Shafer Application, where the application is modeled by Dempster-Shafer theory, and it is a Database related problem (not AI, IR), and the solution is shown with expressive experiment results.

```
//*[about(., "expressive experiment results" solution)
      AND about(., -AI -IR)
      AND about(., "Dempster-Shafer Application")
      AND about(., modeled)
      AND about(., application)
      AND about(., theory)
      AND about(., Database Dempster-Shafer)]
```

C'est dans ce genre de situations que les performances sont les moins bonnes.

7.2.2 NEXI et recherche contextuelle

Les résultats présentées dans cette section concernent la technique de recherche contextuelle avec NEXI, destinée à augmenter le rappel, tout en y adjoignant la recherche conditionnelle pour ne pas faire chûter la précision. La section 6.1.2.3 [1] décrit en détails cette méthode.

Les figures 7.2 à 7.5, détaillant les résultats pour les requêtes CAS, comportent une courbe "NEXI contexte" obtenue avec cette technique. On constate une légère amélioration du gain par rapport à la courbe de l'interface simple ("NEXI simple"), à l'exception des requêtes SSCAS, où les courbes sont quasiment confondues. L'effort-précision n'est pas sensiblement modifié. Cette progression est satisfaisante, étant donnée la faible marge de manœuvre autorisée par les restrictions de NEXI.

Au niveau des requêtes sur le contenu seulement (CO+S), nous n'avons pas ajouté la courbe "NEXI contexte" pour la simple raison que cette courbe était quasiment confondue, pour l'ensemble des mesures, avec celle de l'interface vue à la section précédente ("NEXI simple"). Nous n'avons donc pas surchargé inutilement les graphiques. Ainsi, la recherche contextuelle ne semble pas améliorer la recherche lorsque le besoin ne concerne que le contenu.

7.2.3 Les résultats officiels de la campagne INEX 2005

Les résultats que nous détaillons dans cette section sont les résultats officiels acquis à l'issue de la campagne d'évaluation d'INEX 2005 pour la tâche NLQ2NEXI. Ils sont

[1] Page 159.

donc obtenus sur des requêtes nouvelles dont les réponses sont inconnues au moment de la soumission.

Ils sont difficilement comparables aux autres séries de résultats, car obtenus avec un moteur de recherche différent et une interface à un point différent de son développement. Il nous a cependant semblé important de présenter ces résultats pour leur caractère officiel. La technique utilisée est celle de la recherche contextuelle avec NEXI, mais avec une analyse sémantique moins avancée.

Nous présentons donc une série de graphiques distincts, ainsi que les tableaux de valeurs du gain cumulé après 10 documents ($ncCG[10]$), 25 documents ($ncCG[25]$) et 50 documents ($ncCG[50]$) retrouvés. Nous ne les avons pas considérées dans nos autres résultats car ces valeurs nous semblent peu pertinentes. En effet, elles apportent peu d'information supplémentaire (puisqu'elles ne sont que des points particuliers des courbes) et sont extrêmement fluctuantes, surtout dans notre cas (comme nous l'avons déjà indiqué, les courbes varient beaucoup aux premiers rangs). En effet, si l'on ne considère que quelques éléments, l'ajout d'un seul nouvel élément (pertinent ou non) modifie fortement le score moyen. Cependant, ces mesures font partie de l'évaluation officielle, nous les indiquons donc ici.

7.2.3.1 Les requêtes CAS

L'évaluation pour l'ensemble de requêtes faisant des références à la structure (CAS) est donnée dans les tables 7.2 et aux figures 7.11 à 7.14.

Les meilleures performances sont observées pour les sous-tâches VVCAS et VSCAS, c'est-à-dire pour des interprétations vagues des indications structurelles ; ceci n'est pas très surprenant, puisque notre mise en œuvre de la recherche contextuelle implique que les contraintes structurelles deviennent plus souples, avec des mouvements des mots-clés entre la cible et le support notamment.

Malgré cela, le ratio reste bon, entre 80 et 100 %, pour la sous-tâche SSCAS. On remarque surtout une grosse faiblesse de précision dans les premiers niveaux de rappel (figure 7.14), mais ensuite les courbes *mines* et *baseline* ont une allure similaire.

Pour VVCAS et VSCAS, les résultats sont globalement supérieurs avec l'interface, en particulier pour l'agrégation généralisée ; en revanche, pour l'agrégation stricte, les ratios sont toujours autour de 1, mais sont un peu moins bons. Ceci montre que notre système permet de mieux classer les éléments pertinents, mais ne contribue pas particulièrement à placer les éléments ***très*** pertinents devant les autres.

7.2.3.2 Les requêtes CO+S

En ce qui concerne la tâche CO+S, dont les résultats sont donnés aux tables 7.3 et aux figures 7.15 à 7.17, les scores sont moins convaincants. S'ils s'approchent toujours de la *baseline* pour l'agrégation généralisée (entre 80 et 100 %), ils sont parfois beaucoup plus bas pour l'agrégation stricte.

7.2.3.3 Conclusion

L'application de la recherche contextuelle au langage NEXI permet une légère amélioration des résultats en ce qui concerne la tâche CAS. Elle ne semble pas modifier les résultats de la tâche CO.

a. VVCAS			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	0.1222	0.1111	0.91
Gen	0.2520	0.2658	1.05
nxCG[25]			
Strict	0.1267	0.1867	1.47
Gen	0.2281	0.2572	1.13
nxCG[50]			
Strict	0.1142	0.2282	2.00
Gen	0.2080	0.2253	1.08
ep-gr			
Strict	0.0454	0.0418	0.92
Gen	0.0694	0.0799	1.15

b. VSCAS			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	0.1167	0.1000	0.86
Gen	0.2039	0.2662	1.31
nxCG[25]			
Strict	0.1267	0.1133	0.89
Gen	0.2531	0.3051	1.21
nxCG[50]			
Strict	0.1200	0.1900	1.58
Gen	0.2493	0.2784	1.12
ep-gr			
Strict	0.0383	0.0363	0.95
Gen	0.0608	0.0682	1.12

c. SVCAS			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	0.0400	0.0400	1.00
Gen	0.0848	0.0968	1.14
nxCG[25]			
Strict	0.0662	0.0662	1.00
Gen	0.1081	0.1004	0.93
nxCG[50]			
Strict	0.0662	0.0662	1.00
Gen	0.1086	0.0991	0.91
ep-gr			
Strict	0.0274	0.0276	1.01
Gen	0.0272	0.0298	1.10

d. SSCAS			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	—	—	–
Gen	0.2517	0.2040	0.81
nxCG[25]			
Strict	0.1578	0.1378	0.87
Gen	0.2885	0.2593	0.90
nxCG[50]			
Strict	—	—	–
Gen	0.3681	0.2595	0.70
ep-gr			
Strict	0.0770	0.0775	1.01
Gen	0.1324	0.1064	0.80

TAB. 7.2 – Résultats pour la tâche CAS, INEX 2005.

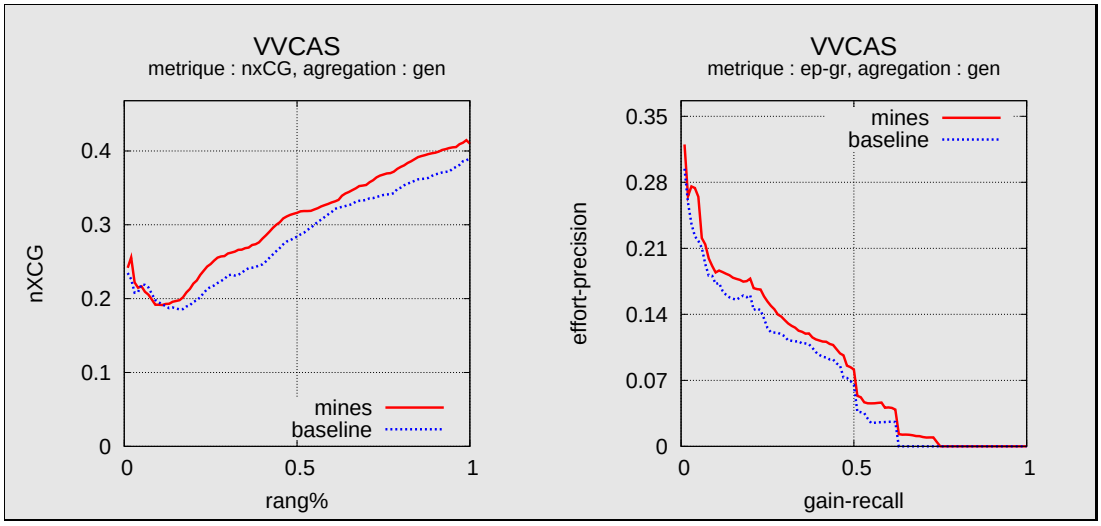


FIG. 7.11 – Courbes pour la tâche VVCAS, agrégation généralisée, INEX 2005.

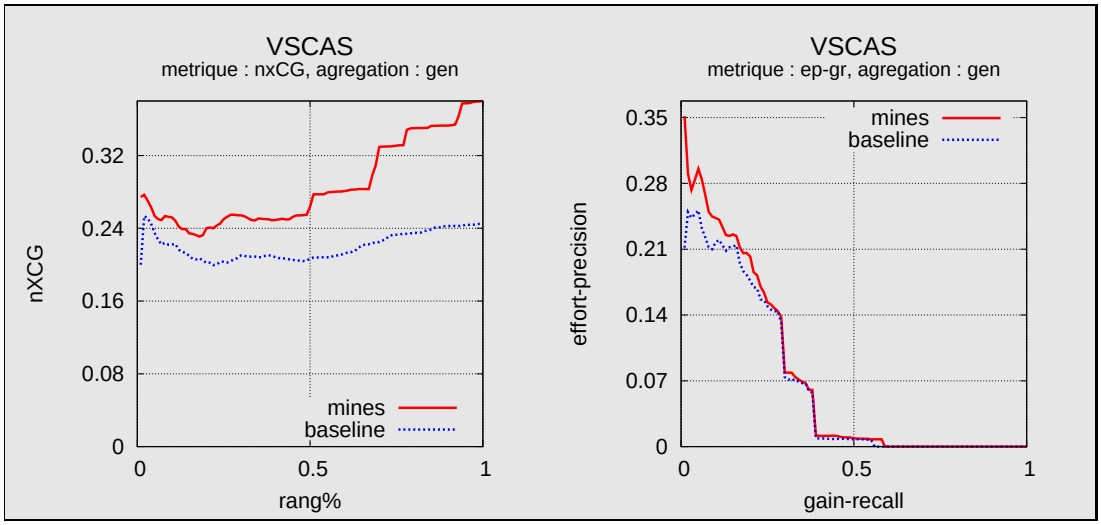


FIG. 7.12 – Courbes pour la tâche VSCAS, agrégation généralisée, INEX 2005.

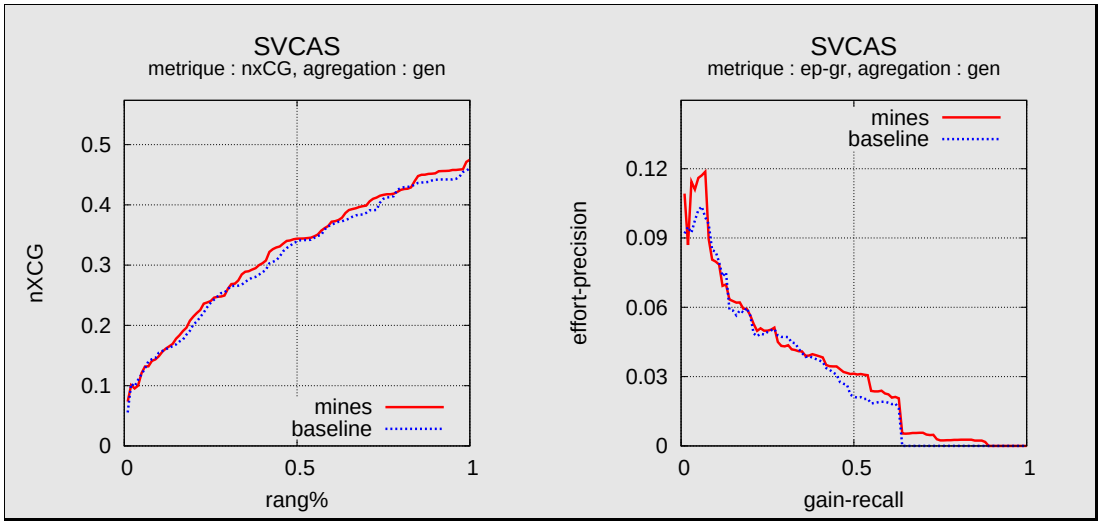


FIG. 7.13 – Courbes pour la tâche SVCAS, agrégation généralisée, INEX 2005.

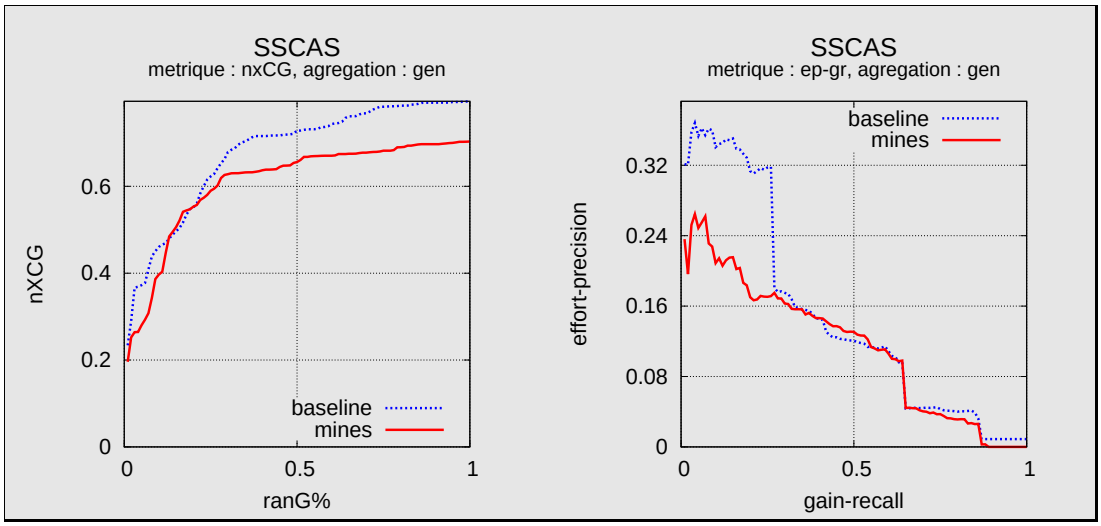


FIG. 7.14 – Courbes pour la tâche SSCAS, agrégation généralisée, INEX 2005.

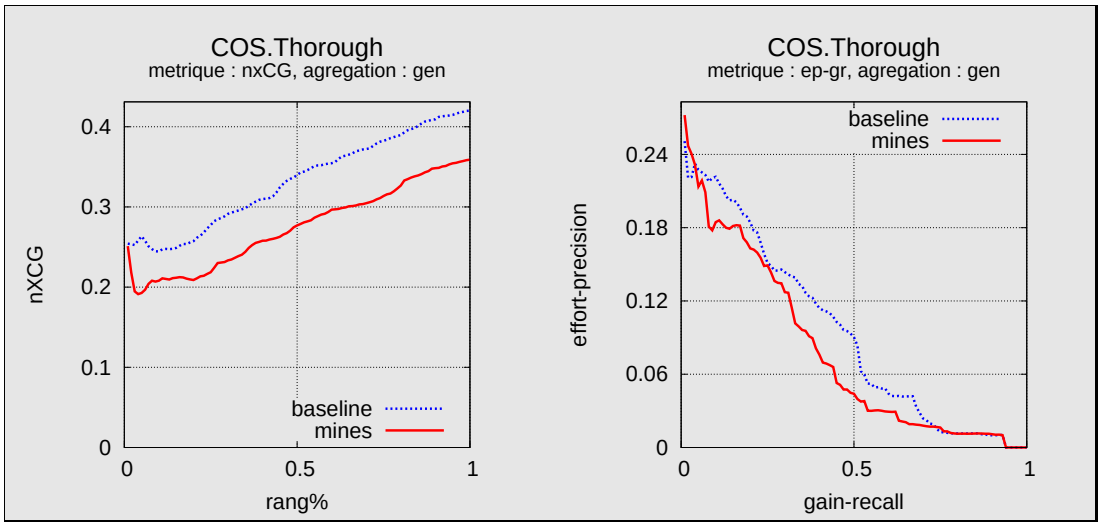


FIG. 7.15 – Courbes pour la tâche CO+S.Thorough, agrégation généralisée, INEX 2005.

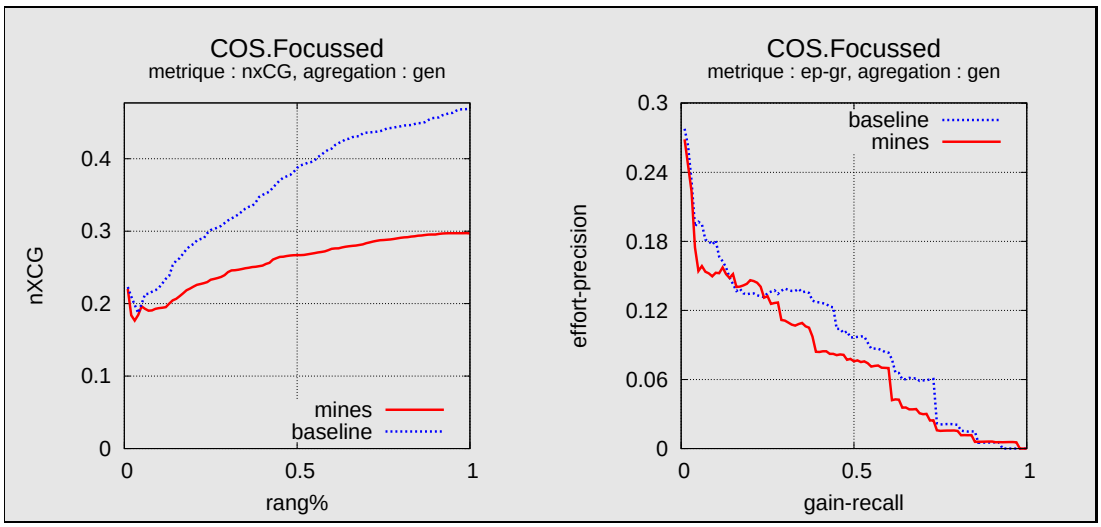


FIG. 7.16 – Courbes pour la tâche CO+S.Focussed, agrégation généralisée, INEX 2005.

