

THÈSE

Présentée devant

l'Université Paul Sabatier de Toulouse

en vue de l'obtention du

Doctorat de l'Université Paul Sabatier

Spécialité : **INFORMATIQUE**

Par

Lobna HLAOUA

Reformulation de Requêtes par Réinjection de Pertinence dans les Documents Semi-Structurés

Soutenue le 14 Décembre 2007, devant le jury composé de :

M. M. BOUGHANEM	Professeur à l'Université Paul Sabatier, Toulouse III	Directeur de thèse
M. C. CHRISMENT	Professeur à l'Université Paul Sabatier, Toulouse III	Examineur
M. P. GALLINARI	Professeur à l'Université Pierre et Marie Curie, Paris VI	Rapporteur
M. M. S. HACID	Professeur à Université Claude Bernard Lyon 1	Examineur
Mme. M. LALMAS	Professeur à l'Université de Queen Mary, Londres	Rapporteur
Mme. K. PINEL-SAUVAGNAT	Maitre de Conférence à l'Université Paul Sabatier, Toulouse III	Invitée
Mme. F. SEDES	Professeur à l'Université Paul Sabatier, Toulouse III	Examineur

INSTITUT DE RECHERCHE EN INFORMATIQUE DE TOULOUSE

Centre National de la Recherche Scientifique - Institut National Polytechnique - Université Paul Sabatier
Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 04. Tel : 05.61.55.66.11

Résumé

En raison de la diversité des masses d'informations, l'utilisateur a en général de plus en plus de difficultés pour accéder aux informations qui répondent à son besoin. XML est aujourd'hui présenté comme un nouveau standard permettant de mieux décrire l'information. L'accès à ce type de document soulève de nouvelles problématiques liées à la co-existence de l'information structurelle et de l'information de contenu. L'objectif des systèmes de Recherche d'Information Structurée n'est plus de renvoyer le document répondant à la requête, mais plutôt l'unité documentaire, la partie du document répondant au mieux à la requête. Afin de mieux préciser le besoin de l'utilisateur, les requêtes peuvent être améliorées via la stratégie de reformulation de requêtes.

Les travaux décrits dans cette thèse s'intéressent à la reformulation de requêtes par réinjection de pertinence dans les documents semi-structurés de type XML. Nous proposons de nouvelles approches de réinjection de pertinence en utilisant différentes sources d'évidences (le contenu et la structure).

Nous proposons dans une première approche d'enrichir le contenu de la requête initiale par des termes pertinents sélectionnés selon leur distribution dans les éléments pertinents et non pertinents ainsi que leur proximité vis-à-vis des termes de la requête initiale. Nous avons aussi proposé d'appliquer la réinjection de la pertinence négative en introduisant le facteur bruit pour la sélection des termes pertinents.

Une autre source d'évidence que nous avons aussi utilisée est l'information structurelle. Nous traduisons ainsi la notion de *structure pertinente*, dont l'existence est prouvée grâce à une étude empirique. Nous proposons l'algorithme *Smallest Common Ancestor (SCA)* pour l'extraction des structures pertinentes. Cette approche a d'abord été appliquée pour des collections homogènes. Nous proposons aussi un processus permettant d'extraire des structures pertinentes dans le cas des collections hétérogènes.

Nous proposons également de faire cohabiter les deux sources d'évidence (contenu et structure) dans une approche combinée. Nous proposons plusieurs méthodes de combinaison. L'approche "naïve" consiste à regrouper les termes pertinents et les structures pertinentes au niveau de la réécriture des requêtes. Une deuxième méthode prend en compte la sémantique des éléments pertinents pour l'extraction des termes

pertinents. Enfin une méthode flexible permet de distribuer les termes pertinents en fonction des structures pertinentes.

Quelle que soit l'approche de reformulation proposée, la réécriture de la requête est formalisée au sein d'une grammaire. L'ensemble de ces méthodes a été appliqué pour les deux types de requêtes structurées et non structurées.

Les résultats montrent l'intérêt des deux approches proposées (réinjection de contenu et réinjection de structures), la combinaison des deux sources d'évidence permettant également d'améliorer les performances.

Mots Clés : Réinjection de pertinence, Recherche d'Information Structurée, XML, termes pertinents, structure pertinente, combinaison de sources d'évidence.

Remerciements

Cette thèse est le fruit de trois années d'efforts incessants, mais aussi d'échanges bénéfiques et de collaborations fructueuses. Ce travail n'aurait pas pu aboutir sans le concours précieux et généreux de personnes qui partagent la même passion pour la recherche scientifique. C'est avec un énorme plaisir que je remercie aujourd'hui toutes les personnes qui m'ont soutenue durant ces trois années de travail pour faire réussir cette thèse.

J'adresse mes sincères remerciements à Monsieur Claude Chrisment, Professeur à l'Université Paul Sabatier, à Monsieur Gilles Zurfluh, Professeur à l'Université des sciences sociales de Toulouse, qui m'ont accueillie au sein de l'équipe SIG.

Je tiens à remercier mon Directeur de thèse et encadrant, Monsieur Mohand Boughanem, Professeur à l'Université Toulouse III, pour avoir accepté de diriger mes travaux de recherches. Je le remercie pour la patience, la gentillesse et la disponibilité dont il a fait preuve. Si j'arrive aujourd'hui au bout c'est grâce à ses conseils et ses remarques constructives. Enfin, je n'oublierai jamais les moments où il était le seul à pouvoir me réalimenter de force et de volonté, qu'il trouve ici l'expression de ma très grande gratitude.

Je tiens à exprimer ma reconnaissance à Mme Karen Pinel-Sauvagnat, Maître de conférence à l'Université Paul Sabatier de Toulouse de m'avoir fait profiter de sa propre expérience, pour ses précieux conseils et sa ferme volonté de collaboration.

Un très grand merci à mes rapporteurs, dont la lecture approfondie de ce mémoire a permis d'en améliorer la qualité : Madame Mounia Lalmas, Professeur de l'Université de Queen Mary de Londres et Monsieur Patrick Gallinari, Professeur à l'université de Marie Curie de Paris.

Je tiens également à remercier Mme Florence Sèdes, Professeur à l'Université Toulouse III, Monsieur Mohand Saïd Hacid, Professeur à l'Université Claude Bernard de Lyon et Monsieur Claude Chrisment, Professeur à l'Université Toulouse III pour

l'intérêt qu'ils ont porté à mes travaux en examinant ce mémoire et pour l'honneur qu'ils me font en participant à ce jury.

Mes remerciements vont de même à tous les membres de l'équipe SIG de l'IRIT pour leur aide et leur gentillesse. Merci aussi au personnel du laboratoire (Annie, Brigitte, Aghathe, Françoise, Jean-Pierre, ...) pour sa gentillesse ainsi que pour son aide.

Je tiens également à remercier tous les thésards qui sont présents (Mouna, Meriam, Desiré, Mohamed, Karim,...) avec qui j'ai partagé de bons moments à la salle machine, aux pauses café, au RU,.... Je n'oublie pas non plus les docteurs qui ont été des anciens thésards (Hamid, Kais,...) et qui m'ont encouragé, leurs conseils m'ont toujours servi.

Bouchra, Dana, les filles les plus adorables qui m'ont toujours supportée et soutenue. Vous avez fait preuve d'une sincère amitié, j'ai vécu des moments inoubliables avec vous, GRAND merci. Merci aussi à tous les amis que j'ai connus à Toulouse, ainsi que mon amie Wafa que j'ai connue en Tunisie, et avec laquelle j'ai partagé également de bons moments à Toulouse.

Enfin, je remercie du fond du cœur et avec un grand amour mes parents qui n'ont jamais cessé de croire en moi pendant toutes mes années d'études. Merci aussi à mes sœurs et frère, à mes oncles (Mahmoud, Mohamed,...), mes tantes et à toute la famille qui m'ont toujours encouragée.

Table des matières

Introduction Générale	1
Contexte du travail	1
Problématique	2
Contribution	4
Organisation	6
I Etat de l'Art	8
1 Recherche d'Information Structurée	9
1.1 Introduction	9
1.2 Processus de Recherche d'Information Classique	10
1.2.1 Indexation	11
1.2.2 Appariement document-requête	13
1.2.2.1 Le modèle booléen	13
1.2.2.2 Le modèle vectoriel	13
1.2.2.3 Le modèle probabiliste	14
1.2.2.4 Le modèle inférentiel bayésien	15
1.2.2.5 Les modèles de langage	17
1.2.3 Reformulation de requêtes	18
1.2.4 Evaluation	19
1.2.4.1 Mesures d'évaluation	19
1.2.5 Collections de référence	22
1.2.6 Conclusion	22
1.3 Documents semi-structurés et enjeux de la Recherche d'Information Structurée	23
1.3.1 Documents semi-structurés	23
1.3.2 Enjeux de la Recherche d'Information Structurée	24
1.3.2.1 Unité d'information recherchée	24
1.3.2.2 Problématiques spécifiques de la RI structurée	28
1.3.3 Principales stratégies en recherche d'information structurée	29
1.4 Indexation et langages de requêtes	30
1.4.1 Indexation de documents semi-structurés	30
1.4.1.1 Indexation de l'information textuelle	31
1.4.1.2 Indexation de l'information structurelle	32

1.4.2	Langages de requêtes	33
1.4.2.1	XQuery	34
1.4.2.2	XQL	35
1.4.2.3	NEXI	35
1.4.2.4	XOR	36
1.4.2.5	Autres langages d’interrogation	36
1.5	Appariement élément-requête	36
1.5.1	Modèle vectoriel étendu	37
1.5.2	Modèle booléen pondéré	39
1.5.3	Modèle probabiliste	40
1.5.4	Modèle inférentiel	41
1.5.5	Modèles de langage	42
1.5.6	Autres modèles de recherche	43
1.5.7	Modèles spécifiques aux collections de documents hétérogènes	44
1.6	Évaluation de la RIS : La campagne INEX	45
1.6.1	Collection	45
1.6.2	Requêtes	46
1.6.3	La tâche ad-hoc	48
1.6.3.1	Tâche CO	49
1.6.3.2	Tâche CAS	49
1.6.3.3	Stratégies de recherche	50
1.6.4	Autres tâches	50
1.6.4.1	Traitement automatique du langage naturel	50
1.6.4.2	Tâche Reformulation par réinjection de pertinence (Relevance Feedback)	50
1.6.4.3	Tâche Hétérogène	50
1.6.4.4	Fouille de données (Data mining)	51
1.6.4.5	Tâche interactive	51
1.6.4.6	Tâche multimedia	51
1.6.5	Jugements de pertinence	51
1.6.6	Mesures d’évaluation	52
1.7	Conclusion	54
2	Reformulation de Requêtes	55
2.1	Introduction	55
2.2	Techniques pour l’amélioration des performances des systèmes de re- cherche	56
2.2.1	Expansion et combinaison de requêtes	57
2.2.2	Combinaison de requêtes	58
2.3	Réinjection de pertinence	59
2.3.1	Motivation	59
2.3.2	Processus général de <i>RF</i>	59
2.3.3	Méthodes d’extraction des termes	61

2.3.4	Principales approches de réinjection de pertinence en RI	64
2.3.4.1	Approche de Rocchio	64
2.3.4.2	Réinjection de pertinence dans le modèle probabiliste	65
2.3.4.3	Réinjection de pertinence dans le modèle inférentiel	67
2.3.4.4	Autres propositions	68
2.3.5	Reformulation par réinjection de pertinence négative	70
2.3.6	Autres formes de Réinjection de pertinence	71
2.3.6.1	Réinjection automatique de pertinence	71
2.3.6.2	Réinjection de pertinence à itérations multiples	73
2.3.6.3	Extension interactive de requêtes	73
2.3.6.4	Combinaison d’algorithmes de réinjection de pertinence	73
2.4	Réinjection de pertinence en RIS	74
2.4.1	Problématiques de la réinjection de pertinence en RIS	74
2.4.2	Principales approches de la réinjection de pertinence en RIS	75
2.4.2.1	Approches orientées contenu	75
2.4.2.2	Approches orientées contexte	77
2.4.3	Bilan	80
2.5	Évaluation de la reformulation de requêtes	81
2.5.1	Différentes stratégies d’évaluation de la reformulation	81
2.5.2	Évaluation selon la campagne d’évaluation INEX	83
2.6	Conclusion	84

II Nouvelles Approches pour la Reformulation de requêtes en Recherche d’Information Structurée 86

3	Reformulation de requêtes par réinjection de contenu et de structures	87
3.1	Introduction	87
3.2	Motivation	88
3.3	Approche orientée Contenu	90
3.3.1	Extraction et Sélection des termes pertinents	90
3.3.1.1	Stratégie de base : Sélection par probabilité de pertinence	90
3.3.1.2	Stratégie basée sur le contexte	91
3.3.1.3	Prise en compte de la pertinence négative	94
3.3.2	Pondération des termes de la requête	96
3.3.3	Réécriture de la requête	97
3.3.4	Conclusion	99
3.4	Réinjection de la structure	99
3.4.1	La notion de structure pertinente	100
3.4.2	Extraction de la structure pertinente	102

3.4.3	Extraction de structures pertinentes dans des documents homogènes	103
3.4.3.1	Algorithmes de recherche des ancêtres communs	103
3.4.3.2	L’algorithme SCA (Smallest Common Ancestor)	104
3.4.3.3	Exemple d’application de l’algorithme SCA	105
3.4.4	Extraction des structures pertinentes dans des documents hétérogènes	108
3.4.5	Réécriture de la requête	111
3.5	Approche Combinée	114
3.5.1	Combinaison naïve	114
3.5.2	Combinaison avec dépendance contextuelle	116
3.5.3	Combinaison flexible	118
3.6	Conclusion	122
4	Evaluations & Expérimentations	123
4.1	Introduction	123
4.2	Plateforme pour l’évaluation	124
4.2.1	Le système de recherche XFIRM	124
4.2.1.1	Évaluation de pertinence des noeuds feuilles	124
4.2.1.2	Propagation de pertinence dans une requête non structurée	125
4.2.1.3	Propagation de pertinence dans une requête structurée	125
4.2.2	Rappel sur les collections de test	127
4.2.2.1	Collection de documents	127
4.2.2.2	Topics	127
4.2.2.3	Jugements de pertinence	128
4.2.2.4	Mesures d’évaluation	129
4.2.3	Stratégies d’évaluation	130
4.2.4	Résultats de base	130
4.2.5	Démarche d’évaluation	131
4.3	Échantillonnage	131
4.3.1	Choix du nombre d’éléments jugés	132
4.3.1.1	Tâche CO	133
4.3.1.2	Tâche CO+S	135
4.3.1.3	Tâche VVCAS	136
4.3.1.4	Discussion et bilan	137
4.3.2	Choix du nombre d’éléments jugés pertinents dans un échantillon	139
4.3.3	Discussion	142
4.4	Évaluation de la RF Orientée Contenu	145
4.4.1	Nombre de termes réinjectés	146
4.4.1.1	Tâche CO	146
4.4.1.2	Tâche CO+S	147
4.4.1.3	Tâche VVCAS de la collection 2005	148
4.4.1.4	Discussion	148