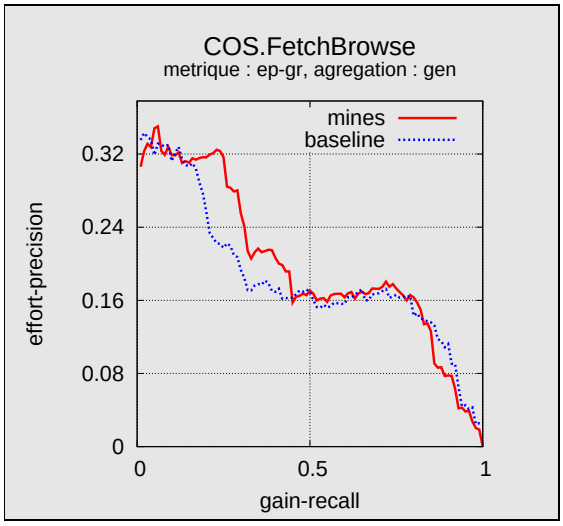


FIG. 7.17 – Courbes pour la tâche CO+S.FetchBrowse, agrégation généralisée, INEX 2005.

a. COS.Thorough			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	0.0359	0.0588	1.63
Gen	0.2557	0.2525	0.99
nxCG[25]			
Strict	0.0903	0.0329	0.36
Gen	0.2541	0.2146	0.84
nxCG[50]			
Strict	0.1207	0.0331	0.27
Gen	0.2537	0.1919	0.76
ep-gr			
Strict	0.0189	0.0124	0.66
Gen	0.0904	0.0741	0.82

b. COS.Focussed			
	Baseline	*Mines*	*Ratio*
nxCG[10]			
Strict	0.0824	0.0647	0.79
Gen	0.2027	0.2128	1.04
nxCG[25]			
Strict	0.1849	0.1023	0.55
Gen	0.2067	0.1853	0.90
nxCG[50]			
Strict	0.1662	0.0865	0.52
Gen	0.1850	0.1863	1.01
ep-gr			
Strict	0.0420	0.0311	0.74
Gen	0.0934	0.0805	0.86

c. COS.FetchBrowse			
	Baseline	*Mines*	*Ratio*
ep-gr			
Strict	0.1504	0.1092	0.73
Gen	0.1523	0.1637	1.07

TAB. 7.3 – Résultats pour la tâche CO+S, INEX 2005.

7.3 Expérimentations avec le langage XOR

Nous proposons ici nos expérimentations effectuées dans le cadre de notre langage XOR, extension de NEXI détaillée à la section 6.4 [1]. Une fois encore, c'est sur la collection INEX 2005 que nous avons réalisé ces tests.

7.3.1 Les requêtes multiples

Comme nous le décrivons dans la section 6.5.1, nous utilisons la possibilité offerte par XOR de construire des requêtes multiples (plusieurs cibles liées par des opérateurs logiques), en particulier pour permettre une application pleine et entière du principe de la recherche contextuelle. Cette section commente les résultats obtenus dans ce cadre, en les comparant avec la "baseline" et avec les requêtes en NEXI sans aucune recherche contextuelle (section 7.2). Les courbes *nxCG* et *ep/gr* correspondantes sont présentées à la section 7.2 ; celles illustrant les scores de la recherche contextuelle avec XOR sont intitulées "XOR multiples".

Les résultats pour l'ensemble des sous-tâches CAS (figures 7.2 à 7.5) sont nettement améliorés au niveau du gain par la technique de la recherche contextuelle. Pour chaque graphique, la courbe "XOR" reste constamment au-dessus de la "baseline", mais aussi au-dessus de l'interface NEXI. On peut constater qu'un peu plus de doxels pertinents sont retrouvés, mais surtout qu'ils sont mieux classés et plus pertinents.

En revanche, il est décevant de constater que pour la tâche CO (dont les résultats sont illustrés aux figures 7.6 à 7.8), les résultats obtenus avec NEXI ne sont pas améliorés.

[1] Page 162.

La conclusion de cette série d'expérimentations concernant la recherche contextuelle est que la technique peut être très efficace si l'analyse de la requête a été performante. Des mots-clés importants, bien identifiés, et des éventuelles contraintes structurelles bien reconnues, sont la base nécessaire. Dans ce cas, comme le montrent les résultats de la tâche CAS, les scores peuvent être améliorés.

Dans le cas contraire, le bruit provoqué par les mots sans rapport avec la requête est accentué, et le bénéfice de la recherche structurelle annulé.

7.3.2 Autres

Nous avons tenté de réaliser quelques expérimentations supplémentaires concernant les techniques proposées à la section 6.5.2 [1], avec les prédicats *contains*, les liens entre les documents et les limitations de la taille des éléments. Malheureusement, très peu de requêtes de la collection INEX 2005 sont concernées par ces traitements (deux pour le prédicat *contains*, une seule pour la relation *linkToAbout* et aucune pour la taille des éléments), et les résultats ne sont quasiment pas modifiés[2].

Ceci est dû au fait que, comme nous l'avons déjà remarqué, le langage officiel des campagnes INEX étant NEXI, les topics proposés sont choisis pour leur adéquation avec les possibilités de ce langage. Les requêtes pouvant difficilement s'exprimer en NEXI sont donc évitées.

Nous voulons cependant insister sur l'importance de telles techniques, qui s'éloignent certes du cadre d'INEX, voire de la recherche d'information au sens strict, mais qui concourent selon nous à la prise en compte d'une palette plus large de besoins. Cette ambition nous semble naturelle, étant données les possibilités offertes par les documents semi-structurés.

7.4 Quelques exemples d'analyse

Pour illustrer nos expérimentations et nos commentaires, nous proposons ici quatre exemples d'analyses de requêtes d'INEX, avec le résultat de l'analyse syntaxique, l'issue de l'application des règles sémantiques et enfin la requête NEXI obtenue. De plus, nous proposons une comparaison avec l'analyse de la description en Anglais effectuée par un autre analyseur syntaxique, Link Grammar [216] [3]. Cet analyseur est robuste, puisqu'il intègre la possibilité, avec les liens "nuls", de produire des analyses partielles si besoin est [98].

7.4.1 Topic 141.

(141) *Retrieve sections about threads from articles about java*

Figures 7.18 et 7.19. Ce premier exemple simple montre bien l'utilité du principe de désambiguïsation syntaxique par les règles. La construction de la phrase est basique et Link Grammar, tout comme notre système, fournit une analyse syntaxique complète et viable. Pourtant, la présence de trois groupes prépositionnels successifs multiplient les combinaisons d'attachements possibles, et les deux analyseurs produisent en premier choix des solutions erronées (figure 7.18). En revanche, comme l'indique la figure 7.19,

[1] Page 168.
[2] C'est pourquoi ils présentent trop peu d'intérêt pour que nous les présentions.
[3] http://hyper.link.cs.cmu.edu/link/.

Retrieve sections about threads from articles about java

a. Texte de la requête.

```
(S (VP Retrieve
       (NP sections)
       (PP about
           (NP threads))
       (PP from
           (NP articles))
       (PP about
           (NP java))))
```

b. Analyse syntaxique avec Link Grammar.

valeurs
- c1 (sym:Retrieve, cat:v, lem:retrieve)
- c2 (sym:sections, cat:nn, lem:section, tag:sec)
- c3 (sym:threads, cat:nn, lem:thread)
- c4 (sym:articles, cat:nn, lem:article, tag:article)
- c5 (sym:java, cat:nn, lem:java)

relations
- objet(c1, c2)
- about(c4, c5)
- **from(c3, c4)**
- about(c2, c3)

cible
- –

c. Analyse syntaxique avec l'interface.

FIG. 7.18 – Topic 141 (1/2).

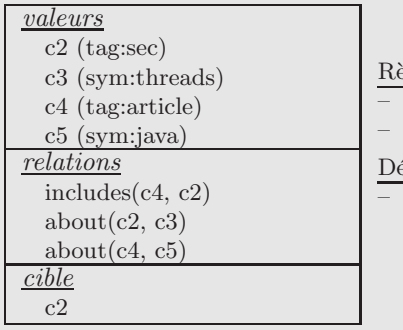
des actions sémantiques simples permettent de rétablir la forme appropriée et de parvenir à une requête NEXI satisfaisante. Notamment, la relation *from(<élément>, <élément>)* est préférée à juste titre à *from(<élément>, <texte>)*.

7.4.2 Topic 156.

(156) *We are looking for sections in articles, whose abstracts contain "spatial join", that describe "performance evaluation".*

Figures 7.20 et 7.21. On voit là que Link Grammar échoue à analyser la dernière proposition relative, et que l'interface attache de préférence le verbe *"describe"* à *"articles"*. Ainsi les deux propositions relatives sont liées à un seul élément, tandis que l'autre (*"section"*) est privé de contenu.

La figure 7.21.a détaille les modifications apportées par les règles sémantiques, ainsi que le choix de l'attachement de la seconde proposition relative à l'élément `'sec'`.



Règles utilisées :
– détection de la cible, avec *verbe_cible(retrieve, objet)*.
– lien prépositionnel, avec *relation_prep(from, included)*.

Désambiguïsation :
– choix de la représentation proposant *from(c2, c4)* plutôt que *from(c3, c4)* pour obtenir de préférence une telle relation entre deux éléments et pour éviter un élément sans contenu.

a. Application des règles sémantiques avec l'interface.

```
//article[about(., java)]//sec[about(., threads)]
```

b. Requête NEXI obtenue.

FIG. 7.19 – Topic 141 (2/2).

Par ailleurs les guillemets utilisés dans la requête ne sont pas considérés par Link Grammar, tandis que l'interface reproduit les guillemets de la description dans la requête finale, quelles que soient les catégories des mots entre guillemets.

7.4.3 Topic 162.

(162) *Algorithm for text and index compression*

Figures 7.22 et 7.23. Comme c'est bien souvent le cas, cet exemple n'est pas une phrase entière, mais un groupe nominal seul. Link Grammar ne prévoit pas ce cas et propose une analyse totalement erronée, cherchant à attribuer contre toute logique le rôle de verbe au nom *"index"*.

Si notre interface gère bien cet aspect, elle rencontre en revanche un problème avec le mot *"text"*, qu'elle reconnaît comme une référence à une balise de type `'*'`, ce qui n'est pas le cas en l'occurrence. S'ensuit une confusion au niveau de la structure et une requête NEXI séparant support et cible, alors que rien ne laisse penser que cela soit pertinent. Les dégats sont cependant limités, le mot *"text"* étant très peu discriminant et les mots-clés principaux étant bien reconnus.

7.4.4 Topic 164.

(164) *I am surveying knowledge management for a report I am writing, and am looking for knowledge management frameworks and technologies for organizational memories.*

Figures 7.24 et 7.25. Nous terminons avec un bon exemple de requête non structurée, longue et générant du bruit.

La proposition relative *"I am writing"* pose problème aux systèmes qui échouent tous deux à la reconnaître correctement. Notre interface fait ensuite une suite d'erreurs

We are looking for sections in articles, whose abstracts contain "spatial join", that describe "performance evaluation".

a. Texte de la requête.

```
(S (NP We)
   (VP are
       (VP looking
           (PP for
               (NP sections))
           (PP in
               (NP (NP articles)
                   ,
                   (SBAR (WHNP whose abstracts)
                         (S (VP contain
                                (NP (ADJP spatial join ,)
                                    (that describe performance))
                            evaluation .)))))))
```

b. Analyse syntaxique avec Link Grammar.

valeurs
- c1 (sym:We, cat:pp, lem:speaker)
- c2 (sym:are looking, cat:v, lem:look)
- c3 (sym:sections, cat:nn, lem:section, tag:sec)
- c4 (sym:articles, cat:nn, lem:article, tag:article)
- c5 (sym:abstracts, cat:nn, lem:abstract, tag:abs)
- c6 (sym:contain, cat:v, lem:contain)
- c7 (sym:'spatial join', cat:quot)
- c8 (sym:describe, cat:v, lem:describe)
- c9 (sym:'performance evaluation', cat:quot)

relations
- sujet(c2, c1)
- for(c2, c3)
- sujet(c6, c5)
- objet(c6, c7)
- of(c5, c4)
- objet(c8, c9)
- **sujet(c8, c4)**
- includes(c4, c3)

cible
- –

c. Analyse syntaxique avec l'interface.

FIG. 7.20 – Topic 156 (1/2).

importantes. Elle remplace tout d'abord *"report"* par la balise `'*'`, puis considère le verbe *"survey"*, ne correspondant à aucune règle sémantique, comme un mot-clé, ce qui est un facteur de bruit dans la requête finale.

Le verbe *"write"*, en revanche, déclenche une règle qui devrait introduire un auteur, ce qui n'est pas du tout pertinent dans ce cas. Cependant, la proposition *"I am writing"*

valeurs c3 (tag:sec) c4 (tag:article) c5 (tag:abs) c7 (sym:'spatial join', cat:quot) c9 (sym:'performance evaluation', cat:quot)
relations included(c3, c4) about(c3, c9) includes(c4, c5) about(c5, c7)
cible c3

Règles utilisées :
- détection de la cible, avec *verbe_cible(look, for).*
- lien verbal, avec *relation_verbale(contain, sujet, objet, about).*
- lien verbal, avec *relation_verbale(describe, sujet, objet, about).*
- lien prépositionnel, avec *relation_prep(in, included).*
- transformation de la relation *of(c5, c4)* pour *includes(c4, c5)* à l'aide de la DTD.

Désambiguïsation :
- attachement de *c9* à la section (*c3*) plutôt qu'à l'article (*c4*, dans la représentation précédente) pour éviter un élément sans contenu (*a fortiori* la cible).

a. Application des règles sémantiques avec l'interface.

```
//article[about(.//abs, "spatial join")]//sec[about(., "performance evaluation")]
```

b. Requête NEXI obtenue.

FIG. 7.21 – Topic 156 (2/2).

Algorithm for text and index compression

a. Texte de la requête.

```
(S algorithm for text and
   (S (VP index
          (NP compression))))
```

b. Analyse syntaxique avec Link Grammar.

valeurs c1 (sym:algorithm, cat:nn, lem:algorithm) c2 (sym:text, cat:nn, lem:text, **tag:***) c3 (sym:'index compression', cat:np, lem:'index compression')
relations for(c1, c2) coord(c4, and, [c1, c3])
cible –

c. Analyse syntaxique avec l'interface.

FIG. 7.22 – Topic 162 (1/2).

valeurs c1 (sym:algorithm) c2 (**tag:***) c3 (sym:'index compression')
relations for(c1, c4) about(c2, c3)
cible c2

Règles utilisées :
– détermination de la cible : seul élément présent.

a. Application des règles sémantiques avec l'interface.

```
//article[about(., algorithm)]//*[about(., "index compression")]
```

b. Requête NEXI obtenue.

FIG. 7.23 – Topic 162 (2/2).

n'étant rattachée à rien d'autre, cette règle ne mène à rien.

Enfin, le regroupement en groupes nominaux par des guillemets est là plutôt contre-productif en ce qui concerne *"knowledge management frameworks"*, mais est utile pour *"organizational memories"*. Ceci pose le problème des groupes nominaux composés de plus de deux termes atomiques, et dont le regroupement n'est peut-être pas judicieux. Plusieurs propositions de regroupements en bi-termes, ou l'absence totale de regroupement, pourraient être envisagés.

7.5 Conclusion

Nous avons présenté dans ce chapitre trois séries d'expérimentations. La première concerne la traduction "basique" d'une requête en langage naturel vers une requête en NEXI, de la façon décrite au chapitre 5. La seconde introduit l'idée de recherche contextuelle tout en restant dans le cadre imposé de NEXI. Enfin, la troisième utilise le langage XOR, que nous avons proposé.

Tous les résultats pour les requêtes concernant la structure et le contenu (CAS) sont très satisfaisants, ils montrent une efficacité des requêtes produites par l'interface supérieure à celles des humains. En particulier, nous constatons le bien-fondé de la recherche contextuelle.

En ce qui concerne les requêtes sur le contenu seulement (CO), le bilan est moins bon, et l'interface ne parvient pas à égaler les performances des requêtes manuelles. Ce résultat est bien entendu un peu décevant, mais peu surprenant. En effet, la plupart de nos efforts, notamment au niveau des règles sémantiques, ont porté sur la reconnaissance des références structurelles. Ce travail était destiné à étudier les apports mutuels entre l'analyse des requêtes et les documents XML. Nous avons peu de doutes sur le fait qu'il est désormais possible de prendre appui sur notre interface pour mettre en place d'autres techniques de traitement du contenu et pour améliorer encore les résultats.

Quoi qu'il en soit, parvenir à approcher dans certains cas, à dépasser dans d'autres

I am surveying knowledge management for a report I am writing, and am looking for knowledge management frameworks and technologies for organizational memories.

a. Texte de la requête.

```
(S (NP I)
   (VP am
       (VP surveying
           (NP knowledge management)))
           (PP for
               (NP (NP (NP a report)
                       (SBAR (S (NP I)
                                (VP am))))
                   (VP writing , and am looking
                       (PP   for
                             (NP (NP knowledge management frameworks)
                                 and
                                 (NP technologies)))
                       (PP   for
                             (NP organizational memories)))))))
.)
```

b. Analyse syntaxique avec Link Grammar.

valeurs
- c1 (sym:I, cat:pp, lem:speaker)
- **c2 (sym:am surveying, cat:v, lem:survey)**
- c3 (sym:'knowledge management', cat:np, lem:'knowledge management')
- c4 (sym:report, cat:nn, lem:report, **tag:***)
- c5 (sym:I, cat:pp, lem:speaker)
- **c6 (sym:am writing, cat:v, lem:write)**
- c7 (sym:am looking, cat:v, lem:look)
- c9 (sym:'knowledge management frameworks', cat:np, ...)
- c10 (sym:technologies, cat:nn, lem:technology)
- c12 (sym:'organizational memories', cat:np, ...)

relations
- sujet(c2, c1)
- objet(c2, c3)
- for(c3, c5)
- coord(c8, and, [c6, c7])
- for(c6, c11)
- for(c7, c11)
- coord(c11, and, [c9, c10])
- for(c11, c13)

cible
- –

c. Analyse syntaxique avec l'interface.

Fig. 7.24 – Topic 164 (1/2).

cas, les performances d'un processus manuel complexe avec une procédure automatique est une bonne performance. Cela indique que le but initialement recherché, à savoir la démonstration de la viabilité de l'utilisation du langage naturel pour l'expression du

valeurs
c2 (sym:am surveying)
c3 (sym:'knowledge management')
c4 (tag:*)
c6 (sym:am writing)
c9 (sym:'knowledge management frameworks')
c10 (sym:technologies)
c12 (sym:'organizational memories')
relations
for(c3, c5)
coord(c8, and, [c6, c7])
for(c6, c11)
coord(c11, and, [c9, c10])
for(c11, c12)
includes(s1, c9)
includes(s1, c10)
cible
s1

a. Application des règles sémantiques avec l'interface.

```
//article[about(., "knowledge management frameworks" technologies) AND
      about(., "organizational memories") AND about(., survey)]
```

b. Requête NEXI obtenue.

FIG. 7.25 – Topic 164 (2/2).

besoin d'information, est atteint.

Conclusion générale

Synthèse

Les travaux présentés dans ce mémoire ont visé à confronter les techniques du traitement automatique de la langue à la problématique des documents semi-structurés, et en particulier de la recherche d'information dans ces documents. Trois axes de travail principaux ont été suivis dans cette optique.

Le premier axe concerne l'analyse textuelle du contenu des documents XML. Nous avons défini et utilisé la notion de "contexte de lecture", qui permet de conserver la linéarité du texte présent dans un document, et ce malgré la structure, qui met à mal cette linéarité. Une classification automatique des balises ainsi qu'un outil d'analyse textuelle ont été développés.

Nous avons également proposé une interface permettant de formuler des requêtes en langage naturel pour effectuer une recherche d'information dans les documents semi-structurés. Cette interface, basée sur une analyse syntaxique et des règles sémantiques, a été conçue pour s'adapter sans difficulté à tous types de structure et de domaines. Le système génère une requête dans un langage formel déjà existant, ce qui le rend indépendant du moteur de recherche utilisé. Cette caractéristique est très utile pour populariser l'usage de l'interface, mais laisse apparaître un certain nombre de limites.

Pour dépasser ces limites, nous avons enfin utilisé cette interface pour affiner les requêtes formelles générées et les adapter aux spécificités de la structure, ceci afin d'améliorer la qualité des résultats retournés. En particulier, nous avons posé les bases de la "recherche contextuelle", qui tient compte de l'imbrications des éléments XML, et nous avons étudié les liens entre les documents. Pour mettre en œuvre nos idées, nous avons défini un nouveau langage d'interrogation plus conforme à nos besoins, et modifié un moteur de recherche pour qu'il se conforme à ce nouveau langage.

Des résultats expérimentaux ont été proposés pour chacune de ces étapes : les premiers, concernant la classification des balises, sont basés sur notre propre protocole, s'appuyant sur une étude statistique ; les autres, concernant les interfaces, utilisent des collections et des métriques d'évaluation reconnues dans le domaine de la recherche d'information semi-structurée.

Ces expérimentations ont confirmé la validité de nos approches. Notamment, nous avons montré, grâce au cadre d'INEX, que le fait d'utiliser, en recherche d'information semi-structurée, des besoins d'information exprimés en une langue naturelle plutôt que dans un langage formel, pouvait conduire, outre à un confort supérieur évident pour l'utilisateur, à des résultats de meilleure qualité.

Améliorations

De nombreuses améliorations peuvent et doivent être apportées aux techniques décrites dans ce mémoire. En ce qui concerne la classification automatique des balises, un outil rapide et simple à utiliser reste à concevoir. Au sujet des interfaces en langage naturel, toutes les étapes, les analyses syntaxique et sémantique comme les procédés d'amélioration de la requête, ainsi que le traitement de celle-ci par le moteur de recherche, peuvent être améliorées. En particulier, nous avons remarqué dans nos séries d'évaluation des lacunes en ce qui concerne la sélection des mots-clés les plus discriminants pour la requête.

Un autre point important est la gestion de la robustesse du système en termes de corrections des fautes de frappe, d'orthographe ou de syntaxe de l'utilisateur, qui sont très fréquentes. De nombreuses recherches ont porté sur ce sujet, notamment pour les interfaces de bases de données, et la problématique est peu différente dans notre cas.

Par ailleurs, rendre la gestion des règles sémantiques plus simple est très important. Nous avons mis en place un certain nombre de cadres génériques ou de formulaires facilitant l'entrée ou la modification des données pour certains types de règles, mais des progrès restent à accomplir avant d'envisager de mettre l'administration du système entre les mains d'un gestionnaire non expert.

Perspectives

La problématique à laquelle nous nous sommes attaqué n'en est qu'à ses débuts. Chaque étape, chaque semaine de notre travail nous a ouvert un nombre considérable de perspectives que la communauté, à défaut de votre serviteur, se doit d'explorer. Parmi ces pistes de travail, nous pouvons citer :

- les applications "classiques" du traitement automatique de la langue en recherche d'information, que nous avons délibérément délaissées pour nous consacrer aux aspects spécifiques des documents semi-structurés. Il serait pourtant intéressant d'adapter ces techniques pour observer leurs effets en présence d'une structure.

- l'interrogation des **collections hétérogènes**, c'est-à-dire dont les documents n'obéissent pas tous à la même DTD, et qui nécessitent donc des requêtes formelles et des approches différentes. Il nous semble évident que les interfaces en langage naturel, en plaçant l'utilisateur à un niveau plus conceptuel de l'expression de son besoin, peut résoudre en grande partie ce problème. Pourtant, nous n'avons réalisé aucune expérimentation à ce sujet.

- l'instauration d'un **dialogue** avec l'utilisateur, à l'image des interfaces en langage naturel pour les bases de données. Nous avons indiqué que dans ce domaine le dialogue apportait beaucoup. Même si le type de besoins, moins précis, de la recherche d'information s'y prête moins, il est probable qu'un système permettant une véritable interaction donnerait une grande satisfaction aux utilisateurs.

- l'opportunité de pousser la logique jusqu'à un point qui nous rapprocherait des systèmes de **question-réponse**. Un même système, appliqué à des documents semi-structurés, devrait être capable de gérer à la fois les demandes vagues de la recherche d'information et les requêtes précises du question-réponse. La détermination adéquate de la taille de l'élément retourné, associée à un traitement supplémentaire éventuel, ferait le reste.

– la possibilité d'utiliser les résultats d'un système de recherche d'information structurée pour effectuer des tâches de **résumé automatique**. En effet, il est possible d'identifier des passages pertinents en provenance de multiples sources. Une simple étude sur la présentation agrégée des résultats, ou un remodelage des contenus pour obtenir un "véritable" résumé, seraient des tâches utiles.

– la liaison avec d'autres disciplines de **gestion des connaissances** structurées, comme le Semantic Web. Par exemple, associer une interface en langage naturel à des documents RDF, sémantiquement bien plus élaborés que les documents XML classiques, pourrait nous conduire vers la conception de systèmes "intelligents" plus perfectionnés.

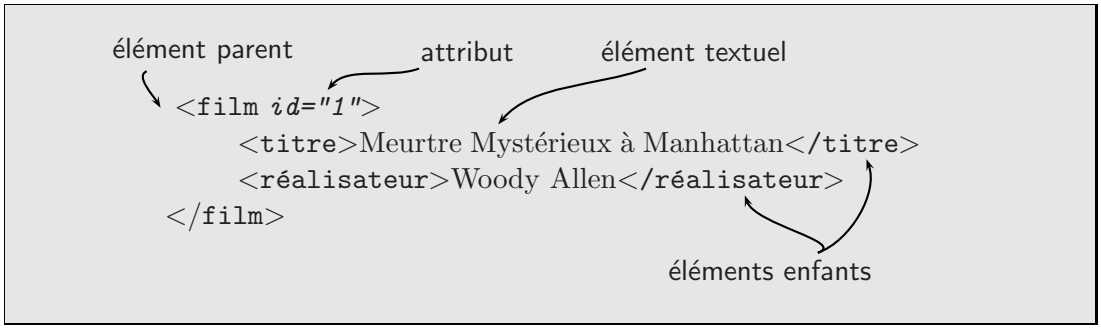
Annexes

Annexe A

Le langage XML

A.1 Les bases de XML

Un document XML peut se représenter sous la forme d'une arborescence d'éléments. Cette arborescence comporte une racine (unique), qui englobe l'ensemble des autre éléments, ainsi que des branches et des feuilles, qui forment la structure même du document. Ils peuvent contenir du texte, ou bien d'autres éléments (ses "enfants") :



Les autres notions de la terminologie XML sont données au chapitre 1.

commentaires XML

Il est possible d'ajouter des commentaires, qui commencent par `'<!--'` et se terminent par `'-->'`. Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise.

Certains caractères ayant un sens précis en XML, on a recours à des entités prédéfinies pour les remplacer lorsqu'on en a besoin. Par exemple `'&'` remplace le caractère `'&'`, `'<'` le signe `'<'`, `'>'` le signe `'>'`, etc. D'autres entités peuvent être définies par l'utilisateur. Un autre moyen d'échapper aux caractères prédéfinis est de les insérer dans des sections CDATA :

<exemple><![CDATA[Une section CDATA peut contenir des < et des >.]]>
</exemple>

Il est aussi possible de définir des règles plus strictes indiquant quelles sont les séquences et imbrications de balises autorisées. Cela se fait à l'aide d'une DTD ou d'un Schéma.

A.2 La DTD

DTD

La DTD est une description générique d'une classe de documents XML. Elle impose des contraintes concernant le nom des balises à employer et les relations que celles-ci possèdent entre elles : inclusion, ordre, possibilité d'absence, etc.

On peut dire par abus de langage que deux documents obéissant à la même DTD ont la même structure. Cependant il faut garder à l'esprit qu'ils peuvent présenter des différences sensibles dans leur structuration. Ainsi, pour un DTD représentant des articles scientifiques, comme celles de la collection INEX 2005, un document peut contenir une bibliographie ou un résumé, tandis que l'autre non ; l'un peut contenir trois sections dans le corps du texte, l'autre quatre, etc.

On dit d'un document respectant une DTD donnée qu'il est *valide* par rapport à cette DTD. Un exemple de DTD correspondant au document XML montré à la figure 1.1 (page 10).

```
<!ELEMENT cinémathèque (film*)>
<!ELEMENT film (titre, réalisateur, année?, durée?, affiche?, site?,
acteurs?)>
<!ATTLIST film id CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT réalisateur (#PCDATA)>
<!ELEMENT année (#PCDATA)>
<!ELEMENT durée (#PCDATA)>
<!ATTLIST durée unité (min|h) #REQUIRED>
<!ELEMENT affiche (#PCDATA)>
<!ELEMENT site (#PCDATA)>
<!ELEMENT acteurs (acteur+)>
<!ELEMENT acteur (#PCDATA)>
```

FIG. A.1 – Exemple de DTD

A.3 XLink

XLink

XLink [248] permet les interactions entre documents XML. Il rend possible les liens simples, mais aussi multiples (plusieurs cibles, internes, externes, etc.), ce qui le distingue des liens HTML, et les accompagne éventuellement d'annotations. Il est également bi-directionnel, c'est-à-dire qu'il garde la trace du document source lorsqu'un lien est déclenché.

Enfin, n'importe quel élément peut devenir un lien et, grâce à XPointer, on peut indexer des positions arbitraires d'un document.

Un exemple de lien étendu est donné à la figure A.2.

```
<element         xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
xlink:type="extended">
     <xlink:locator href="Source" role="role1"></xlink:locator>
     <xlink:locator href="Target" role="role2"></xlink:locator>
     <xlink:arc               from="role1" to="role2" show="embed" ac-
     tuate="auto"></xlink:arc>
     <xlink:title>The link title</xlink:title>
     <xlink:title xml:lang="fr">Description du lien</xlink:title>
</element>
```

FIG. A.2 – Exemple de lien XLink.

Annexe B

Traitement automatique de la langue

B.1 Les catégories grammaticales

B.1.1 Aperçu rapide

On distingue les catégories fermées, dont les membres sont en nombre limité, des catégories ouvertes, qui évoluent et contiennent un nombre potentiellement infini d'éléments. Dans la première classe, on trouve :

- les **prépositions** (*à*, *de*, *en*, *par*, *pour*, *sur*, etc.) ; *préposition*
- les **déterminants** (*un*, *une*) ; *déterminant*
- les **pronoms** (*elle*, *qui*, *je*) ; *pronom*
- les **conjonctions** (*et*, *mais*, *si*) ; *conjonction*
- les **verbes auxiliaires** (*être*, *avoir*) ; *auxiliaire*
- les **numéraux** (nombre infini mais connu) ;
- en Anglais, les **particules** (*up*, *down*, *out*).

Les classes ouvertes sont :

- les **noms** communs et propres ; *nom*
- les **verbes** ; *verbe*
- les **adjectifs** ; *adjectif*
- les **adverbes** (*délicatement*, *très*, *vite*). *adverbe*

B.1.2 Les étiquettes de la *Penn Treebank*

La *Penn Treebank* [152] est un corpus en Anglais d'environ 4,5 millions de mots. Ce corpus est annoté de façon semi-automatique par les catégories grammaticales des mots et comporte également des indications syntaxiques.

Le jeu d'étiquettes grammaticales utilisé pour ce projet est devenu un standard de la langue anglaise. Il comporte les 36 étiquettes que liste la table B.1, ainsi que 12 étiquettes concernant la ponctuation. C'est cet ensemble qu'utilise pour l'Anglais l'analyseur morphosyntaxique TreeTagger [210] dont nous nous sommes servi, ainsi que l'analyseur syntaxique de surface Cass [6].

Symbole	*Signification*
CC	conjonction de coordination
CD	nombre
DT	déterminant
EX	*there*
FW	mot étranger
IN	préposition or conjonction de subordination
JJ	adjectif
JJR	adjectif comparatif
JJS	adjectif superlatif
LS	marqueur de liste
MD	modal
NN	nom singulier
NNS	nom pluriel
NP	nom propre singulier
NPS	nom propre pluriel
PDT	prédéterminant
POS	marque du possessif (*'s*)
PP	pronom personnel
PP$	pronom possessif
RB	adverbe
RBR	adverbe comparatif
RBS	adverbe superlatif
RP	particule
SYM	symbol
TO	*to*
UH	interjection
VB	verbe, forme de base (infinitif)
VBD	verbe au passé
VBG	verbe au gérondif ou participe présent
VBN	participe passé
VBP	verbe au présent, sauf 3ème personne du singulier
VBZ	verbe au présent, 3ème personne du singulier
WDT	déterminant en '*wh*'
WP	pronom en '*wh*'
WP$	pronom possessif en '*wh*'
WRB	adverbe en '*wh*'

Tab. B.1 – Les étiquettes de la *Penn Treebank*.

B.2 Quelques éléments de grammaire

B.2.1 Les groupes nominaux

Les groupes nominaux sont organisés autour d'une *tête*, qui est le nom central. Les autres éléments sont des modificateurs.

Avant la tête, on trouve les déterminants (*"le"*, *"la"*, *"des"*, etc.) et les pré-déterminants (*"tout"*, *"tous"*). Les groupes adjectivaux peuvent être situés avant ou après (*"bleu"*, *"tout bleu"*, *"bleu comme une orange"*).

– le **chat**
– le premier et dernier **chat**
– tous les *chats*
– les deux **chats**
– les deux petits *chats*
– les *chats* blancs

Après le nom de tête, on trouve des groupes prépositionnels (*le chat* ***sur le mur***), les clauses non définies (gérondifs, *le chat* ***faisant une petite sieste***) ou des propositions relatives (*le chat* ***qui ronronne au soleil***).

B.2.2 Les groupes verbaux

Les verbes peuvent être de trois types principaux : les verbes auxiliaires (*"être"* et *"avoir"* en Français), les verbes modaux (*"pouvoir"*, *"vouloir"*...) et les verbes principaux, qui sont tous les autres.

On peut également distinguer les verbes selon leur transitivité. Les verbes *intransitifs* n'attendent aucun complément et peuvent faire du sens en eux-mêmes (*"Je cours"*, *"Il rit"*). Les verbes *transitifs* demandent un complément d'objet, direct pour les verbes transitifs directs (*"Il a retrouvé ses clés"*, *"Il a lu un livre"*), indirect pour les verbes transitifs indirects (*"J'obéis* à *ma mère"*, *"Je me méfie* de *lui"*). Certains verbes acceptent les deux types de compléments d'objet (*"Je donne ma bénédiction à leur union"*). Il faut remarquer que la plupart des verbes transitifs peuvent être utilisés de façon intransitive, avec d'autres types de compléments (*"Je lis dans mon lit"*, *"Je me méfie toujours"*).

intransitif

transitif

Un groupe verbal est un verbe accompagné de ces compléments, d'objet ou autre.

B.2.3 Les propositions relatives

Les propositions relatives sont des groupes verbaux ou des phrases complètes que l'on attache à des noms ou des verbes pour les complémenter. Un pronom relatif, parfois optionnel (en Anglais), fait le lien entre les deux.

Exemples :

– La thèse que j'ai écrite.
– Le directeur qui m'a encadré.
– Je me demande qui a signé ce livre.