4.4.2	Impact des stratégies de sélection et de pondération des termes de la requête	151
4.4.2.1	Tâche CO	151
4.4.2.2	Tâche CO+S	152
4.4.2.3	Tâche VVCAS	153
4.4.3	Bilan	153
4.5	Évaluation de la reformulation Orientée-Structure	154
4.5.1	Nombre adéquat de structures à réinjecter	154
4.5.2	Réinjection de la balise ou du chemin	156
4.5.3	Bilan	157
4.6	Évaluation de la reformulation Orientée-Contenu & Structure	158
4.6.1	Tâche CO	159
4.6.2	Tâche CO+S	159
4.6.2.1	Tâche VVCAS	161
4.6.3	Conclusion	161
4.7	Autres études qualitatives	162
4.7.1	Impact des jugements de pertinence	162
4.8	Autres applications de la Réinjection de pertinence	163
4.8.1	Application de plusieurs itérations de réinjection	163
4.8.2	Utilisation de la réinjection de pertinence "aveugle"	165
4.9	Bilan	166
4.9.1	Résumé	166
4.9.2	Étude comparative	167
4.9.3	Conclusion	169
	Conclusion Générale	171
	Synthèse	171
	Perspectives	173
A	Les Documents XML	175
A.1	Structure du document XML	175
A.2	Les DOMs	178
A.3	XPath	179

Liste des tableaux

3.1	Propriétés des jugement de pertinence	101
3.2	Répartition des éléments pertinents en fonction des types de structures - INEX 2005-2006	101
3.3	Récapitulation des différences de distance entre les structures	103
3.4	Algorithme d'extraction de la structure générique.	105
3.5	Grammaire de la réécriture des requêtes par injection de structure. . .	113
3.6	Grammaire de la réécriture des requêtes par injection des structures et des mots clés.	115
3.7	Distribution des termes dans les structures génériques.	118
3.8	Les relations termes pertinents-structures génériques.	119
3.9	Grammaire de la réécriture des requêtes par injection flexible des structures et des mots clés.	120
3.10	Distribution des termes dans les structures génériques.	121
4.1	Résultats de base des collections 2005 et 2006.	131
4.2	Impact du nombre d'éléments jugés sur l'échantillon dans le cas de la tâche CO de la collection 2005	134
4.3	Impact du nombre d'éléments jugés sur l'échantillon dans le cas de la tâche CO de la collection 2006	134
4.4	Impact du nombre d'éléments jugés sur l'échantillon dans le cas de la tâche CO+S de la collection 2005	135
4.5	Impact du nombre d'éléments jugés sur l'échantillon dans le cas de la tâche CO+S de la collection 2006	136
4.6	Impact du nombre d'éléments jugés sur l'échantillon dans le cas de la tâche VVCAS de la collection 2005	137
4.7	Moyennes des éléments jugés pertinents dans les échantillons	138
4.8	Impact du nombre d'éléments jugés pertinents sur l'échantillon dans le cas de la tâche CO de la collection 2005	139
4.9	Impact du nombre d'éléments jugés pertinents sur l'échantillon dans le cas de la tâche CO+S de la collection 2005	140
4.10	Impact du nombre d'éléments jugés pertinents sur l'échantillon dans le cas de la tâche CO+S de la collection 2006	140
4.11	Impact du nombre d'éléments jugés pertinents sur l'échantillon dans le cas de la tâche VVCAS de la collection 2005	141
4.12	Moyennes des éléments jugés dans les échantillons	142

4.13	Résultats selon le nouvel échantillon de test pour les différentes tâches de recherche	143
4.14	Comparaison des résultats du nouvel échantillon et l'échantillon fixe	144
4.15	Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO de la collection 2005	146
4.16	Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO+S de la collection 2005	147
4.17	Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO+S de la collection 2006	148
4.18	Impact du nombre de termes pertinents à réinjecter dans le cas des requêtes VVCAS de la collection 2005	149
4.19	Impact des stratégies de sélection et pondération des termes dans le cas des requêtes CO de la collection 2005	151
4.20	Impact des stratégies de sélection et pondération des termes dans le cas de la tâche CO+S de la collection 2005	152
4.21	Impact des stratégies de sélection et pondération des termes dans le cas de la tâche CO+S de la collection 2006	152
4.22	Impact des stratégies de sélection et pondération des termes dans le cas de la tâche VVCAS de la collection 2005	153
4.23	Impact du nombre de structures pertinentes à réinjecter dans le cas des tâches CO, CO+S et VVCAS de la collection 2005 et la tâche CO+S de la collection 2006	155
4.24	Réinjection de structure (Element cible, Chemin spécifique et Chemin générique)	157
4.25	Reformulation de requêtes par combinaison dans le cas de la tâche CO de la collection 2005	159
4.26	Reformulation de requêtes par combinaison dans le cas de la tâche CO+S de la collection 2005	160
4.27	reformulation de requêtes par combinaison dans le cas de la tâche CO+S de la collection 2006	160
4.28	Reformulation de requêtes par combinaison dans le cas de la tâche VVCAS de la collection 2005	161
4.29	Réinjection de pertinence basée sur un jugement de pertinence généralisé	163
4.30	Réinjection de pertinence en 2 itérations	164
4.31	Réinjection de pertinence en 3 itérations	164
4.32	Réinjection de pertinence "aveugle"	165
4.33	Evaluation selon le protocole d'INEX	167
4.34	Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2005 dans le cas de la tâche CO	168
4.35	Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2005 dans le cas de la tâche CO+S	168
4.36	Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2006 dans le cas de la tâche CO+S	169

Table des figures

1.1	Le Processus en U de la Recherche d'Information	11
1.2	Modèle de réseau inférentiel bayésien simple	16
1.3	Définition du rappel et de la précision	20
1.4	Courbes de rappel-précision pour deux requêtes R1 et R2	21
1.5	Exemple d'un document XML	25
1.6	Modèle d'augmentation [65]	41
1.7	Exemple d'un article de la collection IEEE au format XML	46
1.8	Exemple d'un article de la collection Wikipédia au format XML	47
1.9	Exemple de requête CO de la collection 2005	47
1.10	Exemple de requête de la collection 2006	48
2.1	Le Processus général de l'amélioration de la recherche	57
2.2	Le Processus général de la réinjection de pertinence	60
3.1	Mécanisme de reformulation	89
3.2	Variation du bruit en fonction de fréquences	95
3.3	Recherche d'une structure générique :A	106
3.4	Recherche d'une structure générique : C	106
3.5	Recherche d'une structure générique : B	107
3.6	Recherche d'une structure générique : C	107
3.7	Présentation des structures dans un graphe orienté	111
4.1	Nombre de termes à réinjecter en fonction de la taille des requêtes.	150
A.1	Exemple d'un document XML	176
A.2	L'arbre DOM d'un document XML	178
A.3	Axes de navigation XPath	179

Introduction Générale

Contexte du travail

Chercher une information sur le web devient un geste quotidien que font des utilisateurs diversifiés en âge, en culture, en spécialité, et ayant des domaines d'intérêt variés. De nos jours, la richesse documentaire augmente, et ce essentiellement grâce à la croissance massive des documents numériques, souvent hétérogènes dans leur forme et leur contenu.

En raison de la diversité des masses d'informations, l'utilisateur a en général de plus en plus de difficultés pour accéder aux informations qui répondent à son besoin. C'est cette diversité qui a conduit le *W3C* (*World Wide Web Consortium*) à mettre en œuvre de nombreux chantiers permettant de mieux décrire l'information. Les premiers résultats en sont les langages *XML* (*eXtensible Markup Language*) [202] et *RDF* (*Resource Description Framework*) [117].

XML est aujourd'hui présenté comme un nouveau standard dont la vocation n'est rien de moins que de standardiser le formatage des données indépendamment d'un quelconque format propriétaire, quel que soit le type de données. Ce langage présente une véritable révolution dans la manière de traiter ces données.

Les documents *XML*, outre les données elles mêmes, intègrent des méta-informations et des informations structurelles. On parle ainsi de documents semi-structurés.

L'accès à ce type de document soulève de nouvelles problématiques liées à la co-existence de l'information structurelle et de l'information de contenu. Les Systèmes de Recherche d'Information (*SRI*) conçus en Recherche d'information (*RI*) traditionnelle, traitent les documents comme des unités atomiques d'information et ne répondent pas à la nature des documents structurés et semi-structurés.

Afin de valoriser au mieux l'ensemble des informations disponibles, les méthodes existantes de RI doivent être adaptées ou de nouvelles méthodes doivent être proposées. C'est dans ce contexte de Recherche d'Information Structurée (*RIS*) que se situent nos travaux. L'objectif des systèmes de RIS n'est plus de renvoyer le document répondant à la requête, mais plutôt l'unité documentaire, la partie du document répondant au mieux à la requête. Pour répondre à ce challenge, plusieurs modèles de recherche ont été proposés dans la littérature [64, 66, 67, 69, 68].

Nous nous intéressons dans nos travaux à l'application de la reformulation de requêtes en RIS afin de satisfaire l'utilisateur en lui restituant les meilleurs granules documentaires (parties de documents) répondant à son besoin.

Problématique

La recherche d'information est un processus qui se base essentiellement sur la requête exprimée par l'utilisateur pour répondre à ses besoins. En effet, quel que soit le système de recherche utilisé, le résultat d'une recherche ne peut être pertinent si la requête ne décrit pas explicitement et clairement les besoins de l'utilisateur. Or, il est généralement reconnu que l'utilisateur se contente de donner quelques mots clés. Ces derniers sont issus d'une connaissance générale sur le sujet recherché. Par conséquent, les documents renvoyés par le système de recherche peuvent ne pas satisfaire les besoins de l'utilisateur.

La reformulation de requêtes est une des stratégies qui permet d'améliorer la construction d'une requête. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin [56]. Une des techniques les plus répandues en RI est la reformulation par réinjection de la pertinence, communément appelée *Relevance Feedback (RF)*. Elle consiste à extraire à partir d'un échantillon de documents jugés pertinents par l'utilisateur les mots clés les plus pertinents, et à les ajouter à la requête.

Les travaux décrits dans cette thèse s'intéressent à la reformulation de requêtes par réinjection de pertinence dans les documents semi-structurés de type XML. La nature des documents manipulés dans ce contexte, comportant du texte et des informations structurelles sous forme de balises, réactualise la problématique de la RI classique (plein texte) en général et de la reformulation de requêtes en particulier.

- Tout d’abord au niveau de l’expression des requêtes, l’utilisateur peut exprimer ses besoins de deux manières, soit en n’utilisant que des mots clés (on parle alors de requêtes orientée contenu), ou bien en utilisant des requêtes comportant des mots clés et des contraintes structurelles (on parle alors de requêtes orientées contenu et structure).
En pratique, la plupart des utilisateurs se contentent de formuler leurs requêtes par de simples mots clés qui représentent le langage de requêtes le plus simple. Leurs requêtes peuvent également contenir des contraintes structurelles au sens large, c’est à dire des contraintes structurelles assez vagues. En effet, la formulation d’une requête bien structurée nécessite d’une part une connaissance de la structure des documents, d’autre part une certaine compétence dans le langage de requête.
- Ensuite au niveau de la recherche, contrairement à la RI traditionnelle qui considère le document comme une unité d’information atomique, la RI structurée a pour but d’identifier de manière automatique la partie du document (l’élément du document XML), répondant à la fois de manière **exhaustive** et **spécifique** à la requête de l’utilisateur. Une information est dite exhaustive si elle contient toute information répondant aux besoins de l’utilisateur et spécifique si elle ne contient que l’information répondant aux besoins de l’utilisateur.
- Enfin, au niveau du processus de *Relevance Feedback* (*RF*), il est nécessaire de tenir compte de l’information structurelle des documents, à la fois dans la requête initiale, les documents jugés par l’utilisateur et la requête reformulée.

Nous nous intéressons dans le cadre de cette thèse à la réinjection de pertinence en Recherche d’Information Structurée. Plusieurs questions se posent dans ce contexte, elles portent en général sur la manière de prendre en compte l’information structurelle. Plus précisément :

- En RI classique, l’unité documentaire jugée et donc à partir de laquelle les termes sont extraits, est le document entier. Les méthodes proposées ont montré leur intérêt en termes de rappel-précision [158], [156]. Or dans le contexte de la RIS, l’unité documentaire peut avoir différentes formes. Elle peut être le document entier ou tout élément du document. Une adaptation simpliste des méthodes de la RI classique à la RI structurée consisterait à extraire les termes pertinents à partir des éléments de différentes granularités jugés pertinents par l’utilisateur. Cette adaptation simpliste est-elle en adéquation avec la RI structurée ? Comment tenir compte du fait que les éléments peuvent être imbriqués les uns dans les autres ? Permet-elle effectivement d’améliorer les performances de la

recherche ? Au lieu de sélectionner indifféremment tous les éléments pertinents pour l'extraction des termes, doit-on au contraire prendre en compte les sémantiques différentes des éléments (par exemple, *paragraphe*, *titre*, *section*) ?

- La reformulation de requêtes s'est intéressée à enrichir la requête initiale par extraction et réinjection des termes pertinents, mais qu'en est-il de la dimension structurelle ? Est-il intéressant d'enrichir une requête avec des contraintes structurelles ? Avant de répondre à ces questions il faut tout d'abord répondre à celle-ci : Existe-t-il des structures pertinentes et comment sont-elles définies ?
- Comme nous l'avons signalé, en RI structurée, la pertinence des éléments dépend de deux dimensions : la spécificité et l'exhaustivité. Ainsi, la pertinence ne peut plus être évaluée d'une façon booléenne (pertinent/non pertinent). La problématique considérée à ce niveau est : comment prendre en compte cette graduation de la pertinence dans la reformulation de requêtes ?
- Une dernière question concernant le processus de la reformulation est la réécriture de la requête. D'une manière générale, on aura à rajouter des termes pertinents et/ou des structures pertinentes à des requêtes structurées et non structurées. La question est comment intégrer ces deux évidences dans la requête initiale ? Comment pondérer les termes ? Doit-on re-pondérer les termes originaux ? Comment rajouter des structures à des requêtes déjà structurées ? A quels groupes de mots-clés doit-on ajouter des conditions structurelles ?

Contribution

Afin de répondre aux questions listées précédemment, nous avons proposé un mécanisme complet et flexible de reformulation partant de la sélection de l'échantillon des éléments jugés jusqu'au renvoi d'un ensemble d'éléments répondant à la requête reformulée.

Les approches proposées se basent sur l'extraction et la réinjection de différentes évidences (mots clés et structures) dans la nouvelle requête. Nous avons proposé deux principales approches : l'approche orientée contenu et l'approche orientée structure.

Plus précisément, au niveau de l'**approche orientée contenu**, nous avons procédé de manière à extraire et sélectionner des termes pertinents, au sein des éléments jugés pertinents en fonction de leur probabilité de pertinence et de leur contexte. Dans notre approche, nous estimons le contexte en fonction de la position d'un terme pertinent vis à vis des termes de la requête initiale. Ces termes sont ensuite pondérés soit en se basant directement sur le score ayant permis leur sélection ou selon une formule prenant en compte leur importance dans la collection d'éléments et la collection de documents. Nous avons aussi proposé d'appliquer la réinjection de la pertinence négative en introduisant le facteur bruit pour la sélection des termes pertinents. Cette approche est évaluée pour les requêtes structurées et non structurées en utilisant deux collections provenant d'INEX¹.

Au niveau de l'**approche orientée structure**, nous avons tout d'abord effectué une étude empirique qui nous a permis de conclure qu'il existe une ou plusieurs structures pertinentes pour une requête donnée. Nous avons ramené la notion de structure pertinente à la notion de *structure générique* que nous avons définie en fonction de son apparition dans les structures des éléments jugés pertinents. Nous avons ensuite proposé un algorithme appelé *Smallest Common Ancestor (SCA)* pour l'extraction de cette dernière. Cette approche a d'abord été appliquée pour des collections homogènes (c'est à dire possédant des documents aux structures similaires), puis nous avons proposé d'étendre l'algorithme pour supporter les collections hétérogènes (c'est à dire ayant des documents aux structures différentes). Cette approche a également été appliquée pour les deux types de requêtes orientées contenu et orientées contenu et structure.