Ainsi, les groupes verbaux peuvent contenir des groupes nominaux, et réciproquement, ce qui conduit à une récursivité théoriquement infinie.

B.2.4 La phrase

Trois types essentiels de construction au niveau de la phrase peuvent être distingués.

Les phrases déclaratives. Ce sont les plus courantes, elles sont composées d'un groupe nominal sujet suivi d'un groupe verbal.

Par exemple :

– Le petit chat miaule
– Mon avion part à 23h.
– Les pâtes que nous avons mangées à Florence n'étaient pas aussi bonnes que celles que la cousine de Loïc nous a cuisinées hier soir.

Les phrases impératives. Elles commencent par un groupe verbal, n'ont pas de sujet et sont utilisées en général pour commander ou suggérer :

- Miaule, petit chat !
- Ferme la porte.
- Passe-moi le sel.

Les questions. On peut séparer les questions fermées (réponse oui ou non, les deux premiers exemples) et les questions ouvertes (avec un pronom interrogatif, les deux exemples suivants).

- Le chat miaule-t-il ?
- La période d'essai de deux ans va-t-elle être maintenue ?
- Pourquoi le chat miaule-t-il ?
- Combien de temps la période d'essai doit-elle durer ?

B.2.5 Les abréviations utilisées

Les abréviations utilisées pour représenter les constituants principaux sont les suivantes :

- **NP**, pour Noun Phrase (syntagme nominal) ;
- **VP**, pour Verbal Phrase (syntagme verbal) ;
- **PP**, pour Prepositional Phrase (syntagme prépositionnel) ;
- **S**, pour Sentence (phrase).

B.3 L'analyse sémantique et le lambda-calcul

lambda-calcul

Le lambda-calcul (λ-calcul) constitue un formalisme capable de répondre aux problèmes de compositionnalité dans l'analyse sémantique [1]. Le λ-calcul est destiné à définir et à manipuler des fonctions [154] [2]. Les notions d'*abstraction* et d'*application* modélisent respectivement la *définition* et l'*appel* d'une fonction.

Ainsi, l'expression suivante est une abstraction :

(B.1) $\lambda x.chien(x)$

Les *variables* accolées au λ sont les paramètres d'appel. Ici on peut appliquer la *constante* MILOU à cette application et obtenir, par un mécanisme de *réduction* (nous représentons l'application par un signe @) :

(B.2) $\lambda \boldsymbol{x}.chien(\boldsymbol{x})@\text{MILOU} \equiv chien(\text{MILOU})$

Ainsi chaque catégorie grammaticale peut être représentée par une abstraction. Par exemple :

- article indéterminé (*un*) : $\lambda P \lambda Q\ \exists x(P@x \wedge Q@x)$
- déterminant universel (*tous*) : $\lambda P \lambda Q\ \forall x(P@x \rightarrow Q@x)$
- un nom propre : $\lambda P.P@\text{MEDOR}$ (et pas simplement une constante comme dans l'exemple simplifié B.2)
- verbe intransitif* (*miauler*) : $\lambda x.miauler(x)$
- verbe transitif direct* (*manger*) : $\lambda X.\lambda z.(X@(\lambda x.manger(z,x)))$

[1] Voir la section 3.4.2.1.

[2] Seul un très rapide aperçu du λ-calcul est donné ici, le lecteur intéressé pourra consulter les cours disponibles en ligne [25] ou d'autres ouvrages de référence [20].

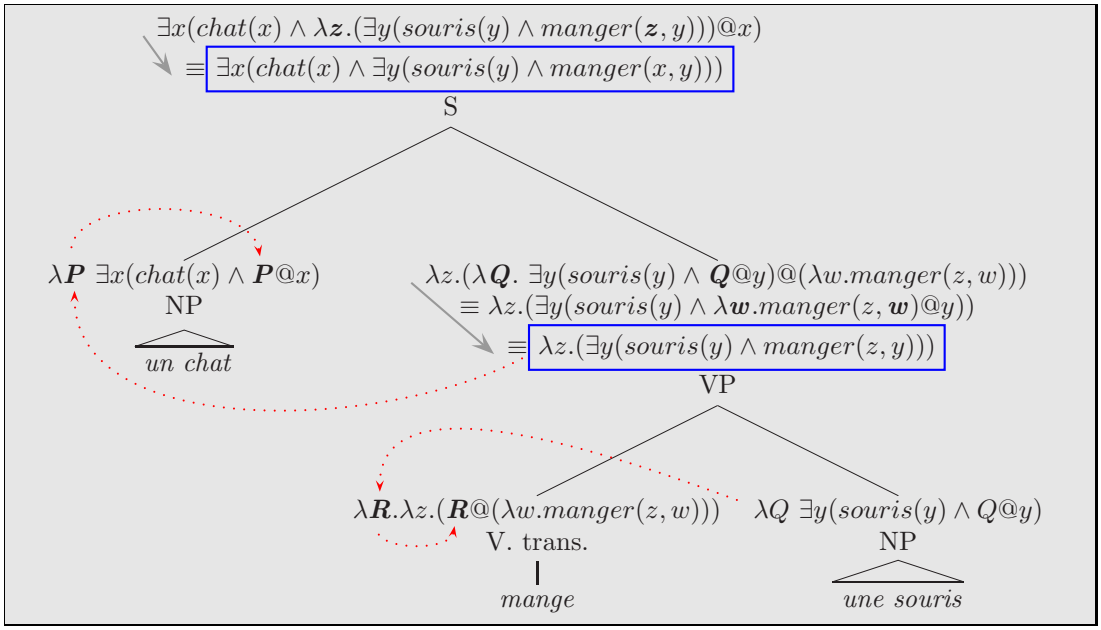


FIG. B.1 – Analyse sémantique de la phrase *"Un chat mange une souris"*.

Et chaque règle syntaxique est l'occasion d'une application :

– Pour un syntagme nominal composé d'un article indéfini et d'un nom :

$$\overbrace{\lambda \boldsymbol{P} \lambda Q\ \exists x(\boldsymbol{P}@x \wedge Q@x)}^{un}\ @\ \overbrace{\underline{\lambda y.chat(y)}}^{chat} \equiv \lambda Q\ \exists x(\lambda \boldsymbol{y}.chat(\boldsymbol{y})@x \wedge Q@x) \equiv \boxed{\lambda Q\ \exists x(chat(x) \wedge Q@x)}$$

– Pour une phrase composée d'un syntagme nominal et d'un verbe intransitif :

$$\overbrace{\lambda \boldsymbol{Q}\ \exists x(chat(x) \wedge \boldsymbol{Q}@x)}^{un\ chat}\ @\ \overbrace{\underline{\lambda y.miauler(y)}}^{miaule} \equiv \exists x(chat(x) \wedge \lambda \boldsymbol{y}.miauler(\boldsymbol{y})@x) \equiv \boxed{\exists x(chat(x) \wedge miauler(x))}$$

– On voit que l'application est la même si le syntagme nominal est un nom propre :

$$\overbrace{\lambda \boldsymbol{P}.\boldsymbol{P}@\text{TOM}}^{Tom}\ @\ \overbrace{\underline{\lambda y.miauler(y)}}^{miaule} \equiv \lambda \boldsymbol{y}.miauler(\boldsymbol{y})@\text{TOM} \equiv \boxed{miauler(\text{TOM})}$$

La figure B.1 utilise ces exemples et illustre l'analyse sémantique d'une phrase complète.

Annexe C

XGTagger

C.1 Description

XGTagger[1] est une interface générique traitant le texte contenu dans les documents XML. Il ne fonctionne pas seul, mais englobe n'importe quel système $\mathcal{S}$ d'analyse textuelle souhaité par l'utilisateur. A partir d'une liste (dépendant de la DTD des documents traités, et donc fournie par l'utilisateur) de balises transparentes d'une part, de balises de saut d'autre part, il analyse le document XML et recompose les contextes de lecture (texte pur). Il fournit le texte obtenu en entrée du système $\mathcal{S}$. Après exécution du programme, il récupère sa sortie, l'analyse et recompose le document initial, enrichi par les données ajoutées par $\mathcal{S}$ (voir figure C.1). La seule contrainte importante concernant le système $\mathcal{S}$ est qu'il doit reproduire le texte original d'une façon ou d'une autre dans sa sortie.

XGTagger évite toute perte d'information, et l'opération est totalement réversible, le document initial pouvant être récupéré grâce à une simple feuille de style de type XSLT [246]. Les paragraphes suivants proposent quelques exemples d'utilisation de ce logiciel générique : analyse morphosyntaxique, syntaxique, lexicale ou simple regroupement des contextes de lecture.

C.2 Exemples

Quelques exemples présenteront bien mieux l'utilité et le fonctionnement de ce logiciel que de longs discours. A ce jour un certain nombre d'analyseurs morphosyntaxiques ou syntaxiques sont capables de produire du XML en sortie. A notre connaissance aucun ne gère les documents XML en entrée. Deux exemples traitent de ce problème (sections C.2.1 et C.2.3). Deux autres sections proposent des applications plus simples concernant l'enrichissement lexical (comme la traduction, section C.2.2) et le simple regroupement des contextes de lecture (section C.2.4).

C.2.1 Analyse morphosyntaxique

Voici un exemple d'analyse morphosyntaxique réalisé avec XGTagger, interfacé avec l'analyseur TreeTagger [211] (système $\mathcal{S}$ = TreeTagger).

① Soit le document XML suivant :

[1] http://www.emse.fr/~tannier/fr/xgtagger.html

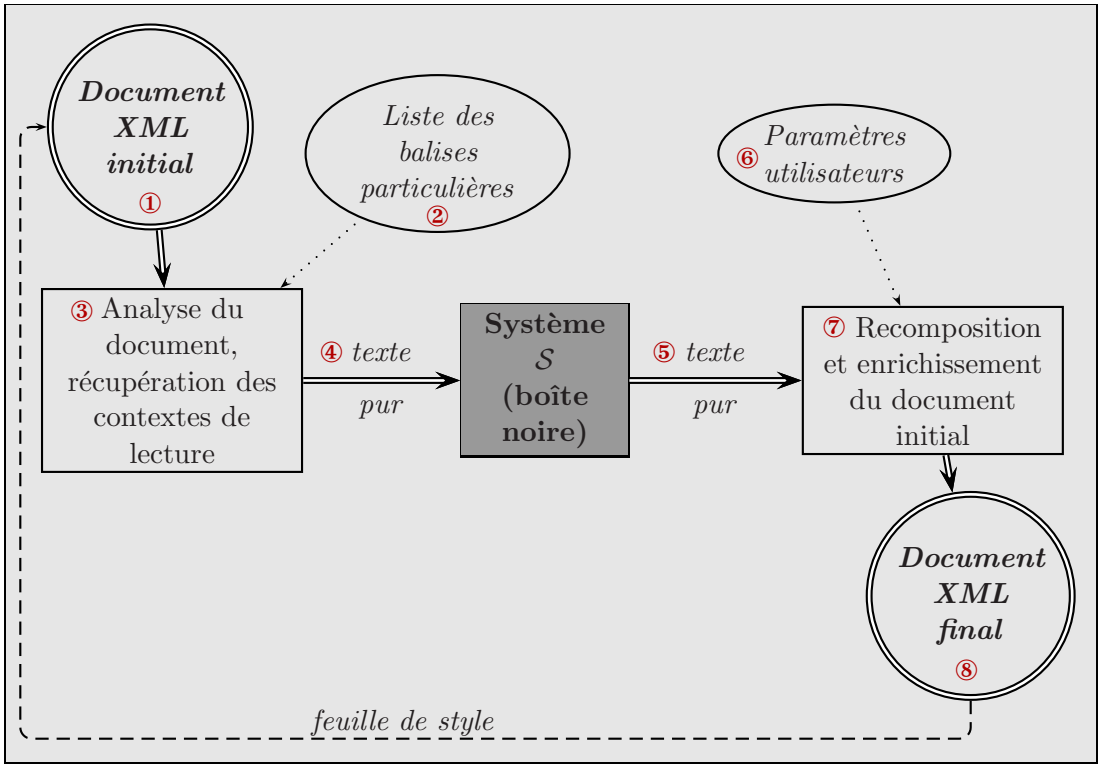


FIG. C.1 – Schéma de fonctionnement de XGTagger. Le processus est réversible, c'est-à-dire qu'il est possible de retrouver le document initial à partir du document final.

```
<article>
    <titre>Visitez I<pc>stanbul</pc></titre>
    <par>
        Cette ancienne capitale de trois empires<note>Istanbul traversa
        les empires romain, byzantin et ottoman</note> est maintenant
        en <gras>Turquie</gras>.
    </par>
</article>
```

② `'note'` est une balise de saut, `'gras'` et `'pc'` des balises transparentes[1].

③ Avec l'aide de cette information, XGTagger reconstitue les contextes de lecture et insère des points entre chacun d'entre eux, "forçant" ainsi le système à ne pas les mélanger[2]. Ces points seront retirés à la fin du processus. Le texte constitué par XGTagger et donné en entrée de TreeTagger est donc :

④
```
Visitez Istanbul . Cette ancienne capitale de trois empires est
maintenant en Turquie. . Istanbul traversa les empires romain,
byzantin et ottoman
```

⑤ Sortie de TreeTagger (présentée en deux colonnes) :

[1] Dans l'état actuel du logiciel, XGTagger n'utilise pas la technique de détermination automatique de la classe des balises, l'utilisateur doit fournir lui-même cette information dans un fichier de configuration.

[2] Il peut s'agir de n'importe quel autre caractère, selon les besoins spécifiques de l'utilisateur. Dans le cas de l'analyse morphosyntaxique, le point présente l'avantage de former une coupure forte et d'éviter toute confusion et toute ambiguïté du type de celles présentées à la section 4.1.3.

Visitez	VER:pres	visiter	Turquie	NAM	Turquie
Istanbul	NAM	Istanbul	.	SENT	.
.	SENT	.	.	SENT	.
Cette	PRO:DEM	ce	Istanbul	NAM	Istanbul
ancienne	ADJ	ancien	traversa	VER:simp	traverser
capitale	NOM	capitale	les	DET:ART	le
de	PRP	de	empires	NOM	empire
trois	NUM	trois	romain	ADJ	romain
empires	NOM	empire	,	PUN	,
est	VER:pres	être	byzantin	ADJ	byzantin
maintenant	ADV	maintenant	et	KON	et
en	PRP	en	ottoman	ADJ	ottoman

⑥ Les paramètres utilisateurs précisent le format de la sortie du système $\mathcal{S}$, notamment :
- le séparateur de mots (ici le retour chariot) ;
- le séparateur de champs (ici les espaces ou la tabulation) ;
- le champ correspondant au mot initial (ici le premier) ;
- la sémantique des champs importants (ici la catégorie – *cat* – est dans le deuxième champ et le lemme* – *lem* – dans le troisième).

⑦ La sortie du système $\mathcal{S}$ et ces paramètres sont utilisés pour recomposer le document initial et l'enrichir avec les informations apportées par l'analyse.

⑧ La sortie finale est donnée à la figure C.2. Notons que les identifiants (`'id'`) permettent de reconstituer les contextes de lecture : deux identifiants consécutifs correspondent au même contexte de lecture (voir autour de l'élément `'note'`). De plus, deux balises ayant le même identifiant représentent un même mot (par exemple l'identifiant 2).

Ainsi XGTagger peut permettre de faire le lien entre les documents XML et des formats d'annotation morphosyntaxique utilisant XML, comme MAF [47]. Par ailleurs, une option annexe du logiciel permet d'opérer des analyses portant sur des groupes de mots, comme par exemple des analyses syntaxiques.

C.2.2 Enrichissement lexical

On peut également utiliser un système $\mathcal{S}$ ajoutant des informations à certains mots du texte. Par exemple, un enrichissement à base de synonymes ou une traduction des noms dans diverses langues :

① Document XML :

```
<proverbe>De gustibus et coloribus non disputandum</proverbe>
```

⑤ Exemple de sortie possible pour $\mathcal{S}$:

```
De
gustibus/goût/Geschmack, Vorliebe
et
coloribus/couleurs/Farbe
non
disputandum/débattre, discuter/diskutieren
```

⑥ Options :
- $2^{ème}$ champ = `français` ;
- $3^{ème}$ champ = `allemand` ;

```
<article>
     <titre>Visitez I<pc>stanbul</pc></titre>
     <par>
          Cette ancienne capitale de trois empires<note>Istanbul traversa les empires ro-
          main, byzantin et ottoman</note> est maintenant en <gras>Turquie</gras>.
     </par>
</article>
```

a. Document initial

```
<article>
     <titre>
          <w id="1" cat="VER:pres" lem="visiter">Visitez</w>
          <w id="2" cat="NAM" lem="Istanbul">I</w>
          <pc>
              <w id="2" cat="NAM" lem="Istanbul">stanbul</w>
          </pc>
     </titre>
     <par>
          <w id="4" cat="PRO:DEM" lem="ce">Cette</w>
          <w id="5" cat="ADJ" lem="ancien">ancienne</w>
          <w id="6" cat="NOM" lem="capitale">capitale</w>
          <w id="7" cat="PRP" lem="de">de</w>
          <w id="8" cat="NUM" lem="trois">trois</w>
          <w id="9" cat="NOM" lem="empire">empires</w>
          <note>
              <w id="16" cat="NAM" lem="Istanbul">Istanbul</w>
              <w id="17" cat="VER:simp" lem="traverser">traversa</w>
              ...
              <w id="24" cat="ADJ" lem="ottoman">ottoman</w>
          </note>
          <w id="10" cat="VER:pres" lem="être">est</w>
          <w id="11" cat="ADV" lem="maintenant">maintenant</w>
          <w id="12" cat="PRP" lem="en">en</w>
          <gras>
              <w id="13" cat="NAM" lem="Turquie">Turquie</w>
          </gras>
          <w id="14" cat="SENT" lem=".">.</w>
     </par>
</article>
```

b. Document final

FIG. C.2 – Document XML avant (a.) et après (b.) étiquetage morphosyntaxique.

– délimiteur de champs : '/' ;
– Identifiants non demandés.

⑧ La sortie est présentée à la figure C.3.

C.2.3 Analyse syntaxique

Une option particulière de XGTagger permet d'effectuer (toujours par l'intermédiaire du système "boîte noire" $\mathcal{S}$) des analyses dépassant le cadre du mot. Nous pre-

```
<proverbe>De gustibus et coloribus non disputandum</proverbe>
```

a. Document initial

```
<proverbe>
    <w>De</w>
    <w français="goûts" allemand="Geschmack, Vorliebe">gustibus</w>
    <w>et</w>
    <w français="couleurs" allemand="Farbe">coloribus</w>
    <w>non</w>
    <w français="débattre, discuter" allemand="diskutieren">disputandum</w>
</proverbe>
```

b. Document final

FIG. C.3 – Document XML avant (a.) et après (b.) enrichessement lexical.

nons ici l'exemple de l'analyse syntaxique, en imaginant par simplicité un système $\mathcal{S}$ qui, après analyse morphosyntaxique, regroupe les groupes nominaux *GN* prépositionnels de la forme :

GN → *GN_SIMPLE (PREPOSITION GN_SIMPLE)+*

avec :

GN_SIMPLE → *((ART? NOM) | (NOM_PROPRE))*

Ainsi :

① Pour l'élément suivant :

```
<phrase>Le futur accélérateur LHC pourrait permettre l'observation du
<a href="http://fr.wikipedia.org/wiki/Boson_de_Higgs">boson de Higgs</a>
dans quelques années.</phrase>
```

② ... et en considérant que `'a'` est une balise transparente...

④ Le texte fourni en entrée de $\mathcal{S}$ sera :

```
Le futur accélérateur LHC pourrait permettre l'observation du
boson de Higgs dans quelques années.
```

⑤ Le système, appliquant les règles décrites ci-dessus, retournera par exemple :

```
Le futur accélérateur LHC pourrait permettre
l'__observation__du__boson__de__Higgs/GN dans quelques années.
```

⑥ Parmi les paramètres utilisateurs doit alors figurer la chaîne de caractère permettant de séparer les mots dans le résultat de l'analyse (ici : `'__'`). Par ailleurs le séparateur de champs est `'/'` et le second champ est la catégorie (`cat`).

⑦ Conservant le marquage (notamment la balise `'a'` coupant le syntagme) et ajoutant la nouvelle information, le système composera le document XML final.

⑧ Ce document, présenté à la figure C.4, permet grâce aux identifiants *'id'* de retrouver le groupe nominal malgré la présence de la balise `'a'`.

Cependant XGTagger n'est pas initialement conçu pour le traitement des groupes de mots. L'utilisation de cette technique reste donc limitée ; notamment il est impossible d'attribuer des valeurs supplémentaires aux mots à l'intérieur des groupes (ici, on ne

```
<phrase>Le futur accélérateur LHC pourrait permettre l'observation du
<a href="http://fr.wikipedia.org/wiki/Boson_de_Higgs">boson de Higgs</a>
dans quelques années.</phrase>
```

a. Document initial

```
<phrase>
    <w id="1">Le</w>
    <w id="2">futur</w>
    <w id="3">accélérateur</w>
    <w id="4">LHC</w>
    <w id="5">pourrait</w>
    <w id="6">permettre</w>
    <w id="7" cat="GN">l'</w>
    <w id="7">observation</w>
    <w id="7">du</w>
    <a href="http://fr.wikipedia.org/wiki/Boson_de_Higgs">
        <w id="7">boson</w>
        <w id="7">de</w>
        <w id="7">Higgs</w>
    </a>
    <w id="8">dans</w>
    <w id="9">quelques</w>
    <w id="10">années</w>
</phrase>
```

b. Document final

FIG. C.4 – Document XML avant (a.) et après (b.) analyse syntaxique.

peut pas renseigner la catégorie grammaticale des mots seuls). De plus on peut imaginer que les systèmes de base devront être modifiés pour respecter les contraintes de séparateurs spéciaux. Néanmoins il aurait été dommage de se priver d'une telle fonctionnalité qui peut être utile à maints égards.

C.2.4 Regroupement des contextes de lecture

Un dernier exemple d'application consiste en un système qui se contenterait de retourner en sortie le texte fourni en entrée (éventuellement avec une séparation de la ponctuation). Le résultat est que les mots sont insérés dans des balises et que, grâce aux identifiants des éléments, les contextes de lecture sont regroupés et les mots coupés réassemblés (voir l'exemple figure C.5). Cette opération peut servir de prétraitement avant une indexation des documents, ou avant une recherche d'information par un système prenant en compte la proximité (logique) des termes.

C.2.5 Conclusion

Les exemples donnés ci-dessus pour illustrer le fonctionnement de XGTagger proviennent tous d'applications de traitement automatique de la langue ou de recherche

```
<article>
     <titre>
          <w id="1">Visitez</w>
          <w id="2">I</w>
          <pc><w id="2">stanbul</w> </pc>
     </titre>
     <par>
          <w id="4">Cette</w>
          <w id="5">ancienne</w>
          <w id="6">capitale</w>
          <w id="7">de</w>
          <w id="8">trois</w>
          <w id="9">empires</w>
          <note>
              <w id="16">Istanbul</w>
              <w id="17">traversa</w>
              ...
              <w id="24">ottoman</w>
          </note>
          <w id="10">est</w>
          <w id="11">maintenant</w>
          <w id="12">en</w>
          <gras>
              <w id="13">Turquie</w>
          </gras>
          <w id="14">.</w>

     </par>
</article>
```

FIG. C.5 – Regroupement des contextes de lecture d'un document XML avec XGTagger.

d'information. Cependant, nous ne doutons pas des possibilités d'utilisation de cet outil dans d'autres domaines. Même en l'absence d'utilité de la notion de contexte de lecture (pour des documents orientés données par exemple), il permet d'adapter de façon très simple des systèmes recevant du texte pur au traitement du XML.

C.3 Mise en œuvre

C.3.1 Contraintes

Les principales contraintes que le système doit respecter sont :

- la préservation des contextes de lecture ;
- la réversibilité ;
- autant que possible, la généricité ;
- un appel seul au système $\mathcal{S}$ par document. En effet beaucoup de systèmes d'analyse de texte ont une phase d'initialisation assez longue, qui ne doit donc pas être relancée plusieurs fois, pour chaque élément par exemple.

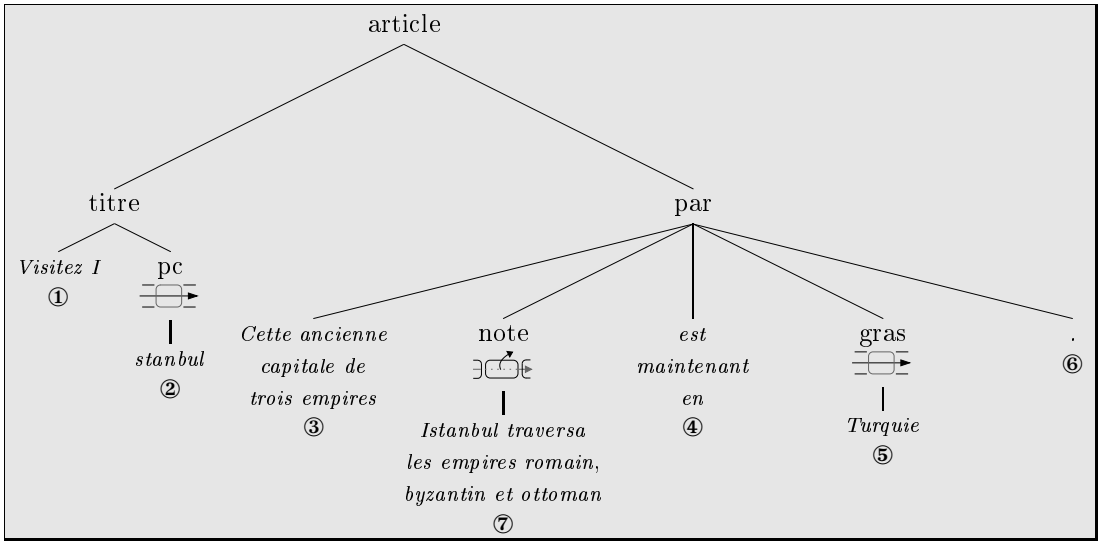


FIG. C.6 – XGTagger, ordre de parcours des balises.

```
Visitez Istanbul . Cette ancienne capitale de trois empires est
①       ②       ③                                           ④
maintenant en Turquie. . Istanbul traversa les empires romain,
              ⑤       ⑦
byzantin et ottoman
```

FIG. C.7 – XGTagger, reconstitution du texte et des contextes de lecture.

C.3.2 Fonctionnement

XGtagger parcourt le document XML à deux reprises : une première fois avant de lancer le système $\mathcal{S}$, une seconde après. Seuls les nœuds textuels (les feuilles*) subissent un traitement.

Comme l'illustre la figure C.6, ces éléments textuels sont numérotés de gauche à droite, avec une exception pour les textes contenus dans des éléments *de saut* comme la balise ⑦ de l'exemple. Ceux-ci sont repoussés à la fin du parent* de l'élément de saut concerné (ici : `'par'`).

Le texte qui doit être fourni en entrée du système $\mathcal{S}$ est reconstitué en suivant l'ordre de numérotation des balises, c'est-à-dire que le texte contenu dans les éléments de saut est repoussé (figure C.7). Les ruptures de contexte de lecture (éléments durs et éléments de saut) sont matérialisées par un caractère de séparation, ici un point, entouré d'espaces.

Finalement, les positions des caractères correspondant au début et à la fin de chaque élément textuel sont conservés (table C.1). Les espaces ne sont pas comptabilisés[1].

Le texte reconstitué est écrit dans un fichier, et le système de l'utilisateur est appelé[2]. La seule contrainte concernant ce système est qu'il doit retourner, entre autres informations, le texte initial qui lui est passé en entrée (espaces exclus ou ajoutés).

[1] Le fait de ne pas considérer les espaces permet une souplesse indispensable aux systèmes d'analyse de texte, qui peuvent ainsi couper les mots comme bon leur semble. Ainsi un analyseur morphosyntaxique comme TreeTagger va couper la chaîne de caractères *"l'a"* en *"l' "* (article) + *"a"* (verbe).

[2] Pour les détails pratiques du paramétrage de l'outil, consulter le manuel [225].

Numéro	*Caractères concernés*	*Texte correspondant*
①	1-8	Visitez I
②	9-15	stanbul
	16	*(fin contexte)*
③	17-51	Cette ancienne … trois empires
④	52-66	est maintenant en
⑤	67-73	Turquie
⑥	74-74	.
	75	*(fin contexte)*
⑦	76-124	Istanbul traversa … ottoman

Tab. C.1 – XGTagger, conservation des informations du document initial.

Sortie de $\mathcal{S}$		*Mise en correspondance*		
Texte initial	`cat`	*Caractères concernés*	*Elém. parent*	*Nouvel élément créé*
Visitez	VER:pres	1-7	①	`<w id="1" cat="VER:pres">Visitez</w>`
Istanbul	NAM	7-8	①	`<w id="2" cat="NAM">I</w>`
		9-15	②	`<w id="2" cat="NAM">stanbul</w>`
.	SENT	16	–	*fin contexte*, `id` ← `id` +1
Cette	PRO:DEM	17-21	③	`<w id="4" cat="PRO:DEM">Cette</w>`
ancienne	ADJ	22-29	③	`<w id="5" cat="ADJ">ancienne</w>`
capitale	NOM	30-37	③	`<w id="6" cat="NOM">capitale</w>`
…	…	…	…	…
…	…	…	…	…
en	PRP	65-66	④	`<w id="12" cat="PRP">en</w>`
Turquie	NAM	67-73	⑤	`<w id="13" cat="NAM">Turquie</w>`
.	SENT	74-74	⑥	`<w id="14" cat="SENT">.</w>`
.	SENT	75	–	*fin contexte*, `id` ← `id` +1
Istanbul	NAM	76-83	⑦	`<w id="16" cat="NAM">Istanbul</w>`
…	…	…	…	…

Tab. C.2 – XGTagger, mise en correspondance de la sortie du système avec les informations concernant le document initial. Le lien entre les numéros des caractères concernés et l'élément qui les contient est fait grâce à la table C.1.

Le document XML initial est ensuite parcouru une seconde fois, dans le même ordre que la première, et la sortie de $\mathcal{S}$ est consultée en parallèle. La table C.2 montre la dernière phase, celle de la mise en correspondance entre le document initial d'une part et la sortie du système d'autre part. De nouveaux éléments sont ainsi créés et insérés à la structure, pour former un nouveau document XML (voir la sortie finale à la figure C.2 page 216).

Annexe D

Interfaces en langage naturel

Nous donnons ici un certain nombre de requêtes sur lesquelles nous avons travaillé dans le cadre d'INEX. La description en Anglais est celle proposée par l'auteur de la requête, tandis que l'expression NEXI est celle générée par notre interface de base, sans aucune modification concernant la recherche contextuelle.

La distinction entre "contenu seulement" et "contenu et structure" est celle suggérée par INEX.

D.1 Contenu seulement

- Find articles about LDAP and single sign on procedures.
  //article[(about(., "single sign" LDAP) and about(., procedures))]
- Look for an article, published in either the year 2000 or the year 2001, that deals with quantum computation and is not about quantum mechanics.
  //article[((about(.//yr, 2000) and (about(., "quantum computation") and about(.//yr, 2001))) AND NOT about(., "quantum mechanics"))]
- Use of natural language processing techniques and/or artificial intelligence techniques as a Intelligent Information Retrieval approach to discover information and knowledge from Medical Informatics Literature
  //article[((about(., "Intelligent Information Retrieval" "artificial intelligence") and about(., "natural language processing")) and (about(., "Medical Informatics Literature" knowledge) and about(., discover)))]//*[about(., knowledge)]
- I am interested in articles discussing machine translation approaches but not programming languages translation approaches.
  //article[(about(., "machine translation") AND NOT about(., "programming languages translation"))]
- I want to know what personal privacy concerns have been raised by the surveillance of the Central Intelligence Agency and he Federal Bureau of Investigation. I cam particularly interested in project Carnivore.
  //article[((about(., surveillance) and (about(., "Central Intelligence Agency") and about(., "Federal Bureau of Investigation"))) and about(., cam))]//*[about(.//*, Carnivore project)]
- I want information about conferences and workshops in the multimedia field. Relevant items might be in the articles titled "call for papers" or "upcoming events".
  //article[(about(., "multimedia field" workshops) and about(.//article//atl, "call for papers" "upcoming events"))]//*[about(., conferences workshops)]
- I want to find documents describing the development of synthesizers for music

creation, including documents describing the history of music synthesizers
//article[(about(., "music synthesizers" history) and (about(., "music creation" synthesizers) and about(., development)))]
- Find references that are about Dempster-Shafer Application, where the application is modeled by Dempster-Shafer theory, and it is a Database related problem (not AI, IR), and the solution is shown with expressive experiment results.
//article[(about(., "expressive experiment results" solution) AND NOT (about(.//*, AI) AND NOT about(.//*, IR)))]//*[(about(., "Dempster-Shafer Application") and (about(., modeled) and (about(., application) and (about(., theory) and about(., Database Dempster-Shafer)))))]
- I want to know about applications of differential geometry in brain research. I'm looking for relevant information within the content of the document.
//article[about(., "differential geometry" brain)]//*[about(.//*, "differential geometry")]
- I am looking for information in the body of the article about latent semantic analysis or about latent semantic indexing
//article//*[about(., "latent semantic analysis" "latent semantic indexing" LSA LSI)]
- support and deployment of IPv6
//article//*[(about(., deployment support) and about(., IPv6))]
- Seek information about two well-known ensemble learning methods : Adaboost and Bagging.
//article[about(., Adaboost Bagging)]//*[about(., ensemble learning)]
- I would like to know about the ATM technology used in the wireless network with applications multimedia.
//article//*[(about(., technology) and (about(., "applications multimedia" "wireless network") and about(., ATM)))]
- I want to know concepts and some technical information about eletronic commerce. But is relevant articles that describes e-commerce as a business strategy.
//article//*[(about(., "business strategy" e-commerce) and about(., "eletronic commerce" concepts))]
- Return the names of commercial speech recognition software packages, along with the capabilities and limitations of each.
//article//*[(about(., "commercial speech recognition software packages") and about(., capabilities limitations))]
- Find documents or document components, most probably sections, that discuss the granularity of learning objects.
//article//*[(about(., learning objects) and about(., granularity))]
- Retrieve information about composing music with computer assistance. MIDI is a known method used in composing and note writing.
//article[(about(., "note writing" composing) and about(., MIDI))]//*[(about(., composing music) and about(., "computer assistance"))]
- Find articles and sections on user-centered design for web sites
//article//(article|sec)[about(., "user-centered design" "web sites")]
- I would like to find a document component containing textual description or a figure explaining the architecture of a multimedia retrieval system.
//article//*[about(.//*, "multimedia retrieval system" architecture)]
- I'm looking for text about the Gibbs sampler.
//article//*[about(., Gibbs sampler)]
- Find text containing equations for a hidden Markov model (HMM).
//article//*[(about(., equations) and (about(., HMM) and (about(., hidden) and about(., Markov model))))]

- I'm searching for article sections or abstracts that deal with multimedia document models or approaches for multimedia content authoring.
  //article//(sec|abs)[about(., "multimedia content authoring" "multimedia document models")]
- Return all the articles related to how to mine frequent pattern (e.g. graph, itemset, sequence). Note that freqent itemset is usually referred as association rule mining.
  //article[(about(., "association rule mining" "freqent itemset") and (about(., "mine frequent pattern") and (about(., itemset sequence) and about(., graph))))]
- Find an article dealing with history of Artificial Intelligence.
  //article[about(., "Artificial Intelligence" history)]
- I am interested to know more about DOM and SAX.
  //article//*[about(., DOM SAX)]
- What are the problems and physical limits that are encountered by the microprocessor miniaturization process ?
  //article//*[about(., "microprocessor miniaturization process" "physical limits")]
- Find information about the relevance of Marshall McLuhan's ideas for current digital technologies.
  //article//*[(about(., ideas) and about(., "Marshall McLuhan" "digital technologies"))]
- Find documents or document components, most probably sections, that describe the approach of code signing and verification.
  //article//*[about(., "code signing" verification)]
- Case studies in the use of ontologies
  //article//*[about(., "Case studies" ontologies)]