Nous avons également proposé une approche **combinée** utilisant les deux approches précédentes. Elle considère les deux sources d'évidence contenu et structure. Nous avons proposé plusieurs méthodes de combinaison, dont la plus simple, appelée "naïve", consiste à regrouper les termes pertinents et les structures pertinentes au niveau de la réécriture des requêtes. Une deuxième méthode prend en compte la sémantique des éléments pertinents pour l'extraction des termes pertinents. Enfin une méthode flexible permet de distribuer les termes pertinents en fonction des structures pertinentes. L'ensemble de ces méthodes a été appliqué pour les deux types de requêtes.

Quelle que soit l'approche de reformulation proposée la réécriture de la requête est formalisée au sein d'une grammaire. De plus, toutes les méthodes sont évaluées sur le système de recherche d'information structurée XFIRM [170]

¹INEX : *INiative for the Evaluation of XML REtrieval*, est une campagne d'évaluation de la recherche d'information dans les documents XML

élaboré au sein de notre équipe.

Enfin, toutes nos propositions ont été évaluées sur des collections standards issues des campagnes d'évaluation INEX (*INiative for the Evaluation of XML REtrieval*) 2005 et INEX 2006. Nous proposons également d'appliquer les différentes approches en mode aveugle, dans lequel l'utilisateur n'intervient pas sur le jugement des éléments pertinents. Les résultats montrent l'intérêt des deux approches proposées (réinjection de contenu et réinjection de structures). La combinaison des deux sources d'évidence permet également d'améliorer les performances de manière significative.

Organisation

Ce mémoire de thèse est composé de la présente introduction générale, de deux principales parties (état de l'art et contribution) et d'une conclusion générale dans laquelle nous présentons les principales conclusions ainsi que les perspectives de nos travaux.

Les deux principales parties sont organisées comme suit :

- La première partie, composée de deux chapitres présente un état de l'art. Dans le premier chapitre nous introduisons le cadre général de notre contribution. Nous présentons brièvement le processus de recherche d'information traditionnelle (section 1.2). Ensuite, nous détaillons les enjeux de la recherche d'information structurée (section 1.3) ainsi que les approches d'indexation et d'appariement développées dans ce cadre (section 1.4 et section 1.5). Enfin, nous présentons l'évaluation des systèmes de recherche en RIS dans la section 1.6.

Dans le deuxième chapitre, nous présentons les différentes méthodes et approches proposées pour l'amélioration des performances des systèmes de recherche en général (section 2.2). Nous décrivons ensuite les différentes propositions développées pour la reformulation des requêtes par réinjection de pertinence appliquées aux systèmes de recherche classique (section 2.3) et structuré (section 2.4). Nous présentons également les différentes méthodes d'évaluation de la réinjection de pertinence dans la section 2.5.

- La deuxième partie concerne notre contribution. Dans le premier chapitre nous détaillons nos trois propositions (l'approche orientée contenu, l'approche orientée structure, et l'approche combinée). Pour la première approche (section 3.3), nous décrivons les trois étapes sous-jacentes : l'extraction et la sélection des termes (section 3.3.1), la pondération de termes (section 3.3.2) et la réécriture de la requête (section 3.3.3). Nous

présentons également la réinjection de pertinence négative dans la section 3.3.1.3. Pour l’approche orientée structure (section 3.4), nous montrons statistiquement l’intérêt du concept de structure pertinente (section 3.4.1). Nous définissons ensuite la notion de structure générative et nous détaillons le processus d’extraction dans les sections 3.4.2 et 3.4.3. Puis nous proposons l’extension de cette approche à des collections hétérogènes dans la section 3.4.4. Enfin, nous définissons la grammaire de réécriture pour les deux types de requêtes structurées et non structurées dans la section 3.4.5. L’approche combinée propose trois combinaisons différentes de l’approche orientée contenu et l’approche orientée structure. Elles sont détaillées respectivement dans les sections 3.5.1, 3.5.2 et 3.5.3.

Dans le second chapitre, après avoir décrit notre plateforme d’évaluation dans la section 4.2, nous étudions dans la section 4.3 l’échantillonnage pour l’évaluation de nos approches. Les impacts des approches orientée contenu, orientée structure et combinée sont détaillés dans les sections 4.4, 4.5 et 4.6. Nous réalisons d’autres études expérimentales dans la section 4.7 pour évaluer l’impact de la nature du jugement de pertinence (section 4.7.1). Enfin (section 4.8), nous testons l’application de la reformulation en plusieurs itérations (section 4.8.1) et de la réinjection aveugle (section 4.8.2).

Première partie

Etat de l'Art

Chapitre 1

Recherche d'Information Structurée

1.1 Introduction

Un Système de Recherche d'Information (SRI) permet de retrouver à partir d'une collection de documents les documents pertinents répondant à une requête d'utilisateur. Trois notions clés caractérisent un SRI : document, requête et pertinence.

Un **document** désigne toute unité qui peut présenter une réponse à une requête donnée. En effet, un document peut être un morceau de texte, une page Web, une image, une séquence vidéo, etc. En outre, les documents textuels peuvent avoir plusieurs spécifications ; un document peut être un texte sans aucune structuration (appelé plein texte), mais peut aussi contenir des balises descriptives on parle alors de documents semi-structurés de type XML par exemple. Les documents peuvent aussi être complètement structurés, c'est à dire qu'ils possèdent une structure fixe comme par exemple des formulaires.

Une **requête** exprime le besoin d'information d'un utilisateur. Elle peut être exprimée selon différents langages. Le langage le plus utilisé est le langage naturel.

La **pertinence** est une notion fondamentale en RI. Elle est l'objet de tout système de recherche d'information. Elle peut être définie comme la correspondance entre un document et une requête selon le système ou l'utilisateur.

La recherche d'Information (RI) est un domaine apparu en même temps que les ordinateurs. Au début, la RI se concentrait sur les applications dans les bibliothèques. A la fin des années 1960 et au début des années 1970, G. Salton a développé le système SMART [158], qui a grandement influencé le domaine

de la RI.

Depuis les années 1990, marquées par l'apparition d'Internet, le champ d'application de la RI s'est accru, et ce à cause de la nature des documents disponibles sur le web. En particulier, les documents semi-structurés ont donné naissance à un nouveau domaine de la RI : la Recherche d'Information structurée (RIS). Ce domaine, bien qu'il présente de nouvelles problématiques, s'est servi des notions et des approches déjà développées en RI classique.

Dans ce chapitre, nous commençons par présenter brièvement le processus de RI traditionnelle (section 1.2), puis nous détaillons les enjeux de la recherche d'information structurée (section 1.3) ainsi que les différentes techniques développées pour chacune des étapes suivantes : l'indexation et l'interrogation (section 1.4) ainsi que l'appariement éléments-requêtes (section 1.5). L'évaluation des approches de RIS est enfin présentée dans la section 1.6.

1.2 Processus de Recherche d'Information Classique

Un système de recherche d'information a pour but la mise en relation des informations contenues dans le corpus documentaire d'une part, et les besoins de l'utilisateur d'autre part. Le besoin d'information d'un utilisateur est formulé à travers une requête. Le système doit retourner à l'utilisateur le maximum de documents pertinents à la requête (et le minimum de documents non-pertinents). Un SRI est composé de trois fonctions principales, représentées schématiquement par le processus U de recherche d'information [18]. Cette architecture générale est représentée sur la figure 1.1.

On distingue trois modules principaux :

- Le module d'indexation, qui permet une représentation des documents et des requêtes
- Le module d'appariement requête-document, qui permet de répondre à l'interrogation
- Le module de reformulation de la requête.

Ces trois modules sont détaillés ci-après.

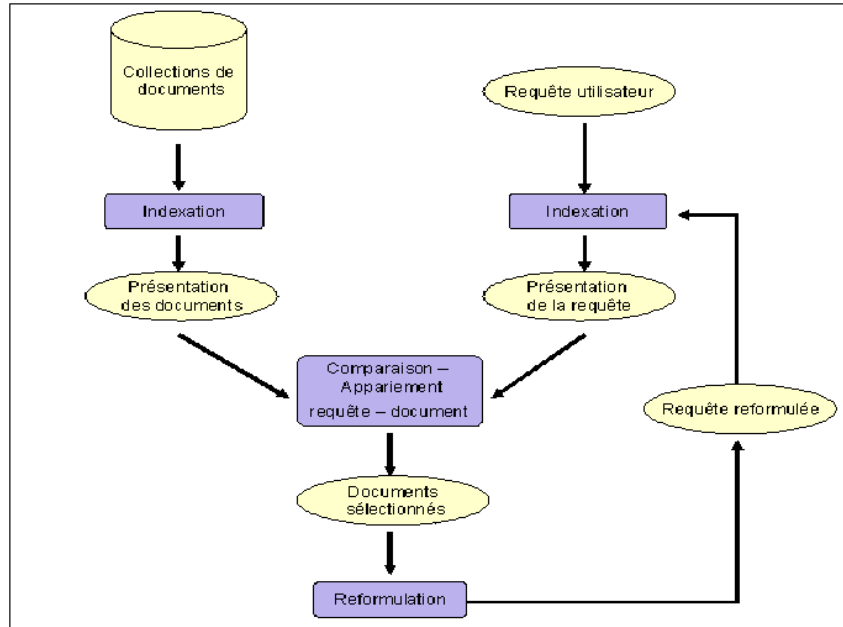


FIG. 1.1 – Le Processus en U de la Recherche d'Information

1.2.1 Indexation

Pour que le coût de la recherche soit acceptable, il convient d'effectuer une étape primordiale sur la collection de documents. Cette étape consiste à analyser les documents afin de créer un ensemble de mots-clés : on parle de l'étape d'indexation. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet de créer une vue logique du document. On entend par vue logique la représentation des documents dans le système. L'indexation peut être :

- Manuelle : chaque document est analysé par un spécialiste du domaine ou par un documentaliste
- Automatique : le processus d'indexation est entièrement informatisé
- Semi-automatique : l'indexeur intervient souvent pour choisir d'autres termes significatifs (synonymes, etc.) à partir de thésaurus ou d'une ontologie.

De manière générale, l'indexation automatique est réalisée selon les étapes suivantes :

analyse lexicale :

L'analyse lexicale (tokenization en anglais) est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est un groupe de caractères constituant un mot significatif [62]. L'analyse lexicale permet de reconnaître les espaces de séparation des mots, des chiffres, les ponctuations, etc.

L'élimination des mots vides :

Un des problèmes majeurs de l'indexation consiste à extraire les termes significatifs des mots vides (pronoms personnels, prépositions, ...). Les mots vides peuvent aussi être des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple *contenir*, *appartenir*). On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire, stoplist en anglais),
- L'élimination des mots dépassant un certain nombre d'occurrences dans la collection.

Lemmatisation :

Un mot donné peut avoir différentes formes dans un texte. On peut par exemple citer *économie*, *économiquement*, *économétrie*, *économétrique*, etc. Il n'est pas forcément nécessaire d'indexer tous ces mots et un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème, une substitution des termes par leur racine ou lemme est utilisée. Frakes et Baeza-Yates [63] distinguent cinq types stratégiques de lemmatisation : la table de consultation (dictionnaire), l'élimination des affixes (on peut citer le très connu algorithme de Porter [148]), la troncature, les variétés de successeur et la méthode des n-grammes.

Pondération des termes :

La pondération permet d'assigner aux termes leurs degré d'importance dans les documents. Un terme peut être expressif s'il apparaît suffisamment fréquemment pour être statistiquement important sans toutefois excéder une certaine limite qui le classerait dans la catégorie des mots outils (vides). La plupart des techniques de pondération sont basées sur les facteurs *TF* et *IDF* :

- *TF (Term Frequency)* : cette mesure est proportionnelle à la fréquence du terme dans le document. Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons ($\log(\text{TF})$, présence/absence,...)
- *IDF (Inverse of Document Frequency)* : ce facteur mesure l'importance d'un terme dans toute la collection. Un terme qui apparaît souvent dans la base documentaire ne doit pas avoir le même impact qu'un terme moins fréquent. Il est généralement exprimé comme suit : $\log(N/df)$, où df est le nombre de documents contenant le terme et N est le nombre total de documents de la base documentaire

La mesure $\text{TF} \cdot \text{IDF}$ donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de taille homogène.

1.2.2 Appariement document-requête

La comparaison entre le document et la requête permet de calculer une mesure appelée **pertinence système**, supposée représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction de similarité notée $RSV(Q,D)$ (*Retrieval Status Value*), où Q est une requête et D un document. Cette mesure tient compte du poids des termes dans les documents. D'une façon générale, l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'information. L'ordre dans lequel les documents susceptibles de répondre à la requête sont retournés est important. En effet, l'utilisateur se contente généralement d'examiner les premiers documents renvoyés (les 10 ou 20 premiers). Si les documents recherchés ne sont pas présents dans cette tranche, l'utilisateur considérera le SRI comme mauvais vis-à-vis de sa requête. De nombreux modèles de recherche ont été proposés dans la littérature [12]. Dans ce qui suit, nous présentons les principaux, qui ont par la suite été repris dans le cadre de la recherche d'information structurée.

1.2.2.1 Le modèle booléen

Le modèle booléen [164] est historiquement le premier modèle de RI, et est basé sur la théorie des ensembles. Un document est représenté par une liste de termes (termes d'indexation). Une requête est représentée sous forme d'une équation logique. Les termes d'indexation sont reliés par des connecteurs logiques ET, OU et NON.

Le processus de recherche mis en œuvre consiste à effectuer des opérations sur l'ensemble de documents afin de réaliser un appariement exact avec l'équation de la requête. L'appariement exact est basé sur la présence ou l'absence des termes de la requête dans les documents.

La décision binaire sur laquelle est basée la sélection d'un document ne permet pas d'ordonner les documents renvoyés à l'utilisateur selon un degré de pertinence.

1.2.2.2 Le modèle vectoriel

C'est un modèle qui préconise la représentation des requêtes utilisateurs et des documents sous forme de vecteurs, dans l'espace engendré par tous les termes d'indexation [165]. D'une manière formelle, les documents (D_j) et les requêtes Q sont des vecteurs dans un espace vectoriel des termes d'indexation (t_1, t_2, \dots, t_T) de dimension T et représentés comme suit :

$$D_j = [d_{j1}, d_{j2}, \dots, d_{jT}], Q = [q_1, q_2, \dots, q_T]$$

où d_{ji} et q_i sont respectivement les poids des termes t_i dans le document D_j et la requête Q . D'après ce modèle, le degré de pertinence d'un document relativement à une requête est perçu comme le degré de corrélation entre les vecteurs associés. Ceci nécessite alors la spécification d'une fonction de calcul de similarité entre vecteurs mais également d'une fonction de pondération des termes. La plus répandue est celle de Sparck et Needham [185] qui définit le poids d'un terme t_i dans un document d_j comme suit :

$$d_{ji} = tf_{ji} * idf_i$$

Où :

tf_{ji} : est la fréquence relative du terme t_i dans le document D_j .

idf_i : est l'inverse de la fréquence absolue du terme t_i dans la collection.

$idf_i = \log \frac{N}{n_i}$; avec n_i le nombre de documents contenant le terme t_i et N est le nombre total de documents dans la collection.