D.2 Contenu et structure

- We are seeking for information on how modern operating systems implement threads.
  //article//*[(about(., implement threads) and about(., "operating systems"))]
- sections about frequent itemsets from articles with abstract about data mining
  //article[about(.//abs, "data mining")]//sec[about(., "frequent itemsets")]
- sections of articles in which an introduction to information retrieval is given
  //article//sec[(about(., "information retrieval") and about(., introduction))]
- applications of information retrieval systems in digital libraries.
  //article//*[about(., "digital libraries" "information retrieval systems")]
- We are interested in articles from statistical machine learning community which contain sections about mutual information criterion.
  //article[(about(.//sec, "mutual information criterion") and (about(., learning "statistical machine")))]
- I would like to find sections in an article describing the application of Hidden Markov Models (HMMs) to hand gesture recognition. The major topic of the article should be hand gesture recognition and it must contain detailed description of the usage of HMMs in the retrieved component.
  //article[(about(.//*//*, HMMs) and (about(., "hand gesture recognition") and about(.//*, "hand gesture recognition")))]//sec[(about(., HMMs) and (about(., "gesture recognition" hand) and about(., "Hidden Markov Models")))]
- Explanation of the state space explosion problem in model checking
  //article//*[(about(., "model checking" "state space explosion") and about(., "state space explosion"))]
- Retrieve sections discussing free public licenses as well as open source code and

the impacts on using them. Linux envionment and gnu public license are related strongly to free licenses.
//article[(about(., "Linux envionment" "gnu public license") and about(., "free licenses"))]//sec[(about(., "free public licenses") and about(., impacts))]

- We are looking for paragraphs describing watermarking in articles which describe data embedding.
//article[about(., "data embedding")]//p[about(., watermarking)]
- Retrieve information about evaluation methodology or evaluation measures used for usability testing (experiments) in digital libraries.
//article[(about(., "usability testing" experiments) and about(., "digital libraries" experiments))]//*[about(., "evaluation measures" "evaluation methodology")]
- Find bibliography entries about database access methods for spatial and text data from articles related to that topic.
//article//bb[(about(., "database access" spatial) and about(., "text data"))]
- We are looking for sections in articles where "image retrieval" is talked about, that describe "latent semantic indexing".
//article[about(., "latent semantic indexing")]//sec[about(., "image retrieval")]
- In articles about bayesian networks find sections that are about learning the structure of the network.
//article[(about(., network structure) and about(., "bayesian networks"))]//sec[about(., learning)]
- I am looking for articles where the main theme is the Turing test or "imitation game" where machines imitate human intelligence and consciousness.
//article[(about(., imitate machines) and (about(., "human intelligence" consciousness) and about(.//*, "imitation game")))]
- We are looking for sections in articles whose abstracts contain "spatial join", that describe "performance evaluation".
//article[about(.//abs, "spatial join")]//sec[about(., "performance evaluation")]
- articles citing research by Abiteboul.
//article[about(.//bib//bb//au, Abiteboul)]
- Vita and details about authors that are phD students.
//article//*[about(.//au, "phD students")]
- Find paragraphs about word processing applications running in Linux operating system.
//article//p[(about(., "processing applications") and (about(., running) and about(., "operating system" Linux)))]
- In articles discussing web searching find sections explaining the vector space model.
//article[about(., web)]//sec[about(., "vector space model")]
- Find sections with section titles about 'inverse kinematics' in articles about animation.
//article[about(., animation)]//sec[about(.//st, "inverse kinematics")]
- Find sections about simulated annealing in articles that mention genetic algorithms
//article[about(., "genetic algorithms")]//sec[about(., "simulated annealing")]
- We are looking for paragraphs in articles about information retrieval dealing with relevance feedback.
//article[about(., "information retrieval")]//p[about(., "relevance feedback")]
- Give me some pre-existing examples of stemmed words out of sections of papers
//article//sec[about(., pre-existing stemmed)]
- we want abstracts which explicitly mention database access in perl
//article[about(., perl)]//abs[about(., "database access")]

- Retrieve sections about threads from articles about java
  //article[about(., java)]//sec[about(., threads)]
- The topic is to find XML articles with a section on information integration or information exchange over the web.
  //article[(about(.//sec, web) and about(., XML))]
- Find bibliografic references that are about text categorisation where Support Vector Machines (SVM) categoriser is used.
  //article//bb[(about(., "text categorisation") and (about(., categoriser) and about(., "Support Vector Machines" SVM)))]
- We are interested in finding the sections about experiment design and experiment results of different classification methods. We hope that the abstract should mention the classification methods.
  //article[about(.//abs, classification)]//sec[(about(., "experiment design" "experiment results") and about(., classification))]
- We are looking for the abstracts of the documents about data mining and written by Jiawei Han.
  //article//abs[(about(., "Jiawei Han") and about(., "data mining"))]
- We are searching paragraphs dealing with version management in articles containing a paragraph about object databases.
  //article[about(.//p, "object databases")]//p[about(., "version management")]
- Find paragraphs or figure-captions containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications
  //article//(p|fgc)[(about(., definition) AND (about(., Godel) OR (about(., Lukasiewicz) OR about(., "fuzzy-logic implications")))))]

Glossaire

XML

ancêtre : Un élément ou nœud XML est ancêtre d'un autre s'il contient celui-ci, quel que soit le nombre d'éléments ou nœuds situés entre eux.

contexte de lecture : Le rétablissement des contextes de lecture dans un document XML consiste à retrouver la linéarité, le "fil" du texte contenu dans ce document, en faisant abstraction des coupures engendrées par certains types de balises. Voir le chapitre 4.

descendant : Un élément ou nœud XML est descendant d'un autre s'il est contenu par celui-ci, quel que soit le nombre d'éléments ou nœuds situés entre eux.

document (vue XML orientée-) : approche des documents XML qui considère le document comme un texte traditionnel comportant un marquage structurel. Cette vue s'oppose à la vue orientée données*. Voir la section 1.2.

document plat : Document ne comportant aucune marque formelle (balisage) indiquant la structure ou la présentation du texte. Voir la section 1.2

données (vue XML orientée-) : Approche des documents XML qui considère le document comme une structure contenant des données, ayant une fonction comparable à celle d'une base de données. Cette vue s'oppose à la vue orientée document*. Voir la section 1.2.

doxel : Elément XML.

enfant : Un élément ou nœud XML est l'enfant d'un autre s'il est contenu par celui-ci de façon directe, sans élément ou nœud situé entre eux.

feuille : Dans un document XML, un nœud (ou élément) *feuille* est un nœud qui se situe au bout de l'arbre, qui ne possède pas d'enfant non textuel.

granularité : La notion de granularité définit la taille du plus petit élément, au-delà duquel l'information n'est plus découpée par du balisage.

nœud : Element XML, dans la représentation arborescente.

parent : Un élément ou nœud XML est l'enfant d'un autre s'il contient celui-ci de façon directe, sans élément ou nœud situé entre eux.

racine : Dans un document XML, un nœud (ou élément) *racine* est le nœud ancêtre de tous les autres, celui qui contient l'ensemble du document.

Recherche d'information

cible : Dans une requête structurée, la cible est la partie de la requête qui spécifie l'élément qui doit être retourné à l'utilisateur (par opposition au support*). Voir la section 2.4.2.2.

exhaustivité : En recherche d'information semi-structurée, l'exhaustivité décrit à quel point l'élément traite du sujet de la requête (voir section 2.5.4.3).

langage naturel (requête) : Se dit d'une requête exprimée par une phrase complète ou un groupe de mots en langage naturel, par opposition à une requête dans un langage formel.

lemmatisation : Opération consistant à extraire la forme canonique d'un mot (son *lemme*), ainsi qu'éventuellement d'autres informations morphologiques. Exemple : *"voitures"* est au féminin pluriel, et a pour lemme *voiture*.

poids : Importance d'un mot dans un énoncé ou dans un document. Voir la section 2.3.1.

précision : taux de documents pertinents parmi tous les documents retrouvés par le système (voir page 36).

rappel : taux de documents pertinents retrouvés par le système parmi l'ensemble des documents pertinents de la collection (voir page 36).

spécificité : En recherche d'information semi-structurée, la spécificité décrit à quel point l'élément se focalise sur le sujet de la requête (et pas sur d'autres sujets – voir section 2.5.4.3).

support : Dans une requête structurée, le support est la partie de la requête qui concerne les éléments qui ne doivent pas être retournés à l'utilisateur, ceux qui ne servent que d'aide au système pour satisfaire le besoin exprimé (par opposition à la cible*). Voir la section 2.4.2.2.

Linguistique et linguistique informatique

Certaines des définitions ci-dessous sont partiellement ou totalement issues du *Dictionnaire de linguistique et des sciences du langage*, de Jean Dubois [67], ou du *Dictionnaire des sciences du langage* de Franck Neveu [171].

anaphore : Relation référentielle qui s'exerce à l'intérieur du discours entre deux expressions linguistiques, dont l'une reçoit son interprétation du sens référentiel de l'autre. Par exemple, par l'anaphore *pronominale*, un pronom est utilisé pour désigner un référent introduit au préalable. Voir la section 3.4.3.1 page 68.

association droite : Règle de réduction des ambiguïtés syntaxiques selon laquelle les attachements de constituants doivent se faire avec le constituant le plus proche (le plus à droite). Voir la section 5.2.4, page 122.

attachement minimal : Règle de réduction des ambiguïtés syntaxiques selon laquelle les constructions arborescentes possédant le moins de nœuds doivent être préférées. Voir la section 5.2.4, page 122.

composition : Procédé de création lexicale réalisée au moyen de la juxtaposition de plusieurs morphèmes* libres (*"malveillant"*, *"portemanteau"*, *"mâchefer"*...).

dérivation : Procédé qui consiste à former de nouveaux mots en modifiant le morphème par rapport à la base (par ajout d'un préfixe ou d'un suffixe.

ellipse : Suppression d'un constituant attendu dans le discours mais dont l'absence ne fait pas obstacle à l'interprétation de l'énoncé. Voir la section 3.4.3.2 page 70.

étiquetage morphosyntaxique : reconnaissance des mots et de leur catégorie grammaticale dans un texte.

flexion : Modification que subissent les mots qui se déclinent, se conjuguent, prennent la marque du pluriel, etc.

hyperonymie : On appelle *hyponomie* un lexème* *subordonné* à un autre lexème, qui lui est par conséquent *superordonné*, et qui est appelé *hyperonyme*. Par exemple, *"basset"*, *"pomme"*, *"tilleul"* sont des hyponymes de *"chien"*, *"fruit"*, *"arbre"*.

hyponymie : voir hyperonymie.

intransitif : un verbe intransitif est un verbe qui n'admet pas de complément d'objet.

langage naturel : Se dit du langage "normal" parlé par un être humain, quelle que soit sa langue. S'oppose aux langages de programmation formels en informatique.

lemme (1) (ou lexie) : unité autonome constituante du lexique d'une langue.

lemme (2) : racine d'un mot, dépouillée des marques d'accord et de conjugaison. C'est la forme graphique conventionnellement choisie comme adresse dans un lexique.

lexique : le lexique d'une langue constitue l'ensemble de ses lemmes* ou, d'une manière plus courante mais moins précise, "l'ensemble de ses mots".

morphème : unité minimale de signification (racine des mots).

morphologie (1) : étude de la forme des mots, à travers les phénomènes ressortissant de la flexion*, de la dérivation* et de la composition*.

morphologie (2) : étude conjointe des règles de structure interne des mots et de leurs règles de combinaison (morphosyntaxe).

polysémie : Propriété d'un mot d'avoir plusieurs significations distinctes.

pragmatique : étude de la signification des énoncés en lien avec le contexte (interlocuteurs, phrases précédentes, connaissance commune du monde, ...).

règle hors contexte : règle de grammaire de la forme G → D, dans laquelle G est un unique élément et D une séquence d'éléments, qui peut être remplacée par G, quel que soit leur contexte d'apparition (voir la section 3.3.3.2).

sémantique : étude de la signification des énoncés, indépendamment de tout contexte.

synonymie : Propriété de plusieurs mots d'avoir une même signification ou des significations approchées.

syntagme : Constituant syntaxique, suite de morphèmes* organisé autour d'une tête* et exerçant dans la phrase la même fonction syntaxique que celle-ci. Un syntagme nominal a pour tête un nom, un syntagme verbal, un verbe, etc.

syntaxe : partie de la grammaire décrivant les règles par lesquelles se combinent en phrases les unités significatives (mots).

tête : la tête est le constituant lexical principal du syntagme, dont la fonction et la distribution sont identiques à celles de l'ensemble du groupe.

trait : caractérique particulière d'un mot ou d'un constituant linguistique. Exemple : le genre, le nombre d'un nom, le temps d'un verbe.

transitif : un verbe transitif implique la présence d'un syntagme nominal* pour le compléter (complément d'objet). Par exemple, le verbe *aimer (quelqu'un)* est transitif.

Bibliographie

[1] A. Abeillé. *Les nouvelles syntaxes : grammaires d'unification et analyse du Français*. Collection Linguistique. Armand Colin, 1993. 60

[2] S. Abiteboul. Querying Semi-Structured Data. In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 1–18, Delphi, Greece, 1997. 12, 14, 28

[3] S. Abiteboul, I. Manolescu, B. Nguyen, and N. Preda. A Test Platform for the INEX Heterogeneous Track. In Fuhr et al. [81], pages 358–371. 101

[4] S. Abney. Part-of-Speech Tagging and Partial Parsing. In *Corpus-Based Methods in Language and Speech*. Kluwer Academic Publisher, 1996. 55

[5] S. Abney. Partial Parsing via Finite-State Cascades. *Journal of Natural Language Engineering*, 2(4):337–344, 1996. 55, 102

[6] S. Abney. *The SCOL Manual*, Apr. 1997. http://www.vinartus.net/spa/. 207

[7] S. Aït-Mokhtar, V. Lux, and E. Banik. Linguistic Parsing of Lists in Structured Documents. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*, Budapest, Hungary, Apr. 2003. 15, 96

[8] A. Albano, D. Colazzo, G. Ghelli, P. Manghi, and C. Sartiani. A Type System for Querying XML Documents. In Baeza-Yates et al. [16]. 28

[9] J. Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc, 1994. 48, 71, 74, 116, 117

[10] I. Androutsopoulos, G. Ritchie, and P. Thanisch. An Efficient and Portable Natural Language Query Interface for Relational Databases. In P. Chung, G. Lovegrove, and M. Ali, editors, *Proceedings of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, pages 327–330, Edimburgh, UK, June 1993. Gordon and Breach Publishers Inc., Langhorne, PA, USA. 80

[11] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural Language Interfaces to Databases – An Introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995. 78, 81, 83

[12] V. N. Anh and A. Moffat. Compression and an IR Approach to XML Retrieval. In Fuhr et al. [78], pages 99–104. 24, 33

[13] A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel. Linguistically-motivated Information Retrieval. In A. Kent, editor, *Encyclopedia of Library and Information Science*, volume 69, pages 201–222. Marcel Dekker, Inc., New York, Basel, Dec. 2000. 75, 76, 120

[14] A. T. Arampatzis, T. Tsoris, C. H. A. Koster, and T. P. van der Weide. Phrase-based Information Retrieval. *Information Processing & Management*, 34(6):693–707, 1998. 122

[15] R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors. *ACM SIGIR 2002 Workshop On XML and Information Retrieval*, Tampere, Finland, Aug. 2002. ACM Press, New York City, NY, USA. 33, 235

[16] R. Baeza-Yates, N. Fuhr, R. Sacks-Davis, and R. Wilkinson, editors. *ACM SIGIR 2000 Workshop On XML and Information Retrieval*, Athens, Greece, July 2000. ACM Press, New York City, NY, USA. 28, 233, 237, 239, 241

[17] R. Baeza-Yates and G. Navarro. Integrating contents and structure in text retrieval. *ACM SIGMOD Record*, 25(1):67–79, Mar. 1996. 21

[18] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York City, NY, USA and Addison Wesley Longman Publishing Co. Inc., 1999. 18, 20, 24, 27, 28, 37

[19] M. Ballabriga. Sémantique du slogan publicitaire. In J.-M. Adam and M. Bonhomme, editors, *Analyse du discours publicitaire*, pages 95–112. Editions Universitaires du Sud, Toulouse, 2000. 72

[20] H. P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, second edition, 1997. 210

[21] M. Bates and R. J. Bobrow. Information retrieval using a transportable natural language interface. In *SIGIR '83: Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–86, Bethesda, Maryland, USA, 1983. ACM Press, New York City, NY, USA. 80

[22] R. K. Belew. *Finding Out About, A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, 2000. 18, 36

[23] R. K. Belew and J. Hatton. RAVE Reviews: Acquiring Relevance Assessment from Multiple Users. In *Machine Learning in Information Access. AAAI Spring Symposia*, 1996. 18

[24] N. J. Belkin, R. N. Oddy, and H. M. Brooks. ASK for Information Retrieval: Part I. Background and Theory. *Journal of Documentation*, 38(2):61–71, 1982. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.299-304). 17

[25] C. Berline. Cours de lambda-calcul (DEA), 2002. http://www.pps.jussieu.fr/~berline/Cours.html. 210

[26] P. Blache. Une introduction à HSPG, 1995. http://www.lpl.univ-aix.fr/ blache/publis.html. 60

[27] P. Blackburn and J. Bos. *Representation and Inference for Natural Language ; A first Course in Computational Semantics*. ComSem, 1999. 59, 63, 66

[28] A. Bonifati and S. Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, 29(1):68–79, 2000. 21, 29