La fonction de similarité permet de mesurer la ressemblance des documents et de la requête. La mesure la plus répandue est celle du cosinus [164] :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^T q_i d_{ji}}{\sqrt{\sum_{i=1}^T q_i^2} \sqrt{\sum_{i=1}^T d_{ji}^2}}$$

Le modèle vectoriel suppose l'indépendance entre termes. En effet, la représentation vectorielle considère chaque terme séparément alors qu'on peut avoir des termes qui sont en relation sémantique entre eux.

1.2.2.3 Le modèle probabiliste

Le modèle probabiliste aborde le problème de la recherche d'information dans un cadre probabiliste. Le premier modèle probabiliste a été proposé par Maron et Kuhns [128] au début des années 1960. Le principe de base consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête. Robertson [155] résume ce critère d'ordre par le "principe de classement probabiliste", aussi désigné par PRP (*Probability Ranking Principle*).

Etant donnés une requête utilisateur notée Q et un document D , formellement, le modèle *PRP* peut être traduit de la manière suivante : pour chaque document D et chaque requête Q , *Quelle est la probabilité que ce document soit pertinent pour cette requête ?* Deux évènements sont alors possibles :

- R , D est pertinent pour Q ;
- \overline{R} , D est non pertinent pour Q .

Selon PRP, le score d'appariement entre le document D et la requête, noté $RSV(Q, D)$ [153], est donné par :

$$RSV(Q, D) = \frac{P(R/D)}{P(\overline{R}/D)} \quad (1.1)$$

En utilisant la règle de Bayes et en simplifiant, cela revient à ordonner les documents selon :

$$\frac{P(D/R)}{P(D/\overline{R})} \quad (1.2)$$

Plusieurs solutions ont été proposées pour représenter le document D et pour estimer les paramètres du modèle. Parmi elles citons *BIR* (*Binary Independence Retrieval*) [156].

Un des inconvénients de ce modèle est l'impossibilité d'estimer ses paramètres si des collections d'apprentissage ne sont pas disponibles. Pour pallier cet inconvénient, Roberston a proposé le modèle 2-poisson basé notamment sur la notion de termes élités [153], [203]. Le résultat de ces travaux est la fameuse formule *BM25*, largement discutée dans les travaux actuels de RI. La formule est la suivante :

$$RSV(Q, D) = \sum_{t \in Q} \frac{qt f \times (k_2 + 1)}{k_2 \times qt f} \times \frac{tf_{ij}(k_1 + 1) \times \log \frac{N - n_i + 0.5}{n_i + 0.5}}{k_1 \times ((1 - b) + b \frac{l_{d_j}}{avg_dl}) + tf_{ij}} \quad (1.3)$$

avec :

$qt f$: la fréquence du terme t dans la requête, l_{d_j} : la longueur du document d_j ; $l_{d_j} = \sum_{i \in d_j} tf_{ij}$, les auteurs ont aussi proposé de mesurer en octets les longueurs des documents ;

avg_dl : la longueur moyenne des documents de la collection. Elle est calculée comme suit : $avg_dl = \sum_{j \in N} \sum_{i \in T} \frac{tf_{ij}}{N}$, N le nombre de documents de la collection ; n_i le nombre de documents contenant le terme t_i , T le nombre de termes de la collection. k_1 , k_2 et b sont des constantes.

Les expérimentations ont montré que $k_1 = 1.2$, $k_2 = 0.8$, $b = 0.75$ ont donné les meilleurs résultats, en termes de performances, sur les collections TREC considérées.

1.2.2.4 Le modèle inférentiel bayésien

Les réseaux inférentiels bayésiens [198] considèrent le problème de la recherche d'information d'un point de vue épistémologique. Ils associent des variables aléatoires avec les termes de l'index, les documents et les requêtes de

l'utilisateur. Les termes de l'index et les documents sont représentés comme des nœuds. Une variable aléatoire associée avec un document d_j représente l'événement d'observer ce document. Les arcs sont dirigés du nœud document vers ses nœuds termes : ainsi, l'observation d'un document est la cause d'une augmentation de la valeur des variables associées avec ses termes d'index. La variable aléatoire associée à la requête de l'utilisateur modélise l'événement que la requête d'information spécifiée dans la requête a été vérifiée. La valeur de ce nœud requête est une fonction des valeurs des nœuds associés aux termes de la requête. Ainsi, les arcs sont orientés des nœuds des termes de l'index vers le nœud de la requête.

La figure 1.2, issue de [198], illustre un réseau inférentiel bayésien simple de pertinence d'un document vis à vis d'une requête composée de trois termes.

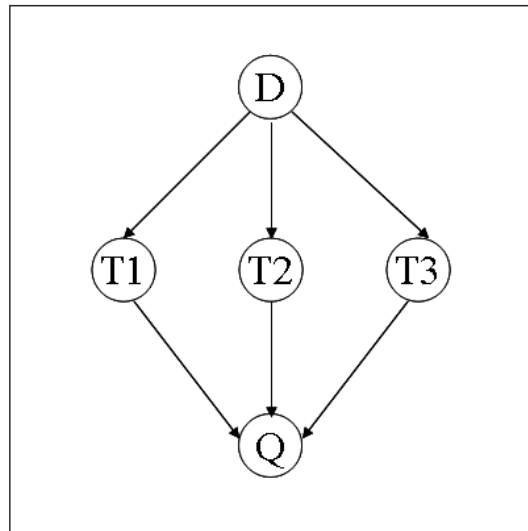


FIG. 1.2 – Modèle de réseau inférentiel bayésien simple

terme est vrai ($T1=1$, $T2=1$ ou $T3=1$), ou une combinaison de ces événements. Les trois sujets sont inférés par l'événement "le document est pertinent" ($D=1$). Par l'enchaînement de règles de probabilités, la probabilité jointe des autres nœuds du graphe est :

$$P(D, T1, T2, T3, Q) = P(D) P(T1|D) P(T2|D, T1) P(T3|D, T1, T2) P(Q|D, T1, T2, T3)$$

La direction des arcs indiquant les relations de dépendance entre les variables aléatoires, l'équation devient :

$$P(D, T1, T2, T3, Q) = P(D)P(T1|D)P(T2|D)(T3|D)P(Q|T1, T2, T3)$$

La probabilité de réalisation de la requête $P(Q = 1|D = 1)$ peut être utilisée comme score d'ordonnement des documents :

$$\begin{aligned}
P(Q = 1|D = 1) &= \frac{P(Q = 1, D = 1)}{P(D = 1)} \\
&= \frac{\sum P(D = 1, T1 = t_1, T2 = t_2, T3 = t_3, Q = 1)}{P(D = 1)} \quad (1.4)
\end{aligned}$$

Le modèle nécessite la connaissance de $P(D = [0|1])$, $P(Ti = [0|1]|D = [0|1])$, $P(Q = [0|1]|(T1, T2, \dots, Tn) \in \{0, 1\}^n)$, cette dernière étant la plus difficile à trouver car le nombre de probabilités à spécifier augmente exponentiellement avec le nombre de termes de la requête. Pour résoudre ce problème, Turtle [197] a identifié quatre formes canoniques de $P(Q|T1, T2, \dots, Tn)$: *and*, *or*, *sum* et *wsum*.

Le modèle inférentiel bayésien a été mis en oeuvre dans le système *Inquery* [8]. Le cadre probabiliste dans lequel se situe *Inquery* peut être utilisé pour formuler des requêtes simples basées sur des mots clés, des requêtes booléennes, des requêtes basées sur des expressions ou bien une combinaison des trois types [46].

D'autres travaux ont été basés sur les réseaux bayésiens. Citons par exemple les "belief networks" introduits par Ribeiro-Neto et Muntz [150], les travaux de Vogues [199], les travaux de Brini [25], [26] et ceux de Turtle [197].

1.2.2.5 Les modèles de langage

Dans les modèles de recherche classique, on cherche à mesurer la similarité entre un document D_j et une requête Q ou à estimer la probabilité que le document réponde à la requête ($P(D_j/Q)$). L'hypothèse de base dans ces modèles est qu'un document n'est pertinent que s'il ressemble à la requête. Les modèles de langage sont basés sur une hypothèse différente : un utilisateur en interaction avec un système de recherche fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver. La requête est alors inférée par l'utilisateur à partir de ces documents. Un document n'est pertinent que si la requête utilisateur ressemble à celle inférée par le document. On cherche alors à estimer la probabilité que la requête soit inférée par le document $P(Q/D_j)$. En se basant sur ce principe d'indépendance des termes (l'apparition d'un terme n'influe pas la probabilité d'existence d'un autre terme dans le document ou dans la requête), $P(Q/D_j)$ peut être réécrite de manière simple en [147] :

$$P(Q/D_j) = \prod_{i=1}^n P(T_i/D_j)$$

Où

n est le nombre de termes dans la requête et T_i est un terme de la

requête, ($1 \leq i \leq n$).

Afin de pallier le problème des termes de la requête absents des documents, (ceci conduirait systématiquement à $P(Q/D_j)=0$), on combine deux modèles de langage : celui du document et celui de la collection.

Etant donné une requête composée des termes T_1, T_2, \dots, T_n , les documents sont ordonnés selon la mesure suivante [147] :

$$P(T_1, T_2, \dots, T_n/D_j) = \prod_{i=1}^n (1 - \lambda_i)P(T_i) + \lambda_i P(T_i/D)$$

Cette mesure est une combinaison linéaire du modèle de document et du modèle de contexte du document (la collection), où :

$P(T_i/D)$ est la probabilité d'un terme important dans le modèle de document,

$P(T_i)$ est la probabilité d'un terme dans le modèle de la collection et

λ_i est une constante.

1.2.3 Reformulation de requêtes

Il est souvent difficile, pour l'utilisateur, de formuler son besoin exact en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent parfois pas. Retrouver des informations pertinentes en utilisant la seule requête initiale de l'utilisateur est toujours difficile, et ce à cause de l'imprécision de la requête. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence du système, une étape de reformulation de la requête est souvent utilisée [22, 24]. La requête initiale est traitée comme un essai (naïf) pour retrouver de l'information. Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'espoir de retrouver plus de documents pertinents. La reformulation de la requête se fait en deux étapes principales : trouver des termes d'extension à la requête initiale, et repondérer les termes dans la nouvelle requête.

La reformulation de la requête peut être interactive ou automatique. La reformulation **interactive** de la requête est la stratégie de reformulation de la requête la plus populaire [158] [23]. On la nomme communément **réinjection de la pertinence** ou "*relevance feedback*" en anglais. Dans un cycle de réinjection de pertinence, on présente à l'utilisateur une liste de documents jugés pertinents par le système comme réponse à la requête initiale. Après les avoir

examinés, l'utilisateur indique ceux qu'il considère pertinents. L'idée principale de la réinjection de pertinence est de sélectionner les termes importants appartenant aux documents jugés pertinents par l'utilisateur, et de renforcer l'importance de ces termes dans la nouvelle formulation de la requête. Cette méthode a pour double avantage une simplicité d'exécution pour l'utilisateur qui ne s'occupe pas des détails de la reformulation, et un meilleur contrôle du processus de recherche en augmentant le poids des termes importants et en diminuant celui des termes non importants.

Dans le cas de la reformulation **automatique**, l'utilisateur n'intervient pas. L'extension de la requête peut être effectuée à partir d'un thesaurus, qui définit les relations entre les différents termes de l'index et permet de sélectionner de nouveaux termes à ajouter à la requête initiale. Le thesaurus regroupe plusieurs informations de type linguistique (équivalence, association, hiérarchie) et statistique (pondération des termes). La construction du thesaurus se fait généralement pendant le processus d'indexation, et peut être automatique ou interactive. Parmi les thesaurus construits automatiquement, on peut citer un thesaurus basé sur les similarités [149], un thesaurus statistique [49], ou bien des mini-thesaurus construits seulement d'après la requête et à partir de techniques de clustering [11].

Enfin, dans le cadre de la reformulation automatique, on peut citer également la réinjection de pertinence automatique : c'est aussi ce qu'on appelle la **réinjection de pertinence aveugle**. Dans ce cas, on applique le même principe de la réinjection de pertinence mais en considérant les n premiers documents renvoyés par le système comme pertinents [45], [138]. On trouvera plus de détails sur la reformulation de requêtes dans le chapitre 2.

1.2.4 Evaluation

L'évaluation constitue une étape importante lors de la mise en oeuvre d'un modèle de recherche d'information puisqu'elle permet de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et enfin de fournir des éléments de comparaison entre modèles.

1.2.4.1 Mesures d'évaluation

L'évaluation nécessite la définition d'un ensemble de mesures et de méthodes d'évaluation, ainsi que de collections de test assurant l'objectivité de l'évaluation.

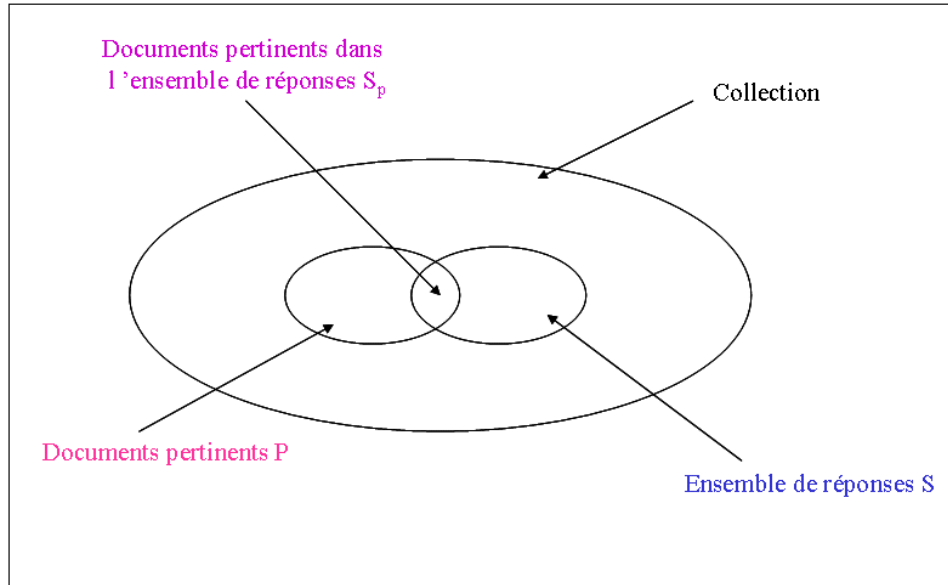


FIG. 1.3 – Définition du rappel et de la précision

Nous présentons dans ce qui suit les deux principales mesures d'évaluation : le rappel et la précision.

Rappel et précision :

Les taux de rappel et de précision sont les mesures les plus utilisées pour l'évaluation d'une recherche. Soient, comme illustré dans la figure 1.3 :

- P l'ensemble des documents pertinents pour une requête Q,
- S l'ensemble des documents retrouvés par le système,
- S_p l'ensemble des documents pertinents sélectionnés par le système et
- |X| le cardinal de l'ensemble X.

Les taux de rappel et de précision sont définis comme suit :

- Le taux de rappel est la proportion de documents pertinents qui ont été retrouvés :

$$rappel = \frac{|S_p|}{|P|} \quad (1.5)$$

- Le taux de précision est la proportion de documents retrouvés qui sont effectivement pertinents par rapport à l'ensemble des documents pertinents selon le système :

$$precision = \frac{|S_p|}{|S|} \quad (1.6)$$

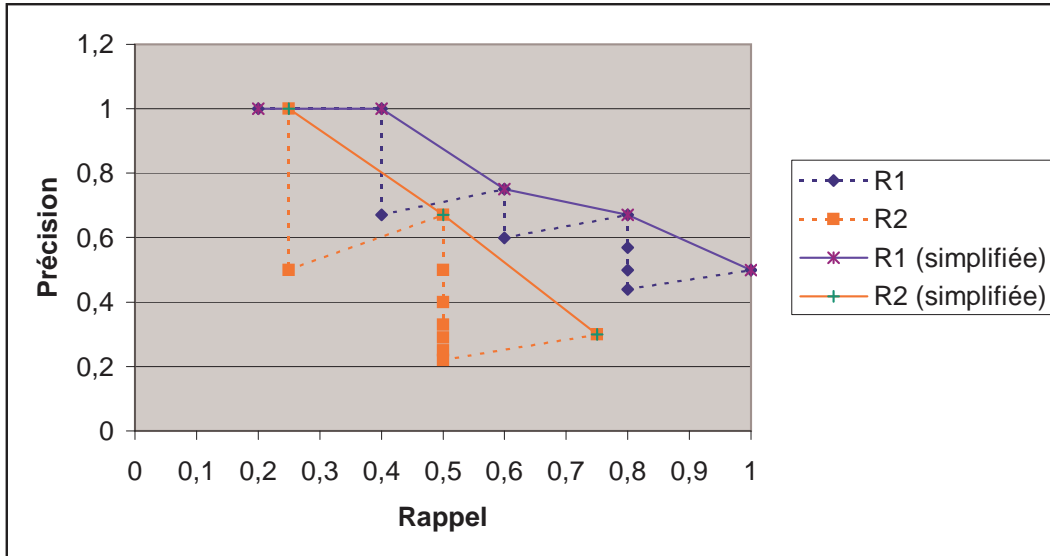


FIG. 1.4 – Courbes de rappel-précision pour deux requêtes R1 et R2

Courbe de Rappel-Précision

On observe les performances des systèmes de recherche à travers des courbes de variation de la précision en fonction des points de rappel appelées courbes de Rappel-Précision. La figure 1.4 illustre des calculs de précision et de rappel sur deux requêtes différentes.

Pour avoir une évaluation de la performance du système sur toutes les requêtes et non pas sur une seule, on calcule une **précision moyenne** à chaque niveau de rappel appelé **MAP** (*Mean Average Precision*). Pour ce faire, il faut unifier les niveaux de rappel pour l'ensemble des requêtes. On retient généralement 11 points de rappel standards, de 0 à 1 à pas de 0.1. Les valeurs de précision non obtenues à partir des valeurs de rappel sont calculées comme suit, par interpolation linéaire.

Pour deux points de rappel, i et j , $i < j$, si la précision au point i est inférieure à celle au point j , on dit que la précision interpolée à i égale la précision à j . Formellement :

$$p'_i = \max(p_i, p_j), \forall i < j \quad (1.7)$$

où p'_i est la précision interpolée au point de rappel i , et p_i est la vraie précision au point de rappel i . Cette interpolation est encore discutable, mais présente un intérêt dans l'évaluation de SRI [167].

Le système parfait trouverait seulement les documents pertinents, avec une précision et un rappel de 100%. En pratique, les mesures de rappel et précision

évoluent inversement, ce qui signifie que le courbe interpolée de précision en fonction du rappel est décroissante. Plus la courbe est élevée, plus le système est performant.

D'autres mesures ont été définies dans le cadre de TREC [200] afin d'évaluer les requêtes aux faibles performances :

- (*%n*) le pourcentage des requêtes n'ayant pas de documents pertinents dans les 10 premiers documents retournés par le système.
- *area* la surface au-dessous de la courbe de MAP.

1.2.5 Collections de référence

Les mesures d'évaluation des SRI permettent certes de les comparer, mais encore faut-il que les évaluations soient faites sur les mêmes bases documentaires. De nombreux projets basés sur des corpus d'évaluation se multiplient depuis des années. On peut par exemple citer la Collection CACM ou la Collection ISI. La campagne d'évaluation TREC (*Text Retrieval Conference*), co-organisée par le NIST et la DARPA, a commencé en 1992. Elle a pour but d'encourager le recherche documentaire basée sur de grandes collections de test, tout en fournissant l'infrastructure nécessaire pour l'évaluation des méthodologies de recherche et de filtrage d'information. De plus amples informations sont disponibles sur le site : <http://trec.nist.gov>. Pour chaque session de TREC, un ensemble de documents et de requêtes est fourni. Les participants exploitent leurs propres systèmes de recherche sur les données et renvoient à NIST une liste ordonnée de documents. NIST évalue ensuite les résultats.

1.2.6 Conclusion

Dans cette section, nous avons présenté le processus de la Recherche d'Information dans le cadre de la RI traditionnelle. Cette dernière, comme nous l'avons vu, a pour but de restituer des documents pertinents dans leur totalité. L'utilisateur se trouve alors obligé de les parcourir pour trouver l'information souhaitée.

L'apparition des documents structurés, de type XML par exemple, a apporté une nouvelle problématique et a conduit à de nouveaux objectifs liés à la manière d'exploiter les différentes caractéristiques de ce type de document. Le but des systèmes de recherche traitant des documents structurés ou semi-structurés est de retourner les parties de documents qui satisfont les besoins de l'utilisateur. Grâce aux informations structurelles contenues dans les do-

cuments, l'utilisateur peut en outre exprimer ses requêtes en intégrant des contraintes sur le contenu ainsi que sur la structure de l'information recherchée.

Avant de présenter les travaux effectués dans ce cadre, nous présentons dans la section suivante une brève description des documents semi-structurés et détaillons les problématiques de la Recherche d'Information Structurée (RIS).

1.3 Documents semi-structurés et enjeux de la Recherche d'Information Structurée

1.3.1 Documents semi-structurés

La structure des documents est définie par des balises encadrant les fragments d'informations. Une *balise* (ou *tag* ou *label*) est une suite de caractères encadrés par "<" et ">", comme par exemple `<nombalise>`. Un *élément* est une unité syntaxique identifiée, délimitée par des balises de début `< b >` et de fin `< /b >`, comme par exemple `<mabalise> mon texte </mabalise>`. Les éléments peuvent être imbriqués comme le montre le document exemple de la figure 1.5, mais ne doivent pas se recouvrir. Les *attributs* des éléments sont intégrés à la balise de début en utilisant la syntaxe `nomattribut=valeur`. Par exemple, `<mabalise monattribut='mavaleur'>texte </mabalise>`.

Les formats SGML (*Standard Generalized Markup Language*) [74] et XML (*eXtensible Markup Language*) [2] permettent de produire des documents *structurés* ou *semi-structurés*. Les documents *structurés* possèdent une structure régulière, ne contiennent pas d'éléments mixtes (c'est à dire d'éléments contenant du texte ET d'autres éléments) et l'ordre des différents éléments qu'ils contiennent est généralement non significatif.

Les documents *semi-structurés* quant à eux sont des documents qui possèdent une structure flexible et des contenus hétérogènes. La modification, l'ajout ou la suppression d'une donnée entraîne une modification de la structure de l'ensemble.

Dans notre contexte, nous nous intéressons plus particulièrement à la recherche d'information dans des documents semi-structurés, les documents structurés servant plutôt à conserver des données au sens bases de données. Par abus de langage, on parlera cependant de *RI structurée*. Le format XML nous permettra d'illustrer nos propos.

XML [2] est un langage standard pour l'échange des données semi-structurées. XML est en quelque sorte un langage HTML (*Hyper Text Markup Language*) amélioré permettant de définir de nouvelles balises et de structurer des documents. Le langage XML a la capacité de décrire n'importe quel domaine de données grâce à son extensibilité. Il permet de structurer, et de poser le vocabulaire et la syntaxe des données qu'il va contenir. Les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML).

XML a été mis au point par le XML Working Group sous l'égide du *World Wide Web Consortium* (W3C) dès 1996. C'est un sous ensemble de SGML, défini par le standard ISO8879 en 1986, utilisé dans le milieu de la Gestion Electronique Documentaire (GED). XML reprend la majeure partie des fonctionnalités de SGML, et il s'agit donc d'une simplification de SGML afin de le rendre utilisable sur le web.

La DTD (*Document Type Definition*) associée au document décrit la structure générique du document : elle contient l'ensemble des balises qu'il est possible d'inclure, ainsi que des relations de composition entre ces balises.

Contrairement à SGML, il n'est pas obligatoire d'associer une DTD à un document XML. Notons que l'on assiste aujourd'hui au développement d'une nouvelle forme de grammaire, qui permet de définir des éléments plus complexes et possède un typage des données plus riche, les *XML-schémas* [59].

Une classe de document possède donc une structure *générique* définie par la DTD (ou le schéma XML) alors qu'un document instance de cette classe possède une structure *spécifique*, exprimée par l'imbrication des éléments via leurs balises. On trouvera plus de détails sur le format XML ainsi que sur les technologies DOM et XPath associées en annexe A. Notons simplement que DOM (*Document Object Model*) permet une représentation arborescente des documents et que XPath permet de naviguer au sein de la structure des documents.

Nous présentons dans les sections suivantes les problématiques et solutions proposées dans la littérature pour la RI structurée.

1.3.2 Enjeux de la Recherche d'Information Structurée

1.3.2.1 Unité d'information recherchée

Le but des systèmes de recherche d'information est d'apporter une réponse non nécessairement exacte (au sens base de données) aux besoins en information

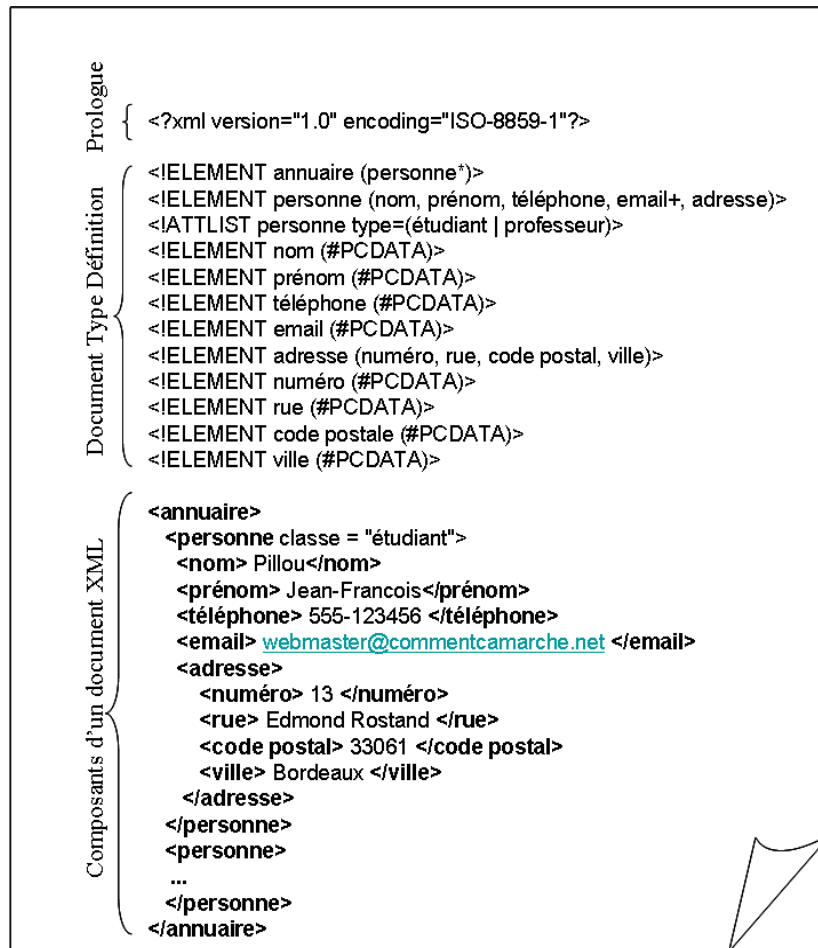


FIG. 1.5 – Exemple d'un document XML

de leurs utilisateurs. Ces derniers s'intéressent rarement à une représentation ou à une structuration précise des collections consultées. S'ils sont capables de préciser leur requête parce qu'ils connaissent la ou les collections interrogées, les réponses fournies par le système ne devront être que plus précises.

En recherche d'information traditionnelle, les SRI, tant dans leur modèle de représentation des données que dans les résultats qu'ils renvoient, traitent les granules des collections (documents) dans leur globalité. Les notions de documents logique et de document physique sont alors confondues. Cependant, un document possède souvent des contenus hétérogènes, et l'utilisateur doit alors aller chercher l'unité d'information pertinente à sa requête au milieu des autres thèmes abordés par le document. Une solution à ce problème serait de dissocier l'unité d'information logique renvoyé à l'utilisateur de l'unité d'information physique de la collection.

Les documents semi-structurés, en permettant le balisage des contenus des documents, réactualisent cette problématique, et permettent ainsi de traiter l'information avec une granularité plus fine. Le but des SRI traitant des documents semi-structurés est alors d'identifier des parties des documents les plus pertinentes à une requête donnée. Ceci nous amène à affiner le concept de granule (unité d'information) renvoyée à l'utilisateur. Une unité d'information est un volume d'information auto-explicatif, c'est à dire que l'information contenue ne dépend pas d'une autre pour être comprise. Le but des SRI dans notre contexte est alors de renvoyer des unités d'information auto-explicatives à l'utilisateur, et non des points d'entrée dans les documents : les résultats renvoyés doivent se suffire à eux même.

Dans le cadre des documents XML, l'unité d'information correspond à un noeud également appelé *élément* dans la suite du document. Chaque élément est évalué selon les deux notions suivantes : l'exhaustivité et la spécificité [40], [114].

On dit qu'une unité d'information est exhaustive à une requête si elle contient toutes les informations requises par la requête et qu'elle est spécifique si tout son contenu concerne la requête.

Dans [40], on trouve "le principe recherche dans les documents structurés" : un système devrait toujours renvoyer la partie la plus spécifique d'un document répondant à une requête. Cette définition suppose que le système sélectionne d'abord des documents entiers répondant de manière exhaustive à une requête, puis extrait de ces documents les unités d'informations les plus spécifiques. La plupart des SRI traitant des documents semi-structurés permettent une recherche directe des unités d'information sans passer au niveau de granularité document entier. Le principe de la recherche dans les documents semi-structurés pourrait donc être étendu ainsi : un système devrait toujours retrouver l'unité d'information exhaustive et spécifique répondant à une requête. Dans des cor-

pus de documents XML, chercher les nœuds les plus exhaustifs et spécifiques pour une requête revient donc à trouver les sous-arbres de taille minimale pertinents à la requête.

De part leur structure, l'utilisateur interrogeant les corpus de documents XML peut formuler deux types de requêtes, selon sa connaissance du corpus :

- des requêtes portant sur le contenu seul des unités d'information : ces requêtes sont composées de simples mots clés, et l'utilisateur laisse le SRI décider de la granularité de l'information à renvoyer.
- des requêtes portant sur la structure et le contenu des unités d'information, dans lesquelles l'utilisateur spécifie des besoins précis sur certains éléments de structure. Dans ce type de requête, l'utilisateur peut utiliser des conditions de structure pour indiquer le type des éléments qu'il désire voir renvoyer, mais aussi plus simplement pour préciser ses besoins.