[29] A. Bonnet. Les grammaires sémantiques, outil puissant pour interroger les bases de données en langage naturel. *R.A.I.R.O. Informatique*, 14(2):137–148, 1980. 67

[30] D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 14th conference on Computational linguistics*, pages 977–981, Nantes, France, 1992. Association for Computational Linguistics, Morristown, NJ, USA. 55

[31] D. Bourigault, C. Fabre, C. Frérot, M.-P. Jacques, and S. Ozdowska. Syntex, analyseur syntaxique de corpus. In *Actes des 12èmes journées sur le Traitement Automatique des Langues Naturelles (TALN'05)*, Dourdan, France, 2005. 118

[32] D. Braga, A. Campi, E. Damiani, P. Lanzi, and G. Pasi. FXPath: Flexible Querying of XML Documents. In *Proceedings of the EUROFUSE Workshop on Information Systems*, 2002. 33

[33] E. Brill. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP-92)*, pages 152–155, Trento, Italy, 1992. Association for Computational Linguistics, Morristown, NJ, USA. 75, 88

[34] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 505–516, Montreal, Quebec, Canada, 1996. ACM Press, New York City, NY, USA. 28

[35] R. Capindale and R. Crawford. Using a Natural Language Interface with Casual Users. *International Journal of Man-Machine Studies*, 32:341–361, 1990. 83

[36] D. Carmel, N. Efraty, G. M. Landau, Y. S. Meerek, and Y. Mass. An extension of the Vector Space Model for querying XML documents via XML fragments. In Baeza-Yates et al. [15], pages 151–158. 29

[37] R. Carré, J.-F. Dégremont, M. Gross, J.-M. Pierrel, and G. Sabah. *Langage Humain et Machine*. Presse du CNRS, 1991. 77

[38] F. Carton. *Introduction à la phonétique du français*. Paris, Bordas, 1974. 48

[39] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A Graphical Language for Querying and Reshaping XML Documents. In QL'98 [183]. 28

[40] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: a Graphical Language for Querying and Restructuring XML Documents. In *Proceedings of the 8th International WWW Conference, WWW8*, Toronto, Canada, May 1999. International World Wide Web Conference Committee (IW3C2). 29

[41] F. Chahuneau. XML. In *Documents Numériques – Gestion de Contenu*. Techniques de l'Ingénieur, 2001. (H7-148). 12

[42] D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. In *Proceedings of WebDB 2000 Conference*, Lecture Notes in Computer Science. Springer-Verlag, New York City, NY, USA, 2000. 28

[43] Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical Report Basic Research Action FERMI 8134, CLIPS-IMAG, 1996. 20

[44] T. T. Chinenyanga and N. Kushmerick. An expressive and efficient language for XML information retrieval. *Journal of American Society for Information Science and Technology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):438–453, 2002. 24

[45] C. L. A. Clarke, G. V. Cormarck, and F. J. Burkowski. An Algebra for Structured Text Search and A Framework for its Implementation. *The Computer Journal*, 38(1):43–56, 1994. 25, 28

[46] V. Claveau. *Acquisition automatique de lexiques sémantiques pour la recherche d'information*. PhD thesis, Université de Rennes 1, Dec. 2003. 76

[47] L. Clément and E. V. de la Clergerie. MAF: a morphosyntactic annotation framework. In *Proceedings of 2nd Language and Technology Conference (LT'05)*, pages 90–94, Poznan, Poland, Apr. 2004. 215

[48] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, and A. Heuer, editors, *Proceedings of 29th International Conference on Very Large Data Bases*, Berlin, Germany, Sept. 2003. Morgan Kaufmann. 24, 25

[49] D. Colazzo, C. Sartiani, A. Albano, G. Ghelli, P. Manghi, L. Lini, and M. Paoli. A Typed Text Retrieval Query Language for XML Documents. *Journal of American Society for Information Science and Technology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):467–488, Apr. 2002. 13, 101

[50] M. Consens and G. Navarro, editors. *String Processing and Information Retrieval: 12th International Conference, SPIRE 2005*, volume 3772 of *Lecture Notes in Computer Science*, Buenos Aires, Argentina, Nov. 2005. Springer-Verlag, New York City, NY, USA. 236, 239

[51] J. H. Coombs, A. H. Renear, and S. J. DeRose. Markup Systems and the Future of Scholarly Text Processing. *Communications of the ACM*, 30(11):933–947, 1987. 11, 107

[52] W. Cooper. A Definition of Relevance for Information Retrieval. *Information Storage and Retrieval*, 7(1):19–37, 1971. 36

[53] A. Copestake and D. Flickinger. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece, 2000. 55

[54] A. Copestake and K. S. Jones. Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, 5(4):225–249, 1990. 81, 82, 83

[55] *Actes de la 1ère Conférence en Recherche d'Information et Applications, CORIA'04*, Toulouse, France, Mar. 2004. IRIT, Toulouse. 243, 244

[56] H. Cui, J.-R. Wen, and T.-S. Chua. Hierarchical Indexing and Flexible Element Retrieval for Structured Document. In F. Sebastiani, editor, *Advances in Information Retrieval, Proceedings of the 25th European Conference on IR Research, ECIR 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 73–87, Pisa, Italy, Apr. 2003. Springer-Verlag, New York City, NY, USA. 24, 25, 33

[57] H. Cunningham. Information Extraction, Automatic. In *Encyclopedia of Language and Linguistics*. Elsevier, second edition, 2005. 20

[58] S. Davis, editor. *Pragmatics, a Reader*. Oxford University Press, 1991. 48, 74

[59] R. S. de Madariaga, J. R. F. del Castillo, and J. R. Hilera. A Generalization of the Method for Evaluation of Stemming Algorithms Based on Error Counting. In Consens and Navarro [50], pages 228–233. 75

[60] S. M. Dekleva. Is Natural Language Querying Practical ? *SIGMIS Database*, 25(2):24–36, 1994. 81

[61] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006. 90, 160

[62] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A Query Language for XML. In QL'98 [183]. 28, 29

[63] A. Dister. Réflexions sur l'homographie et la désambiguïsation des formes les plus fréquentes. In *Actes des Journées Internationales d'Analyse Statistique des Données Textuelles (JADT 2000)*, 2000. 50

[64] P. Dopichaj. The University of Kaiserslautern at INEX 2005. In Fuhr et al. [80], pages 196–210. 32

[65] S. Douglas, M. Hurst, and D. Quinn. Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 535–546, Las Vegas, Nevada, USA, 1995. 15

[66] H. Dreyfus. *What Computers Can't Do*. Harper Row, New York City, NY, USA, 1979. 61

[67] J. Dubois, editor. *Dictionnaire de linguistique et des sciences du langage*. Trésors du Français. Larousse, 1994. 230

[68] C. Fabre and C. Jacquemin. Boosting Variant Recognition with Light Semantics. In *Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000*, pages 264–270, Saarbrücken, Aug. 2000. 76

[69] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, Massachussets, USA, 1998. 76

[70] XML Query Languages: Experiences and Exemplars, 1999. http://www.w3.org/1999/09/ql/docs/xquery.html. 29

[71] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World-Wide Web: a survey. *ACM SIGMOD Record*, 27(3):59–74, 1998. 28

[72] I. O. for Standardisation (ISO). Information Technology – Database Language SQL. Standard No. ISO/IEC 9075:1999. 18, 77

[73] C. Fox. A stop list for general text. *ACM SIGIR Forum*, 24(1-2):19–21, 1989. 23

[74] W. Francis. A tagged corpus – problems and prospects. In S. Greenbaum, G. Leech, and J. Svartvik, editors, *Studies in English linguistics for Randolph Quirk*, pages 192–209. Longman, 1979. 49

[75] C. Froissart. *Robustesse des interfaces homme-machine en langue naturelle*. PhD thesis, Université Pierre Mendès-France, Grenoble II, Dec. 1992. 114

[76] N. Fuhr and K. Großjohann. XIRQL – An Extension of XQL for Information Retrieval. In Baeza-Yates et al. [16]. 28

[77] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In W. Croft, D. Harper, D. Kraft, and J. Zobel, editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New York City, NY, USA, Sept. 2001. ACM Press, New York City, NY, USA. 12, 33

[78] N. Fuhr, N. Gövert, G. Kazai, and M.Lalmas, editors. *The First Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, Schloss Dagstuhl, International Conference And Research Center For Computer Science, Germany, Dec. 2002. 33, 233, 238, 239, 247

[79] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the second Workshop of the Initiative for the Evaluation of XML retrieval (INEX), December 15–17, 2003*, Schloss Dagstuhl, Germany, 2004. 33, 238, 240, 241, 242, 243, 245

[80] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*, volume 3493 of *Lecture Notes in Computer Science*, Schloss Dagstuhl, Germany, November 28-30, 2005, 2006. Springer-Verlag, New York City, NY, USA. 33, 236, 238, 239, 240, 241, 243, 246, 247

[81] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlàvik, editors. *Advances in XML Information Retrieval. Third Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, Schloss Dagstuhl, Germany, December 6-8, 2004, 2005. Springer-Verlag, New York City, NY, USA. 33, 233, 238, 241, 246, 247

[82] N. Fuhr, S. Malik, and M. Lalmas. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2003. In Fuhr et al. [79], pages 1–11. 43

[83] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structured answers for a large structured document collection. In Korfhage et al. [132], pages 204–213. 21, 32

[84] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, Nov. 1987. 76

[85] C. Gardent and K. Baschung. *Techniques d'analyse et de génération pour la langue naturelle*. Adosa, 1994. 57

[86] E. Gaussier, C. Jacquemin, and P. Zweigenbaum. Traitement automatique des langues et recherche d'information. In E. Gaussier and M.-H. Stéphanini, editors, *Assistance intelligente à la recherche d'informations*. Hermes Sciences Publications, Lavoisier, Paris, 2003. 20, 75

[87] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In Fuhr et al. [81], pages 211–223. 33, 34

[88] S. Geva. GPX - Gardens Point XML IR at INEX 2005. In Fuhr et al. [80], pages 240–253. 33, 34

[89] S. Geva and M. Leo-Spork. XPath Inverted File for Information Retrieval. In Fuhr et al. [79], pages 110–117. 33

[90] S. Geva and T. Sahama. The NLP task at INEX 2004. *ACM SIGIR Forum*, 39(1):50–53, 2005. 87

[91] C. F. Goldfarb. Introduction to Generalized Markup. In *Proceedings of the ACM SIGPLAN-SIGOA Symposium on Text Manipulation*, pages 68–73, New York City, NY, USA, 1981. ACM Press, New York City, NY, USA. 13

[92] C. F. Goldfarb. *The SGML Handbook*. Oxford University Press, 1991. 13

[93] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the Lore data model and query language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, USA, June 1999. 28

[94] J. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarran. Indexing with WordNet synsets can improve Text Retrieval. In *Proceedings of the COLING/ACL'98 Workshop on Usage of WordNet for NLP*, pages 38–44, Montreal, Canada, 1998. 76

[95] T. Grabs and H.-J. Schek. ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In Fuhr et al. [78]. 24, 32

[96] L. Gravano. Characterizing Web Resources for Improved Search. In *Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, Dec. 2000. 18

[97] H. P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*, volume 3, pages 41–58. Academic Press, New York City, NY, USA, 1975. 115

[98] D. Grinberg, J. Lafferty, and D. Sleator. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague, Czechoslovakia, 1995. 188

[99] T. Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120, Madison, Wisconsin, USA, June 2002. ACM Press, New York City, NY, USA. 26

[100] M. Géry. *Indexation et interrogation de chemins de lecture en contexte pour la Recherche d'Information Structurée sur le Web*. PhD thesis, Université Joseph Fourier, Grenoble, France, Oct. 2002. 2, 18, 160

[101] N. Gövert and G. Kazai. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002. In Fuhr et al. [78], pages 1–17. 43

[102] E. R. Harold and W. S. Means. *XML in a Nutshell*. O'Reilly, second edition, 2002. 9

[103] Y. Hayashi, J. Tomita, and G. Kikui. Searching text-rich XML documents with Relevance ranking. In Baeza-Yates et al. [16]. 26

[104] M. Hearst. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64, Mar. 1997. 20

[105] G. G. Hendrix and J. G. Carbonell. A tutorial on natural-language processing. In *Proceedings of the ACM '81 conference*, pages 4–8. ACM Press, New York City, NY, USA, 1981. 77

[106] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a natural language interface to complex data. *ACM Transaction on Database Systems*, 3(2):105–147, 1978. 71, 79

[107] D. Hiemstra and V. Mihajlović. The simplest evaluation measures for XML information retrieval that could possibly work. In Trotman et al. [229], pages 43–46. 45

[108] E. W. Hinrichs. Tense, Quantifiers, and Contexts. *Computational Linguistics*, 14(2):3–14, 1988. 81

[109] J. R. Hobbs. Resolving Pronoun References. *Lingua*, 44:311–338, 1978. 68, 70

[110] G. Hubert. XML retrieval based on direct contribution of query component. In Fuhr et al. [80], pages 172–186. 33

[111] J. R. Hurford. *Semantics : a coursebook*. Cambridge University Press, 1983. 48

[112] N. Ide and J. Veronis, editors. *The Text Encoding Initiative: Background and Context*. Kluwer Academic Publisher, 1995. 10, 13

[113] A. Imafouo and X. Tannier. Retrieval Status Values in Information Retrieval Evaluation. In Consens and Navarro [50], pages 222–227. 36

[114] T. Janssen. Compositionality. In van Benthem and ter Meulen [232], chapter 7. 64

[115] J.-H. Jayez. *Compréhension automatique du langage naturel – Le cas du goupe nominal en Français*. Masson, Paris, France, 1982. 73

[116] O. Jespersen. *A Modern English Grammar on Historical Principles, part VII: Syntax*. Allen and Unwin, London, 1954. 68

[117] K. S. Jones. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28(1):11–20, 1972. 24

[118] K. S. Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, 1997. 24

[119] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2000. 50, 62, 63

[120] Y. Kagolovsky and J. Moehr. Current Status of the Evaluation in Information Retrieval. *Journal of Medical Systems*, 27(5):409–424, Oct. 2003. 35

[121] H. Kamp and U. Reyle. *From discourse to logic*. Kluwer Academic Publisher, 1993. 62

[122] A. H. Kao. Montague Grammar. Technical Report EECS 595, University of Michigan, 2004. 64

[123] F. Katamba. *An introduction to phonology*. Longman, 1989. 48

[124] M. Kay. Functional Unification Grammar: a formalism for machine translation. In *Proceedings of the 22nd annual meeting on Association for Computational Linguistics*, pages 75–78, Stanford, California, USA, 1984. Association for Computational Linguistics, Morristown, NJ, USA. 60

[125] G. Kazai. Report of the INEX 2003 Metrics working group. In Fuhr et al. [79], pages 184–190. 45

[126] G. Kazai and M. Lalmas. INEX 2005 Evaluation Measures. In Fuhr et al. [80], pages 16–29. http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf. 44

[127] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79, Sheffield, United Kingdom, 2004. ACM Press, New York City, NY, USA. 38, 45

[128] G. Kazai, M. Lalmas, and B. Piwowarski. INEX'03 Relevance Assessment Guide. In Fuhr et al. [79], pages 203–209. 43

[129] G. Kazai, M. Lalmas, and T. Roelleke. Focussed Structured Document Retrieval. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, Lisbon, Portugal, Sept. 2002. 24

[130] J. Kekäläinen and K. Järvelin. Evaluating Information Retrieval Systems under the Challenges of Interaction and Multi-Dimensional Dynamic Relevance. In H. Bruce, R. Fidel, P. Ingwersen, and P. Vakkari, editors, *Proceedings of the CoLIS 4 Conference*, pages 253–270, Seattle, Washington, USA, July 2002. 21

[131] A. Kent, M. Berry, F. Luehrs, and J. Perry. Machine literature searching: VIII. Operational criteria for designing information retrieval systems. *American Documentation*, 6(2):83–101, 1955. 36

[132] R. Korfhage, E. M. Rasmussen, and P. Willett, editors. *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, Pennsylvania, USA, June 1993. ACM Press, New York City, NY, USA. 238, 244

[133] R. R. Korfhage. *Information Storage and Retrieval*. John Wiley & Sons, Inc., 1997. 17, 18, 27, 37

[134] E. Kotsakis. Structured information retrieval in XML documents. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 663–667, Madrid, Spain, 2002. ACM Press, New York City, NY, USA. 24

[135] G. Lallich-Boidin and D. Maret. *Recherche d'information et traitement de la langue – Fondements linguistiques et applications*. Les Cahiers de l'enssib. Presses de l'enssib, 2005. 75

[136] M. Lalmas. Dempster-Shafer's theory of evidence applied to structured documents: modelling uncertainty. In *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 110–118, Philadelphia, Pennsylvania, USA, Aug. 1997. ACM Press, New York City, NY, USA. 20, 33

[137] M. Lalmas and B. Piwowarski. INEX 2005 Relevance Assessment Guide, 2005. http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/Relevance_Assessment2005.pdf. 43, 44

[138] L. Lamport. *Latex*. Addison-Wesley, 1994. 14

[139] B. Larsen, S. Malik, and A. Tombros. The Interactive Track at INEX 2005. In Fuhr et al. [80], pages 398–410. 21, 23

[140] Y. K. Lee, S.-J. Yoo, K. Yoon, and P. B. Berra. Index structures for structured documents. In *Proceedings of the first ACM international conference on Digital libraries*, pages 91–99, Bethesda, Maryland, USA, 1996. ACM Press, New York City, NY, USA. 26

[141] D. B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38, Nov. 1995. 75

[142] S. C. Levinson. *Pragmatics*. Cambridge University Press, 1983. 48, 74

[143] L. Lini, D. Lombardini, M. Paoli, D. Colazzo, and C. Sartiani. XTReSy: A Text Retrieval System for XML documents. In D. Buzzetti, H. Short, and G. Pancalddella, editors, *Augmenting Comprehension: Digital Tools for the History of Ideas*. Office for Humanities Communication Publications, King's College, London, 2001. 13, 14, 98