Afin de permettre ces différentes recherches, les techniques de la recherche d'information traditionnelle doivent être adaptées, citons par exemple [114], [70], [130],[72], [10], [97], [146] et [174], ou de nouvelles méthodes doivent être proposées pour l'indexation, l'interrogation ou encore la recherche et le tri des unités d'information.

Avant de détailler ces différentes problématiques dans la section suivante, citons le travaux effectués dans [84] pour définir les caractéristiques des unités d'information les plus appropriées. Les auteurs se sont basés sur une analyse de structure (nombre d'élément dans un document, chemin des éléments, nombre des mots dans chaque élément,...), une analyse du contenu (fréquence des mots dans des éléments, leurs poids) et des statistiques. Ils ont défini le *Ratio* du type des mots comme le rapport entre le nombre de types de mots dans un élément et le nombre total de mots. Si le *Ratio* est élevé, l'unité n'est pas informative. Ils ont également considéré une taille seuil pour définir l'unité informative. Ce problème a été aussi traité au niveau de la recherche dans [87]. L'inconvénient principal de ces travaux est qu'ils peuvent difficilement se généraliser à d'autres collections.

1.3.2.2 Problématiques spécifiques de la RI structurée

La problématique dans le cadre de l'**indexation** se situe essentiellement au niveau de l'information structurée. Dans le cas des documents plein-texte, le contenu textuel est traité afin de trouver les termes les plus représentatifs des documents. Dans ce cas des documents semi-structurés, la dimension structurée s'ajoute au contenu, et les questions suivantes se posent alors :

- quelle unité doit-on indexer de la structure des documents ?
- comment relier cette structure au contenu même du document ?
- en fonction de quelle dimension (niveau élément, documents, collection) doit-on pondérer les termes d'indexation ?

Considérons à présent l'**interrogation** des documents. Il s'agit ici de permettre à l'utilisateur d'exprimer des besoins diversifiés (concernant le contenu des documents et/ou la structure), et ce de manière simple.

La dernière problématique concerne les **modèles de recherche** et de **tri** des unités d'information. La problématique traditionnelle liée à l'évaluation de la pertinence d'une information vis-à-vis d'une requête reste d'actualité, mais elle se complique et implique d'autres questions dans le cadre des documents XML, notamment en ce qui concerne la structure.

Les requêtes orientées contenu, qui sont de loin les plus simples pour l'utilisateur, imposent au SRI de décider la granularité appropriée de l'information à renvoyer, et donc d'évaluer l'exhaustivité et la spécificité des éléments.

Dans le cadre des requêtes orientées contenu et structure, deux cas sont possibles. Tout d'abord, l'utilisateur peut spécifier le type des éléments à renvoyer par le système. Dans ce cas la dimension de spécificité n'a plus réellement de sens, puisque l'utilisateur précise la granularité de l'information qu'il désire. Cependant, le contenu des éléments de structure ainsi que les expressions de chemins présentes dans la requête doivent pouvoir être traitées de manière vague. En d'autres termes, la pertinence des informations structurées doit pouvoir être évaluée, et l'arbre de la requête et l'arbre du document doivent pouvoir être comparés de façon non stricte. Le second cas concerne les requêtes pour lesquelles l'utilisateur exprime des conditions sur la structure des documents, mais sans préciser ce qu'il cherche exactement. Si le problème de l'évaluation de la pertinence des informations structurées se pose de nouveau, vient s'y ajouter, comme dans les requêtes orientées contenu, celui de la granularité de l'information à renvoyer.

1.3.3 Principales stratégies en recherche d'information structurée

La notion de recherche de granules de documents a été déjà développée dans la recherche de passage [206], [214], [108],[30] dont le but est de retrouver des passages pertinents dans le texte des documents. Ces approches proposent de renvoyer une partie de document en se basant sur un découpage physique du document. L'application de la recherche de passage est limitée aux documents texte ayant des tailles homogènes.

De nombreuses approches ont été développées pour traiter spécifiquement la recherche d'information dans des corpus de documents semi-structurés. On distingue deux différentes stratégies :

1. Les approches basées sur **la modélisation des données**. Le but est de développer des modèles de données permettant la représentation et l'interrogation en tenant compte à la fois du contenu et de la structure [6], [129], [195]. Dans ce cas, les documents XML sont considérés comme une base de données, dont les champs correspondraient aux éléments et attributs définis dans la DTD (ou le schéma) des documents. Des modèles de recherche ont été développés par la communauté des Bases de Données (BD). Au niveau de l'indexation, la communauté BD procède de manière à ce que toutes les informations textuelles et structurelles des documents soient stockées au sein de tables de bases de données. Des langages de requêtes associés ont été proposés par la communauté BD. Ils sont généralement liés à la syntaxe du langage SQL tout en permettant de spécifier des contraintes sur la structure des documents. Au niveau de l'appariement, la pertinence est généralement calculée d'une manière booléenne. De ce fait, seuls les éléments qui répondent exactement à la requête sont renvoyés.

2. Les approches basées sur **l'agrégation de représentation ou de pertinence**. La pertinence des parties de documents est calculée par agrégation des représentations ou de la pertinence de leur propre contenu ou par agrégation des pertinences des parties auxquelles elles sont reliées [114], [110], [111]. Dans ce cas, les documents XML sont considérés comme un ensemble de documents semi-structurés où les balises servent uniquement à décrire la structure logique des documents. Cette approche a été prise en charge par la communauté de la Recherche d'Information. Les mêmes techniques d'extraction des termes et d'indexation que de la RI classique sont maintenues pour l'indexation de l'information textuelle. D'autres approches spécifiques sont développées pour indexer l'information structurelle. Quant aux langages de requêtes, ils restent

beaucoup plus simples que ceux proposés en BD en se rapprochant du langage naturel avec une extension pour exprimer les contraintes structurelles. En RI, les résultats renvoyés à l'utilisateur sont triés selon le degré de similarité.

Bien que les stratégies orientées BD et orientées RI s'avèrent différentes au niveau de l'indexation, de l'interrogation et de l'appariement requête-élément, des modèles hybrides ont été développés [125].

Quelle que soit la manière d'interpréter les documents, les deux communautés, RI et BD doivent résoudre des problématiques liées à l'indexation de l'information textuelle et structurelle, ainsi que fournir des langages d'interrogation permettant l'expression de contraintes sur le contenu et la structure des éléments.

1.4 Indexation et langages de requêtes

1.4.1 Indexation de documents semi-structurés

L'indexation permet de représenter les documents de manière à faciliter la recherche et de la rendre plus efficace. En RI structurée, l'objectif de l'indexation n'est plus seulement de stocker l'information textuelle mais aussi l'information structurelle et de pouvoir présenter les relations entre les deux types d'information. De ce fait, un schéma d'indexation de document XML devrait principalement permettre la reconstruction du document XML décomposé dans des structures de stockage et la recherche par mot clé et par expressions de chemin sur la structure XML.

L'indexation des documents semi-structurés est caractérisée alors par le schéma de stockage des documents, et les types de transformation possible entre les documents XML et les structures de stockage [77].

Un schéma de stockage peut être conçu soit selon des approches orientées Système de Gestion de Bases de Données soit selon des modèles de stockages XML natifs qui permettent le stockage des documents complets ou des parties de documents dans des fichiers et ne réalisent pas de transformation en tables (cas des SGBD relationnels).

Les approches de transformation (*mapping*) entre les documents XML et les structures de stockage [211] sont généralement basées soit sur un modèle où l'index est fixe et connu à l'avance [119], [61], [79], [122], soit sur la structure

logique des documents XML (ou leur schéma) : la structure d'index varie alors selon les collections. Dans ce cas la construction du schéma d'index se fait automatiquement, en prenant en compte la sémantique de l'application [58], [20], [54].

Un index en RI structurée est alors composé d'une part d'une description des termes et leurs relations avec les unités structurales et d'autre part d'une description de l'information structurale traduite par des relations de hiérarchie.

1.4.1.1 Indexation de l'information textuelle

L'indexation de l'information textuelle, c'est-à-dire l'extraction et la pondération des termes, est similaire à la RI classique. Sa spécificité dans les documents semi-structurés et notamment les documents XML, réside dans la description des relations entre les termes et l'information structurale : c'est ce qu'on appelle la "portée des termes d'indexation".

Portée des termes d'indexation

Pour relier les termes à l'information structurale, deux solutions ont été proposées dans la littérature : une qui procède de manière à agréger le contenu des nœuds (on parle de *sous-arbres imbriqués*) et une deuxième qui indexe tous les contenus des nœuds séparément (*unités disjointes*) :

- sous-arbres imbriqués : On considère que le contenu de chaque nœud de l'index est une unité atomique [7], [180], [106]. Les termes des nœuds feuilles sont donc propagés dans l'arbre des documents. Comme les documents XML possèdent une structure hiérarchique, les nœuds de l'index sont imbriqués les uns dans les autres et par conséquent, l'index contient des informations redondantes. Dans [133], Mass et al. ont considéré que seuls quelques types de nœud sont informatifs (dans la collection d'INEX 2005, ils ont par exemple sélectionné : *article*, *paragraphe (p)*, *section (sec)*, *sous-section (ssec)*). Un sous-index est ensuite construit pour chaque type de nœud. L'index est l'ensemble des sous index associés.
- unités disjointes : Dans ces approches, le document XML est décomposé en unités disjointes, de telle façon que le texte de chaque nœud de l'index est l'union d'une ou plusieurs parties disjointes [139], [65], [71], [111], [159], [10]. Les termes des nœuds feuilles sont uniquement reliés à un et un seul nœud.

Une fois les unités d'indexation spécifiées, il reste à pondérer les termes. Cette tâche est une adaptation des fonctions de pondération déjà proposées en

RI classique.

Pondération des termes d'indexation

En réalité, le problème de pondération n'est traité que dans les approches orientées recherche d'information. Les approches orientées BD se contentent de stocker le texte des documents sous forme de chaînes de caractères.

Des processus similaires à ceux de la RI traditionnelle ont été adaptés dans les approches orientées RI. Par analogie à la mesure *idf* [213], [75], des auteurs [207], [79] ont proposé d'utiliser la mesure *ief* (*Inverse Element Frequency*). Dans [212], Zargayouna et *al.* ont adapté la mesure *tf-idf* (*Term Frequency- Inverse Document Frequency*) pour l'appliquer au niveau des unités d'indexation de manière à ce que le calcul des poids des termes tienne compte du contexte (élément) dans lequel ils apparaissent. Les auteurs ont défini ainsi *tf-itdf* (*Term Frequency- Inverse Tag and Document Frequency*), qui permet de calculer la force discriminatoire d'un terme dans un élément (caractérisé par une balise) relatif à un document.

Le calcul du poids d'un terme peut tenir compte non seulement de son importance dans l'élément dans lequel il apparaît mais en outre de son importance dans le contenu du noeud même, dans le contenu de ses descendants, dans le contenu de ses voisins directs et dans le contenu des noeuds auquel il est relié [111]. Dans ce dernier article, le calcul est effectué par l'opérateur d'agrégation OWA [210].

Que ce soit pour reconstruire les chemins des éléments ou pour répondre aux contraintes structurelles spécifiées dans une requête, il est nécessaire d'indexer l'information structurelle de manière à avoir une description complète de chaque élément.

1.4.1.2 Indexation de l'information structurelle

Différentes approches ont été proposées dans la littérature pour indexer l'information structurelle selon des granularités variées [126]. On distingue trois types d'approches pour l'indexation de l'information structurelle :

– **Indexation basée sur des champs** [80]

Il s'agit de la méthode d'indexation semi-structurée prenant en compte la structure la plus simple. Un document est représenté comme un ensemble de champs (par exemple : *titre*, *auteur*, *abstract*) et de contenu associé à ces champs. Pour permettre une recherche restreinte à certains champs, les termes de l'index sont construits en combinant le nom du champ avec les termes du contenu.

– Indexation basée sur des chemins

Les techniques basées sur les chemins [107], [98] ont pour but de retrouver rapidement des documents ayant des valeurs connues pour certains éléments ou attributs. Il s'agit aussi de faciliter la navigation de façon à résoudre efficacement des expressions XPATH et d'utiliser des index pleins textes sur les contenus. En conséquence les solutions proposées utilisent des index de chemins donnant pour chaque valeur répertoriée d'un chemin de balises la liste des documents contenant un élément atteignable par ce chemin et ayant cette valeur. Dans ces approches, il est difficile de retrouver les relations ancêtres-descendants entre les différents nœuds des documents.

– Indexation basée sur des arbres

Les nœuds d'un arbre sont numérotés dans l'index de façon à pouvoir reconstruire la structure arborescente des documents. Cette approche a été adaptée dans plusieurs systèmes de recherche, parmi lesquels citons [119], [61], [102], [179], [171]. Dans l'index ANOR (*inverted index for All NOdes without Replication*)[119], les documents structurés sont agrégés en un seul arbre interprété ensuite comme un k-arbre virtuel : pour chaque nœud, il existe un identifiant unique (UID). Dans le cas du système XFIRM [171], pour chaque document, un nœud est identifié par les 2 valeurs de pré-ordre¹ et post-ordre² qui permettent de retrouver les relations de hiérarchie entre les différents nœuds.

1.4.2 Langages de requêtes

Comme nous l'avons vu précédemment, lorsqu'ils s'interrogent des collections de documents semi-structurés, les utilisateurs devraient pouvoir exprimer leurs besoins selon deux catégories de requêtes :

1. des requêtes composées de simples mots clés comme en RI. C'est le cas lorsque les utilisateurs n'ont pas d'idée précise de ce qu'ils recherchent ou n'ont pas de connaissance concernant la structure des documents.
2. des requêtes composées de contraintes sur le contenu (donc de mots clés) et de contraintes structurelles. C'est le cas lorsque les utilisateurs ont au moins une connaissance partielle de la structure de la collection qu'ils interrogent.

La majorité des langages de requêtes proposés dans la littérature sont issus de la communauté des bases de données. D'une manière générale, les langages de

¹Un parcours préfixé permet d'assigner à chaque nœud visité une valeur croissante de préordre avant que ses nœuds descendants ne soient aussi récursivement visités de gauche à droite.

²D'une manière inverse, la valeur de post-ordre d'un nœud lui est assignée lors d'un parcours postfixé, c'est à dire une fois que tous ses nœuds descendants ont été visités de gauche à droite.

requêtes doivent supporter à la fois des contraintes portant sur le contenu et sur la structure. De plus l'intégration de fonctions des systèmes documentaires nécessite la prise en compte de requêtes par liste de mots clés du type :

CONTAINS(<élément>, collection de mots clés)

Dans ce qui suit nous présentons brièvement quelques langages de requêtes adaptés à XML.

1.4.2.1 XQuery

XQuery [60] est un langage de requête pour XML proposé par le W3C. Il se base sur XPath pour extraire et travailler sur des fragments de documents XML. Les requêtes basiques de XQuery sont identiques à celles définies par XPath. Si l'on désire faire des requêtes simples, XPath peut donc parfaitement suffire.

XQuery est intéressant dès le moment où l'on désire faire des requêtes complexes ou encore faire appel à la récursivité. XQuery peut être perçu comme un sur-ensemble de SQL. Les fonctionnalités de SQL sur les tables (collection de tuples) sont étendues pour supporter des opérations similaires sur les forêts (collection d'arbres). Ces extensions ont conduit à intégrer les fonctions suivantes : projection d'arbres sur des sous-arbres, sélections d'arbres et de sous-arbres en utilisant des prédicats sur les valeurs des feuilles, utilisation des variables dans les requêtes pour mémoriser un arbre ou itérer sur des collections d'arbres extraits de collection en utilisant des jointures d'arbres, ré-ordonnement des arbres, imbrication de requêtes, calcul d'agrégats, utilisation possible de fonctions utilisateur.

De plus, XML étant fait pour gérer des documents, XQuery supporte les fonctions des systèmes documentaires : en particulier, un prédicat CONTAINS est intégré pour la recherche par mots-clés.

On trouvera ci-dessous un exemple simple d'une requête XQuery qui retourne les numéros de téléphone de toutes les personnes habitant à Toulouse :

*For \$P in ("annuaire.xml")//Personne
Where \$P/Adresse/Ville="Toulouse"
return \$P/Téléphone*

1.4.2.2 XQL

Une motivation importante pour la conception de XQL [3] est la réalisation que XML a son propre modèle implicite de données, qui n'est ni celui des bases de données relationnelles traditionnelles ni de celui des bases de données orientées objet.

Le langage XQL est étroitement lié à XPath, et sa formulation originale a été basée complètement sur la structure arborescente des documents XML : hiérarchie, ordre et position. Dans les instruction de XQL, une simple chaîne de caractère est interprétée comme nom d'un élément.

Les chemins sont toujours décrits à partir de la racine vers le bas et l'élément retourné est celui à l'extrême droite du chemin. Le contenu d'un élément ou d'un attribut peut être décrit en utilisant l'opérateur(=). L'opérateur de descendance (//) indique tous les niveaux intervenants. L'opérateur de filtrage ([]) filtre l'ensemble de nœuds vers sa gauche basée sur les conditions à l'intérieur des parenthèses. Plusieurs conditions peuvent être combinées en utilisant les opérateurs booléens.

On trouvera ci-dessous un exemple d'une requête XQL qui renvoie tous les restaurants 3 étoiles dont un élément descendant Ville contient pour valeur Paris :

```
//Restaurant? (@catégorie[text()="***"])/Ville [text()= "Paris "]
```

1.4.2.3 NEXI

NEXI [196] est un langage d'interrogation développé dans le cadre de la campagne d'évaluation pour la recherche dans les documents XML INEX (*INitiative for the Evaluation of XML REtrieval*). Il a été conçu pour permettre une représentation simple mais efficace des besoins de l'utilisateur.

La syntaxe de NEXI est semblable à XPATH. On utilise la syntaxe pour désigner le descendant et rajoute une clause "about" pour fournir l'information en question. NEXI peut également supporter des spécifications plus complexes en utilisant les parenthèses ainsi que les opérateurs booléens.

L'exemple de requête ci-dessous renvoie une section (*sec*) qui est descendant d'un élément *article* et qui contient un descendant paragraphe (*p*) qui parle de "computer" :

```
//article //sec[about(./p,Computer)]
```

1.4.2.4 XOR

XOR [73] est un langage de requêtes totalement compatible avec les spécifications du langage NEXI. L'avantage majeur qu'il représente est principalement la possibilité de combiner plusieurs requêtes en une seule. Il permet également une meilleure élaboration des spécifications de chemins et des termes ainsi qu'un ensemble plus large de correspondance des informations recherchées.

L'exemple ci-dessous renvoie un article dont l'auteur est "Einstein" de l'année "1905" qui parle d'"électrodynamics".

```
//article[about(./year,1905)
AND about(./author, Einstein)
AND about(./*, electrodynamics)]
```

1.4.2.5 Autres langages d'interrogation

Il existe de nombreux autres langages d'interrogation. Parmi eux citons XML-QL [121], Quilt [37], XML-GL [36], XIRQL [65], Tequyla-TX [43] ou Tex-Query[9].

Notons simplement que nombreuses sont les spécifications de langages mais rares sont les implémentations.

1.5 Appariement élément-requête

Nous nous intéressons dans cette section aux approches orientées RI où une valeur de similarité par rapport à une requête donnée est calculée pour chaque élément.

Les modèles classiques de RI ont été adaptés tout en tenant compte de la dimension structurelle. Ces modèles permettent une recherche des composants des documents en partant soit des requêtes structurées soit des requêtes composées de simples mots clés.

D'une manière générale, indépendamment des modèles de recherche, nous distinguons deux types d'appariement élément-requête :

1. Un appariement qui s'effectue au niveau des éléments restitués grâce à une propagation de termes qu'ils soient pondérés ou non.

2. Un appariement qui s'effectue au niveau de la plus petite unité d'indexation. Dans ce cas les éléments sont restitués grâce à une propagation de pertinence.

Dans ce qui suit nous présentons différents modèles en fonction de modèle de base de la RI qu'ils étendent. Notons simplement à titre d'illustration que :

- Dans le cadre du modèle vectoriel étendu, les approches présentées dans [133], [130] et [48] utilisent une propagation de termes et que dans [72], [10] et [97], il s'agit de propagation de pertinence.
- Dans le cadre du modèle booléen pondéré, les approches de [194] et [115] utilisent une propagation des termes.
- Les adaptations du modèle probabiliste ([114], [70]), du modèle inférentiel ([146] et [113]) et les modèle de langage ([140], [85], [106] et [180]) fonctionnent également grâce à une propagation des termes.

Enfin, le modèle XFIRM [170, 172] développé dans notre équipe et nous avons étendu pour la réinjection de pertinence utilise la propagation de pertinence. On trouvera une description détaillée du modèle dans le chapitre 4 section 4.2.1.

1.5.1 Modèle vectoriel étendu

Le modèle vectoriel étendu est une extension du modèle vectoriel proposé en RI traditionnelle, dans lequel l'information structurelle est séparée du contenu [133, 130, 131].

Mass et *al.* [133] ont proposé un système de recherche où le score d'un terme t_i dans un document D , $w_D(t_i)$, est exprimé par le produit $tf * idf$ et la pertinence d'un document pour une requête donnée est calculée selon l'équation suivante :

$$\rho(Q, D) = \frac{\sum_{t_i \in Q \cup D} w_Q(t_i) * w_D(t_i)}{\|Q\| * \|D\|} \quad (1.8)$$

Avec $\|Q\|$ et $\|D\|$ sont respectivement les normes des vecteurs de la requête Q et du document D . Pour chaque terme de la requête (t_i, c_i) , (t_i est le terme, c_i est le contexte du terme (i.e. le chemin de l'élément où apparaît le terme t_i)) on calcule son poids dans la requête $w_Q(t_i, c_i)$, son poids dans un contexte similaire dans le document $w_D(t_i, c_k)$, ainsi que la similarité entre les contextes $cr(c_i, c_k)$ où

$$cr(c_i, c_k) = \frac{1 + |c_i|}{1 + |c_k|} \quad (1.9)$$

avec $|c_i|$ est le nombre de balises dans un contexte donné de la requête et $|c_k|$ est le nombre des tags dans un contexte donné d'un document.

La formule précédente est par la suite étendue pour mesurer les similarités entre les fragments XML et le document. La formule mesurant la pertinence d'un document est la suivante :

$$\rho(Q, D) = \frac{\sum_{(t_i, c_i) \in Q} \sum_{(t_i, c_k) \in D} w_Q(t_i, c_i) * w_D(t_i, c_k) * cr(c_i, c_k)}{\|Q\| * \|D\|} \quad (1.10)$$

Les auteurs ont considéré par la suite une autre méthode de mesure de similarité entre document et requête en considérant que les différents contextes c_k sont d'égale similarité avec le contexte de la requête. La formule appliquée est alors la suivante :

$$\rho(Q, D) = \frac{\sum_{(t_i, c_i) \in Q} w_Q(t_i) * w_D(t_i) * w(c_i)}{\|Q\| * \|D\|} \quad (1.11)$$

où $w(c_i) = 1 + |c_i|$ est le poids du contexte c_i .

En 2003, Mass et *al.* [130] proposent de distinguer les composants d'un document, ce qui conduit à considérer six différents index (*article, sec, ss1, ss2, p, ip1 et abs*) et à appliquer sur chacun le processus de recherche tout en adaptant le *tf, idf* au niveau des composants. Pour éviter la redondance des résultats, les auteurs ont proposé un algorithme de classification basé sur la comparaison des scores des nœuds avec ceux de leurs descendants et selon un seuil déjà fixe, on décide d'éliminer l'un des groupes.

De plus, pour surmonter le problème des index de différentes caractéristiques, les auteurs ont proposé la notion de pivot pour avoir un nouveau score au niveau de chaque composant C calculé comme suit :

$$score(Q, C) = DocPivot * S_a + (1 - DocPivot) * S_c \quad (1.12)$$

avec *DocPivot* une constante entre 0 et 1, S_a le score du document auquel appartient le composant et S_c le score propre du Composant. L'algorithme de recherche est alors le suivant :

1. Le système effectue des recherches indépendantes au niveau de chaque sous index i . Le résultat de chaque recherche est l'ensemble R_i .
2. Tous les résultats issus des différentes recherches sont normalisés dans $[0,1]$.
3. Le score final de différents résultats est calculé en fonction du score de l'article. La liste des résultats finale est la combinaison de tous les résultats R_i

Une autre extension du modèle vectoriel est développée par Crouch et *al.* dans [48] en considérant un ensemble de sous vecteurs représentant de différents niveaux de granularité. Dans le cas des requêtes comportant plusieurs contraintes structurelles, le résultat final est l'intersection des résultats issus des recherches

effectuées au niveau des sous-index correspondant aux contraintes structurales. Les requêtes non structurées sont traitées selon Smart [182] en considérant un seul index (*article* ou *paragraphe*).

Dans [72], Geva a proposé un modèle simple donnant très bons résultats pendant les campagnes d'évaluation INEX 2003 et INEX 2004. Ce modèle est basé sur un fichier inverse pour l'indexation d'un document XML. La recherche est réalisée par propagation des scores des éléments feuilles. Ces derniers sont calculés comme suit :

$$L = N^{n-1} \sum_{i=1}^n \frac{t_i}{f_i} \quad (1.13)$$

avec

N un entier faible=5 ;

n : nombre de termes de la requête qui existent dans le noeud n .

t_i : fréquence du i^{eme} terme de la requête dans le noeud n .

f_i : fréquence du i^{eme} terme de la requête dans la collection.

N^{n-1} : augmente le score des éléments ayant plusieurs termes de la requête.

La pertinence R des nœuds internes est calculée par la somme des différents descendants :

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

avec L_i est le score du i^{eme} élément retourné et $D(n)=0.49$ si $n=1$ et 0.99 sinon.

Enfin, on trouvera d'autres approches utilisant le modèle vectoriel étendu dans [10], [35], [48], [105], [129], [131], [133], [130], [176], [191], [204], [97], [76].

1.5.2 Modèle booléen pondéré

Le modèle booléen a été étendu avec un nouvel opérateur binaire non commutatif, appelé "contains" [194]. La première opérande est de type XPath et la seconde est une expression booléenne. Ce modèle permet aux requêtes d'être complètement spécifiées en termes de contenu et d'information structurée, basée sur XPath. La recherche consiste à extraire le titre et le convertir en requête booléenne, les éléments considérés comme pertinents sont par la suite classés selon la somme OkapiBM25 [153].

Dans [115], Larson et *al.* utilisent dans une combinaison de méthodes probabilistes utilisant une régression logistique avec une approche basée sur le modèle booléen, pour évaluer la pertinence des documents et des éléments. La valeur de probabilité de pertinence R d'un composant C (élément) est calculée comme

étant le produit des probabilités de la pertinence de C vis-à-vis la requête Q_{bool} présentée par un modèle booléen et de la pertinence de C vis-à-vis la requête Q_{prob} présentée par un modèle probabiliste. La formule est présentée ci-dessous :

$$p(R|Q, C) = P(R|Q_{bool}, C)P(R|Q_{prob}, C)$$

Cette combinaison permet de restreindre l'ensemble des documents pertinents aux documents ayant une valeur booléenne égale à 1 tout en leur attribuant un rang basé sur un calcul probabiliste. Ces deux types d'extension permettent de surmonter les limites des modèles booléens au niveau du tri des résultats.

1.5.3 Modèle probabiliste

Pour étendre le modèle probabiliste inférentiel aux documents XML, les probabilités doivent tenir compte de l'information structurelle. Une approche est d'utiliser des probabilités conditionnelles sur les chemins des documents, avec par exemple $P(d|t)$ devenant $P(d|p \text{ contains } t)$, où d représente un document ou une partie de document, t est un terme et p est un chemin dans l'arbre structurel de d .

Une méthode d'augmentation basée sur le modèle probabiliste est proposée par Fuhr et al. dans [65, 70]. Cette méthode est basée sur le langage de requêtes XIRQL, et a été implémentée au sein du moteur de recherche HyRex.

Dans cette approche, les noeuds sont considérés comme des unités disjointes (section 1.4.1.1). Tous les noeuds feuilles ne sont cependant pas indexés (car d'une granularité trop fine). Dans ce cas-là les termes sont propagés jusqu'au noeud indexable le plus proche. Afin de préserver des unités disjointes, on ne peut associer à un noeud que des termes non reliés à ses noeuds descendants. Le poids de pertinence des noeuds dans le cas de requêtes orientées contenu est calculé grâce à la *propagation* des poids des termes les plus spécifiques dans l'arbre du document. Les poids sont cependant diminués par multiplication par un facteur, nommé facteur "d'augmentation".

Par exemple, considérons la structure de document de la figure 1.6, contenant un certain nombre de termes pondérés (par leur probabilité d'apparition dans l'élément), et la requête "XML".

Le poids de pertinence de l'élément *section* est calculé comme suit, en utilisant un facteur d'augmentation égal à 0.7 :

$$\begin{aligned} & P([\text{section}, \text{XML}] + P([\text{paragraphe}[2]])) \cdot P([\text{paragraphe}[2], \text{XML}]) \\ & - P([\text{section}, \text{XML}]) \cdot P([\text{paragraphe}[2]]) \cdot P([\text{paragraphe}[2], \text{XML}]) \\ & = 0.5 + 0.7 \cdot 0.8 - 0.5 \cdot 0.7 \cdot 0.8 = 0.68 \end{aligned}$$

Le noeud *paragraphe* (ayant une pertinence de 0.8 à la requête) sera donc mieux

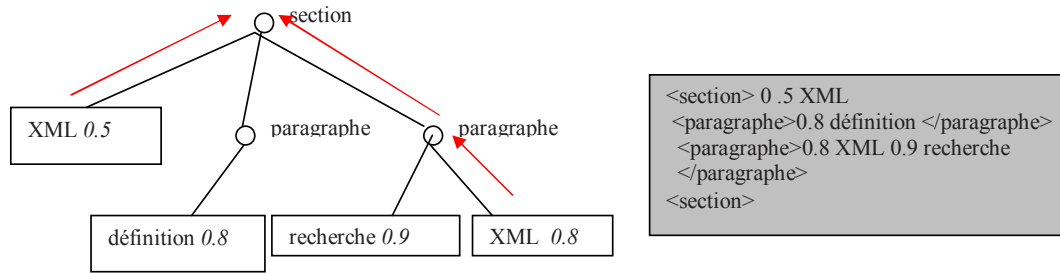


FIG. 1.6 – Modèle d'augmentation [65]

classé que le noeud *section*.

Pour les requêtes orientées contenu et structure, des probabilités d'apparition de chaque terme de la condition de contenu dans les éléments répondant aux conditions de structure sont calculées, et des sommes pondérées de ces probabilités sont ensuite effectués.

On trouvera une autre application du modèle probabiliste dans [114], avec une application de la théorie de Dempster-Shafer [178]. La théorie de l'évidence de Dempster-Shafer est utilisée principalement parce qu'elle possède une règle de combinaison permettant d'effectuer une agrégation du score de pertinence des éléments en respectant la théorie de l'incertain.