[144] S. Liu, Q. Zou, and W. W. Chu. Configurable indexing and ranking for XML information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, Sheffield, United Kingdom, 2004. ACM Press, New York City, NY, USA. 24

[145] J. Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1):22–31, 1968. 75

[146] H. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):307–317, 1957. 24

[147] R. Luk, A. Chan, T. Dillon, and H. Leong. A survey of Search Engines for XML Documents. In Baeza-Yates et al. [16]. 28

[148] R. W. Luk, H. Leong, T. S. Dillon, A. T. Chan, W. B. Croft, and J. Allan. A survey in indexing and searching XML documents. *Journal of American Society for Information Science and Technology (JASIST)*, 53(6):415–437, 2002. 25

[149] S. Malik, G. Kazai, M. Lalmas, and N. Fuhr. Overview of INEX 2005. In Fuhr et al. [80], pages 1–15. 38

[150] S. Malik, M. Lalmas, and N. Fuhr. Overview of INEX 2004. In Fuhr et al. [81], pages 1–15. 43

[151] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. The MIT Press, Cambridge, Massachussets, USA, 1999. 50

[152] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics (Special Issue on Using Large Corpora)*, 19(2):313–330, June 1993. 49, 207

[153] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7(3):216–244, July 1960. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.39-46). 27

[154] D. Marre. Programmation fonctionnelle. INSA Toulouse, 2000. Cours de Génie Industriel et Informatique. 210

[155] A. Martinet. Qu'est-ce que la morphologie ? *Cahiers Ferdinand de Saussure*, 26:85–90, 1969. 49

[156] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML Components. In Fuhr et al. [79], pages 53–58. 32

[157] P. H. Matthews. *Morphology*. Cambridge University Press, 1991. 48

[158] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):54–66, Sept. 1997. 28

[159] C. T. Meadow, B. R. Boyce, and D. H. Kraft. *Text Information Retrieval Systems*. Academic Press, New York City, NY, USA, San Diego, second edition, 2000. 18, 34

[160] G. Mecca, P. Merialdo, and P. Atzeni. Do we really need a new query language for XML ? In QL'98 [183]. 28

[161] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, Nov. 1995. 73, 76

[162] J. Minel. Le résumé automatique de textes : solutions et perspectives. *Traitement automatique des langues*, 45(1), 2004. 20

[163] S. Mizzaro. Relevance, the whole (hi)story. *Journal of American Society for Information Science*, 48(9):810–832, 1997. 36

[164] S. Mizzaro. How many relevances in information retrieval ? *Interacting with Computers*, 10(3):303–320, 1998. 36

[165] A. Moffat, R. Sacks-Davis, and R. Wilkinson. Retrieval of Partial Documents. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC)*, pages 181–190, Gaithersburg, MD, 1994. National Institute of Standards and Technology, Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 500-215. 20

[166] F. Moreau and V. Claveau. Extension de requêtes par relations morphologiques acquises automatiquement. In *Actes de la 3ème conférence en Recherche d'Information et Applications (CORIA'06)*, pages 181–204, Lyon, France, Mar. 2006. 75

[167] F. Moreau and P. Sébillot. Contributions des techniques du traitement automatique des langues à la recherche d'information. Technical Report 1690, IRISA, France, Feb. 2005. 25, 75, 76, 120

[168] F. Namer. Flemm : Un analyseur Flexionnel du Français à base de règles. *Traitement automatique des langues pour la recherche d'information, numéro spécial de la revue T.A.L.*, 41(2):523–548, 2000. 75

[169] F. Namer. Acquisition automatique de sens à partir d'opérations morphologiques en Français : études de cas. In *9ème Conférence annuelle de Traitement Automatique des Langues Naturelles*, pages 235–244, Nancy, France, June 2002. 51

[170] G. Navarro. A Language for Queries on Structure and Contents of Textual Databases. Master's thesis, Dept. of Computer Science, University of Chile, Apr. 1995. 28

[171] F. Neveu. *Dictionnaire des sciences du langage*. Armand Colin, 2004. 49, 230

[172] R. A. O'Keefe and A. Trotman. The Simplest Query Language That Could Possibly Work. In Fuhr et al. [79], pages 167–174. 31, 83

[173] B. Partee. Montague Grammar. In van Benthem and ter Meulen [232], chapter 1. 64

[174] C. Peters. What happened in CLEF 2005. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria, Sept. 2005. 38

[175] J.-M. Pierrel, editor. *Ingénierie des langues*. Hermes Sciences Publications, Lavoisier, Paris, 2000. 50

[176] W. Piez. Beyond the "descriptive vs. procedural" distinction. *Markup Languages: Theory and Practice*, 3(2):141–172, Dec. 2001. 13

[177] B. Piwowarski. *Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information*. PhD thesis, Université Pierre et Marie Curie, Paris, France, July 2003. 18, 20

[178] B. Piwowarski and P. Gallinari. Expected Ratio of Relevant Units: A Measure for Structured Information Retrieval. In Fuhr et al. [79], pages 158–166. 20

[179] B. Piwowarski and M. Lalmas. Interface pour l'évaluation de systèmes de recherche sur des documents XML. In CORIA 2004 [55], pages 109–120. 43

[180] E. Popovici, G. Ménier, and P.-F. Marteau. SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In Fuhr et al. [80], pages 321–335. 33

[181] M. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980. 75

[182] R. T. Pédauque. Document : forme, signe et médium, les re-formulations du numérique. Working Paper, July 2003. http://archivesic.ccsd.cnrs.fr/documents/archives0/00/00/05/11/index_fr.html. 20

[183] *Proceedings of the Query Languages Workshop (QL'98)*, 1998. 28, 235, 236, 242, 243

[184] A. M. Rees and D. G. Schulz. A field experimental approach to the study of relevance assessments in relation to document searching. Technical Report NSF Contract No. C-423, Center for Documentation and Communication Research, School of Library Science, Case Western Reserve University, Cleveland, Ohio, USA, 1967. 36

[185] A. Renear. The descriptive/procedural distinction is flawed. *Markup Languages: Theory and Practice*, 2(4):411–420, 2000. 13

[186] A. Renear, D. Dubin, C. M. Sperberg-McQueen, and C. Huitfeldt. Towards a Semantics for XML Markup. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 119–126, McLean, Virginia, USA, 2002. ACM Press, New York City, NY, USA. 13

[187] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.281-286). 27

[188] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In QL'98 [183]. 28

[189] C. Roussey. *Une méthode d'indexation sémantique adaptée aux corpus multilingues*. PhD thesis, Institut National des Sciences Appliquées (INSA) de Lyon, Dec. 2001. 23

[190] C. Roussey, S. Calabretto, and J.-M. Pinon. Etat de l'art en indexation et recherche d'information. *Document Numérique, numéro spécial Gestion des documents et gestion des connaissances*, 3(3-4):121–150, Dec. 1999. 23

[191] P. Sabatier. *Contribution au développement d'interfaces en langage naturel*. PhD thesis, Université Paris VII, July 1987. 77

[192] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database systems for structured documents. In *Proceedings of the International Symposium on Advanced Database Technologies and Their Integration (ADTI)*, Nara, Japan, 1994. 21

[193] P. Saint-Dizier and P. Muller. Fondements linguistiques et informatiques pour le traitement automatique de la langue naturelle écrite, 2004. Cours de Master 2. 52

[194] A. Salminen and F. W. Tompa. PAT expressions: an algebra for text search. In F. Kiefer, G. Kiss, and J. Pajzs, editors, *Proceedings of the 2nd International Conference on Computational Lexicography, COMPLEX '92*, pages 309–332. Budapest: Linguistics Institute, Hungarian Academy of Sciences, 1992. 25

[195] G. Salton. *The SMART retrieval system ; experiments in automatic document processing*. Prentice-Hall, 1971. 27

[196] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In Korfhage et al. [132], pages 49–58. 20

[197] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.323-328). 24

[198] G. Salton, A. A. Fow, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022–1036, Dec. 1983. 27

[199] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983. 18

[200] M. Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 142–151, Dublin, Ireland, July 1994. Springer-Verlag, New York City, NY, USA. 76

[201] T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of American Society for Information Science*, 26(6):321–434, 1975. 36

[202] T. Saracevic. Evaluation of Evaluation in Information Retrieval. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–145, Seattle, Washington, USA, July 1995. ACM Press, New York City, NY, USA. 36

[203] K. Sauvagnat. XFIRM : Un Modèle Flexible de Recherche d'Information pour le stockage et l'interrogation de documents XML. In CORIA 2004 [55], pages 121–142. 29

[204] K. Sauvagnat. *Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés*. PhD thesis, Université Paul Sabatier, Toulouse, France, June 2005. 18, 21

[205] K. Sauvagnat and M. Boughanem. A la recherche de nœuds informatifs dans des corpus de documents XML: où pourquoi on a toujours besoin de plus petit que soi... In *Actes de la 2ème conférence en Recherche d'Information et Applications (CORIA'05)*, pages 119–134, Grenoble, France, Mar. 2005. 21

[206] J. Savoy. Stemming of French words base on grammatical category. *Journal of American Society for Information Science*, 44(1):1–9, 1993. 75

[207] J. Savoy. A stemming procedure and stopword list for general French corpora. *Journal of American Society for Information Science*, 50(10):944–952, 1999. 23, 75

[208] J. Savoy. Modèles en recherche d'information. In E. Gaussier and M.-H. Stéphanini, editors, *Assistance intelligente à la recherche d'informations*. Hermes Sciences Publications, Lavoisier, Paris, 2003. 75

[209] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. *Journal of American Society for Information Science and Tecnology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):489–503, Apr. 2002. 24, 25, 26, 29, 32, 33

[210] H. Schmid. TreeTagger - a language independent part-of-speech tagger. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html. 207

[211] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, Sept. 1994. 75, 89, 96, 102, 213

[212] D. Shin, H. Jang, and H. Jin. BUS: an effective indexing and retrieval scheme in structured documents. In *Proceedings of the third ACM conference on Digital libraries*, pages 235–243, Pittsburgh, Pennsylvania, USA, 1998. ACM Press, New York City, NY, USA. 26

[213] C. L. Sidner. Focusing for interpretation of pronouns. *Computational Linguistics*, 7(4):217–231, 1981. 70

[214] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-based Approach to XML Retrieval. In Fuhr et al. [79], pages 19–26. 25, 33, 157

[215] B. Sigurbjörnsson, A. Trotman, S. Geva, M. Lalmas, B. Larsen, and S. Malik. INEX 2005 Guidelines for Topic Development, 2005. http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/TD05.pdf. 39, 42

[216] D. Sleator and D. Temperley. Parsing English with a Link Grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, 1993. 55, 58, 188

[217] A. F. Smeaton. Using NLP or NLP Resources for Information Retrieval Tasks. In Strzalkowski [221], pages 99–111. 76

[218] I. Song and P. S. Bayerl. Semantics of XML documents. Internal Working Paper, Apr. 2003. 13

[219] K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, Computer Laboratory, University of Cambridge, 1975. British Library Research and Development Report 5266. 36

[220] C. M. Sperberg-Mcqueen, C. Huitfeldt, and A. Renear. Meaning and interpretation of markup. *Markup Languages: Theory and Practice*, 2(3):215–234, Aug. 2000. 13

[221] T. Strzalkowski, editor. *Natural Language Information Retrieval*. Kluwer Academic Publisher, Dordrecht, NL, 1999. 245, 247

[222] T. Strzalkowski, F. Lin, J. Wang, and J. Perz-Carballo. Evaluating Natural Language Processing Techniques in Information Retrieval. In Strzalkowski [221], pages 113–145. 76

[223] D. R. Swanson. Historical Note: Information Retrieval and the Future of an Illusion. *Journal of the American Society for Information Science*, 39(2):92–98, 1988. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.555-561). 36

[224] J. Swets. Effectiveness of information retrieval methods. *American Documentation*, 20(1):72–89, 1969. 36

[225] X. Tannier. XGTagger User Manual. http://www.emse.fr/~tannier/XGTagger/Manual/, June 2005. 220

[226] X. Tannier, J.-J. Girardot, and M. Mathieu. Classifying XML Tags through "Reading Contexts". In P. R. King, editor, *Proceedings of the 2005 ACM Symposium on Document Engineering*, pages 143–145, Bristol, United Kingdom, Nov. 2005. ACM Press, New York City, NY, USA. 12

[227] A. Theobald and G. Weikum. The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking. In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT)*, pages 477–495, Prague, Czech Republic, Mar. 2002. 26, 33

[228] B. Trippe. Do XML Editors Matter ?, Oct. 2001. http://www.transformmag.com/db_area/archs/2001/10/tfm0110xm.shtml. 11

[229] A. Trotman, M. Lalmas, and N. Fuhr, editors. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, UK, Aug. 2005. 239, 246

[230] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In Fuhr et al. [81], pages 16–40. 29, 31

[231] E. Tzoukermann, J. L. Klavans, and C. Jacquemin. Effective use of natural language processing techniques for automatic conflation of multi-word terms: the role of derivational morphology, part of speech tagging, and shallow parsing. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 148–155, Philadelphia, PA, USA, 1997. ACM Press, New York City, NY, USA. 75, 122

[232] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. The MIT Press, Cambridge, Massachussets, USA, 1997. 239, 242

[233] R. van Zwol, J. Baas, H. van Oostendorp, and F. Wiering. Query Formulation for XML Retrieval with Bricks. In Trotman et al. [229], pages 80–88. 84

[234] R. van Zwol, G. Kazai, and M. Lalmas. The Multimedia Track at INEX 2005: Overview. In Fuhr et al. [80], pages 497–510. 23

[235] R. van Zwol, F. Wiering, and V. Dignum. The Utrecht Blend: Basic Ingredients for an XML Retrieval System. In Fuhr et al. [81], pages 140–152. 101

[236] E. Voorhees. Using WordNet to Disambiguate Word Senses for Text Retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 171–180, Pittsburgh, Pennsylvania, USA, 1993. ACM Press, New York City, NY, USA. 76

[237] E. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, Dublin, Ireland, July 1994. Springer-Verlag, New York City, NY, USA. 76

[238] E. Voorhees. Using WordNet for Text Retrieval. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, Massachussets, USA, 1998. 76

[239] E. Voorhees. Overview of TREC 2004. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the Thirteenth Text REtrieval Conference (TREC)*, Nov. 2004. NIST Special Publication 500-261. 36, 38

[240] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Proceedings of the Sixth Text REtrieval Conference (TREC)*, Nov. 1997. NIST Special Publication 500-240. 36

[241] E. Voorhees and D. M. Tice. The TREC-8 question answering track evaluation. In *Proceedings of the Eighth Text REtrieval Conference (TREC)*, Nov. 1999. NIST Special Publication 500-246. 18, 20

[242] V. Vutukuru, K. Pasupuleti, A. Khare, and A. Garg. Conceptemy: An issue in XML information retrieval. In *Posters of the eleventh International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, 2002. 131

[243] J. Véronis. Morphosyntactic correction in natural language interfaces. In *Proceedings of the 12th conference on Computational Linguistics*, pages 708–713, Budapest, Hungry, 1998. Association for Computational Linguistics, Morristown, NJ, USA. 114

[244] World Wide Web Consortium (W3C). http://www.w3.org/. 9, 13, 29

[245] XML Path Language (XPath). World Wide Web Consortium (W3C) Recommandation, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116. 10, 29, 84

[246] XSL Transformation (XSL). World Wide Web Consortium (W3C) Recommandation, 1999. http://www.w3.org/TR/1999/REC-xslt-19991116. 10, 29, 213

[247] Semantic Web. World Wide Web Consortium (W3C)., 2001. http://www.w3.org/2001/sw/. 13, 93

[248] XML Linking Language (XLink), 2001. http://www.w3.org/TR/xlink/. 204

[249] XML Schema (W3C), 2001. http://www.w3.org/XML/Schema/. 9, 13, 93

[250] XHTML 1.0 The Extensible HyperText Markup Language, 2002. http://www.w3.org/TR/xhtml1/. 10

[251] Extensible Markup Language (XML). World Wide Web Consortium (W3C) Recommandation, 2004. http://www.w3.org/TR/2004/REC-xml-20040204/. 9

[252] XML Syntax for XQuery 1.0 (XQueryX), 2005. http://www.w3.org/TR/xqueryx/. 10

[253] XQuery 1.0: An XML Query Language. World Wide Web Consortium (W3C) Working Draft, 2005. http://www.w3.org/TR/2005/WD-xquery-20050404/. 10, 18, 29, 30

[254] XQuery 1.0 and XPath 2.0 Full-Text Use Cases. World Wide Web Consortium (W3C) Working Draft, 2006. http://www.w3.org/TR/xmlquery-full-text-use-cases/. 29

[255] N. Walsh and L. Muellner. *DocBook: The Definitive Guide*. O'Reilly, first edition, Oct. 1999. 10

[256] D. H. D. Warren and F. C. N. Pereira. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122, 1982. 81

[257] P. Wilson. Situational Relevance. *Information Storage and Retrieval*, 9(8):457–471, 1973. 36

[258] J. E. Wolff, H. Florke, and A. B. Cremers. Searching and Browsing Collections of Structural Information. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL)*, pages 141–150, Washington, D.C., USA, May 2000. IEEE Computer Society. 24, 26

[259] A. Woodley and S. Geva. NLPX at INEX 2004. In Fuhr et al. [81], pages 382–394. 26, 89

[260] A. Woodley and S. Geva. NLPX at INEX 2005. In Fuhr et al. [80]. 88, 89

[261] W. Woods, R. Kaplan, and B. Webber. The Lunar Sciences Natural Language Information System: Final Report. Technical Report BBN Report 2378, Cambridge, MA, USA, 1972. 80

[262] S. Yoo. An XML Retrieval Model based on Structural Proximities. In Fuhr et al. [78], pages 125–132. 33

[263] J. Zhou. Phrasal Terms in Real-Words IR Applications. In Strzalkowski [221], pages 215–259. 25

[264] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949. 24

[265] J. Zobel. How reliable are the results of large scale information retrieval experiments. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, Aug. 1998. ACM Press, New York City, NY, USA. 36