1.5.4 Modèle inférentiel

Dans la recherche d'information dans des documents XML, les diagrammes d'inférence ont été adaptés pour exprimer les relations de causalité entre termes et structures.

Parmi les travaux les plus récents, citons celui Piworwarski et *al.* [146]. Les auteurs ont proposé un modèle probabiliste basé sur les réseaux bayésiens où les dépendances de hiérarchisation sont exprimées par des probabilités conditionnelles. La probabilité de pertinence d'un élément e sachant son parent p pour une requête q est $P(e|p, q)$ est la suivante :

$$P(e = a|p = b, q) \simeq \frac{1}{1 + e^{F_{e,a,b(q)}}} \quad (1.14)$$

où,

$F_{e,a,b(q)}$ est la pertinence de l'élément e selon le modèle Okapi.

Une requête q structurée est décomposée en un ensemble de n sous-requêtes élémentaires q_i . Chacune de ces sous-requêtes reflète une entité structurelle et un besoin d'information. Le score final est donné par la formule suivante :

$$RSV(e_i, q) = RSV_{q_1}(e_i, q) * \dots * RSV_{q_n}(e_i, q)$$

De Compos, Fernandez et Huete [113] ont également proposé un modèle de recherche basé sur les réseaux bayésiens où le diagramme d'inférence est basé sur la probabilité conditionnelle. Deux types de diagrammes sont proposés : SID (*Simple Inference Diagram*) et CID (*Context based Inference Diagram*). Un diagramme se compose de deux parties : une partie qualitative et une partie quantitative.

- Le composant qualitatif est la représentation des variables et des influences. Ceci est réalisé par les différents types de nœud : *noeud de chance*, *noeud de décision* et *noeud d'utilité*. Dans ce modèle les arcs pointent vers les nœuds de chance et d'utilité seulement. Dans SID, les nœuds de chance et de décision sont liés aux nœuds de l'utilité qui seront additionnés à la fin. Dans CID, on rajoute par rapport à SID des arcs provenant des nœuds de chance vers les nœuds d'utilité qui leurs sont au dessus par rapport à l'arborescence.
- Le composant quantitatif est la probabilité des nœuds de chance et des nœuds d'utilité variant entre 0 et 1.

1.5.5 Modèles de langage

Un modèle de langage en recherche d'information dans des documents XML est proposé dans [139]. L'idée est de combiner différents modèles de langage en appliquant l'interpolation linéaire.

Sigurbjornsson et *al.* ont proposé dans [180] un modèle combinant des modèles de langage de l'élément, du document et de la collection. Pour estimer les modèles de langage, les auteurs ont utilisé deux types d'index : un index pour les éléments du document XML qui assure la même fonction qu'un fichier inverse en RI classique et un autre (index article) pour tout le document utilisé pour des calculs statistiques. L'arbre XML est indexé en se basant sur le post et le pre-ordre des nœuds.

Pour chaque élément e , on estime le modèle de langage (score) pour une requête donnée q selon la formule suivante :

$$P(e|q) \propto P(e).P(q|e) \quad (1.15)$$

Les auteurs considèrent l'indépendance entre les termes de la requête et la formule précédente devient alors :

$$P(e|q) \propto P(e). \prod_{i=1}^k P(t_i|e) \quad (1.16)$$

avec t_i terme de la requête.

La probabilité de $P(t_i|e)$ est une interpolation linéaire des trois modèles de langage (*élément*, *article* et *collection*) :

$$P(t_i|e) = \lambda_e \cdot P_{mle}(t_i|e) + \lambda_d \cdot P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{mle}(t_i) \quad (1.17)$$

avec $P_{mle}(t_i|e)$ est la probabilité de t_i dans le modèle de langage de l'élément estimée par les statistiques à partir de l'index des éléments, $P_{mle}(t_i|d)$ est la probabilité de t_i dans le modèle de langage du document estimée par les statistiques à partir de l'index *article* et $P_{mle}(t_i)$ est la probabilité de t_i dans le modèle de langage de la collection. λ_e et λ_d sont des constantes.

On trouvera d'autres applications des modèles de langages à la RI structurée dans [147], [1], [85], [106].

1.5.6 Autres modèles de recherche

D'autres modèles ont été proposés pour la recherche dans des documents XML. Ils ont été conçu de manière à calculer la pertinence d'un élément en tenant compte à la fois de la pertinence du contenu et celle de la structure des éléments à renvoyer.

Dans [192], Trotman a proposé d'attribuer des degrés d'importance pour chaque structure du document et de remplacer le tf par la fréquence du terme en tenant compte du poids de la structure.

Dans le modèle vectoriel une telle approche se traduit dans le calcul de fréquence d'un terme en remplaçant la formule

$$tf_{id} = \sum_{p=1}^n tf_{ipd}$$

par la formule

$$tf'_{id} = \sum_{p=1}^n (C_p * tf_{ipd})$$

où tf_{ipd} est le nombre d'occurrences du terme t à la position p du document d . C_p est le poids de chaque structure du document qui doit être fixé. Cette méthode d'indexation et de recherche des données structurées permet de donner un poids aux structures. Un algorithme génétique est employé pour l'apprentissage des poids.

Une approche d'agrégation est appliquée dans [110] pour déterminer le poids d'un composant tout en respectant à la fois la représentation d'un document et ses composants. Elle suit la structure hiérarchique et la structure linéaire des documents.

L'agrégation est aussi appliquée au niveau de l'indexation et du calcul d'incertitude de la représentation des nœuds. Le résultat est une liste de composants de documents représentant des meilleurs points d'entrée dans les documents.

1.5.7 Modèles spécifiques aux collections de documents hétérogènes

L'hétérogénéité représente un des principaux challenges de la RI structurée. Plusieurs modèles ont été proposés en particulier dans le cadre de la tâche hétérogène d'INEX. La majorité des solutions proposées s'orientent vers la classification de documents [120], [116], [134]. La recherche se fait alors au niveau des classes de documents.

- Denoyer et *al.* [52] ont conçu un format intermédiaire qui permet de classer les documents en suivant un calcul basé sur la probabilité conditionnelle.
- Denoyer et Gallinari [50] ont également modélisé le problème sous forme de réseaux bayésiens. Chaque noeud du réseau comporte un libellé et des informations contextuelles. Deux sortes de variables sont envisagées :
 1. une variable structurelle s_d^i (d : document) qui dépend de ses ascendants.
 2. une variable contextuelle t_d^i qui ne dépend que de ses variables structurelles.

La probabilité de jointure d'un document d à un modèle C est calculée comme suit :

$$P(d, C) = P(c) \prod_{i=1}^{|d|} P(s_d^i / pa(s_d^i, C) P(t_d^i / s_d^i, C)$$

avec t_d^i est une séquence de mots et $pa(s)$ présente le parent d'un noeud. Ce modèle génératif permet de considérer des documents hétérogènes (texte + image), où l'image est considérée comme un ensemble de pixels. Il est par la suite transformé en classifieur discriminant en utilisant la méthode Fisher Kernel.

- Le problème de classification a été également traité par Lee et *al.* [104]. Les auteurs ont proposé un algorithme permettant un matching entre

deux documents grâce une séquence d'opérations de transformations.

- Lian et Cheung [124] ont aussi proposé un algorithme de classification. L'algorithme (S_GRACE) a été proposé pour classifier les documents en se basant sur le paramètre distance et la notion de sous-graphe qui sont codés par des chaînes de bits. La distance entre deux documents C_1 et C_2 est calculée comme suit :

$$dist(C_1, C_2) = 1 - \frac{|sg(C_1) \cap sg(C_2)|}{\max\{|sg(C_1)|, |sg(C_2)|\}}$$

avec $sg(C_i)$ est l'ensemble de graphes et de sous-graphes structurels de C_i ($i=1,2$), $|sg(C_i)|$ est le nombre de d'arcs dans $sg(C_i)$ et $|sg(C_1) \cap sg(C_2)|$ est le nombre d'arcs communs de $sg(C_1)$ et $sg(C_2)$.

1.6 Évaluation de la RIS : La campagne INEX

INEX (*INitiative for the Evaluation of XML Retrieval*) est la seule campagne d'évaluation des différents SRI pour la recherche d'information sur les documents XML. Elle est mise en place chaque année depuis 2002. Elle offre un forum international non seulement pour permettre aux différentes organisations participantes d'évaluer et de comparer leurs résultats, mais aussi pour discuter les différentes problématiques qui se présentent.

La collection de test consiste en un ensemble de documents XML, requêtes, tâches de recherche et jugements de pertinence.

1.6.1 Collection

Les collections de test préparées dans le cadre d'INEX ne cessent d'évoluer dans le but d'améliorer la qualité de l'évaluation. De 2002 à 2004, la collection de documents était composée d'articles scientifiques provenant de la *IEEE Computer Society*, balisés au format XML. La collection, d'environ 500 Mo, contenait plus de 12000 articles, publiés de 1995 à 2002, et provenant de 18 magazines ou revues différents. En 2005, cette collection a été étendue pour comporter environ 17000 articles publiés entre 1995 et 2004 provenant de 21 magazines ou revues différents pour une taille d'environ 750 Mo. En 2006, la collection est composée de 659388 documents en anglais extraits de l'encyclopédie en ligne Wikipedia [51] avec une taille totale d'environ 5 GigaOctets. Dans la collection IEEE, un article est composé d'environ 1500 éléments et la collection contient au total 8 millions de nœuds et 180 balises différentes.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE xmlarticle SYSTEM "../..dtd/xmlarticle.dtd">
<article>
<fno>A1003</fno>
<doi>10.1041/A1003s-1995</doi>
<fm><hdr><hdr1><ti>IEEE Annals of the History of Computing</ti>
<cr><issn>1058-6180</issn>/95/$4.00 <cci><onm>&copy; 1995 IEEE</onm></cci></cr></hdr1>
<hdr2><obi><volno>Vol. 17</volno>, <issno>No. 1</issno></obi>
<pdt><mo>Spring</mo><yr>1995</yr></pdt>
<pp>pp. 3-3</pp></hdr2></hdr>
<tig>
<atl>About this Issue</atl><pn>pp. 3-3</pn></tig>
<au sequence="first"><fnm>J.A.N.</fnm><snm>Lee</snm><role>Editor&hyphen;in&hyphen;Chief</role></au>
</fm>
<bdy>
<p>The first issue of our 17th volume is as diverse in topics as any nontheme issue that we have tried to present over
the past many years. However, it still represents the work of the English&hyphen;speaking world of the North Atlantic
rather than a broader picture of computing in the whole world. The Editorial Board and the article editors of the
<it>Annals</it>
are doing their best to bring the history of the whole world of computing to our readers, but it does require authors in
other countries to offer their manuscripts for our consideration. Please take this as an open invitation to authors in
other parts of the world to submit papers to the <it>Annals</it>
for review and help us to follow the lead of our parent organization in being the &ldquo;The World&rsquo;s Computer
Society.&rdquo;</p>
....
</p>
</bdy>
</article>

```

FIG. 1.7 – Exemple d'un article de la collection IEEE au format XML

Dans la collection Wikipédia, un article est composé d'environ 70 éléments, la collection contient au total plus que 460 millions de nœuds et 5000 balises différentes.

Les figures 1.7 et 1.8 présentent deux exemples de documents des collections 2005 et 2006.

1.6.2 Requêtes

Les requêtes (ou Topics) sont créées par les différents participants et doivent être représentatives des demandes de l'utilisateur moyen. Les topics se divisent en deux catégories principales :

- Les CO (*Content Only*) : ce sont les requêtes composées de simples mots clés. Les mots clés de la requête peuvent être éventuellement groupés sous forme d'expressions et précédés par les opérateurs '+' (signifiant que le terme est obligatoire) ou '-' (signifiant que le terme ne doit pas apparaître dans les éléments renvoyés à l'utilisateur).
- Les CAS (*Content And Structure*) : ces requêtes contiennent des contraintes sur la structure des documents.

Pour chaque Topic, différents champs permettent d'explicitier le besoin de l'utilisateur : le champ *Title* donne la définition simplifiée de la requête, le champ *Keywords* contient un ensemble de mots clés qui ont permis l'exploration du

```

<?xml version="1.0" encoding="UTF-8" ?>
- <article>
  <name id="250237">Effective results in number theory</name>
  <conversionwarning>0</conversionwarning>
- <body>
  <template name="Cleanup date">November 2005</template>
- <p>
  For historical reasons and in order to have application to the solution of
  <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="9109.xml">Diophantine
  equation</collectionlink>
  s, results in
  <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="21527.xml">number
  theory</collectionlink>
  have been scrutinised more than in other branches of
  <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
  xlink:href="18831.xml">mathematics</collectionlink>
  to see if their content is
  <emph3>effectively computable</emph3>
  . This for example brings into question any use of
  <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="44578.xml">big O
  notation</collectionlink>
  and its implied constants: are assertions pure
  <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="251177.xml">existence
  theorem</collectionlink>
  s for such constants, or can one recover a version in which 1000 (say) takes the place of the implied
  constant?
  </p>
- <p>
- .....
  </p>
  <languelink lang="fr">Résultats effectifs en théorie des nombres</languelink>
  </body>
  </article>

```

FIG. 1.8 – Exemple d'un article de la collection Wikipédia au format XML

```

<inex_topic topic_id="98" query_type="CO" ct_no="26">
  <title>
    "Information Exchange", +"XML", "Information Integration"
  </title>
  <description>
    How to use XML to solve the information exchange
    (information integration) problem, especially in heterogeneous data
    sources?
  </description>
  <narrative>
    Relevant documents/components must talk about techniques of
    using XML to solve information exchange (information integration)
    among heterogeneous data sources where the structures of participating
    data sources are different although they might use the same ontologies
    about the same content.
  </narrative>
  <keywords>
    information exchange, XML, information integration,
    heterogeneous data sources
  </keywords>
</inex_topic>

```

FIG. 1.9 – Exemple de requête CO de la collection 2005

```

<inex_topic query_type="CO+S">
<title> Tolkien languages "lord of the rings" </title>
<castitle> //article[about(., Tolkien) or about(., "lord of the rings")]//sec[about(.,
Tolkien languages)]</castitle>
<description> Find information about Tolkien languages from the Lord of the
Rings.</description>
<narrative> The "Lord of the Rings" movie trilogy fascinate me. I have learned from
other fans that the languages spoken by e.g., elves and dwarfs in the screen version
are not just the usual effects. Apparently, these languages were invented by Tolkien
himself and are central to his work with the original books.
For my own personal interest, I would like to learn more background about Tolkien's
artificial languages, and how they have affected the world portrayed in the Lord of
the Rings universe. Later I may want to add a section on the influence languages to my
Lord of the Rings fan web page. As Tolkien's languages seem to be a rather specialized
topic, I expect to find relevant information as elements in larger documents that deal
with Tolkien or Lord of the Rings, e.g., as sections in documents about Tolkien or the
Lord of the Rings (although I would be pleasantly surprised to see whole documents on
the topic of Tolkien's languages).
To be relevant an element should discuss Tolkien's artificial languages and their
influence on the Lord of the Rings books or movies. Information on the languages alone
without explicit discussion of their impact on the books or movies is not relevant;
nor is general information on Tolkien or the Lord of the Rings.</narrative>
<ontopic_keywords> "High Elvish" ; Quenya ; Sindarin</ontopic_keywords>
<offtopic_keywords> inspired, film</offtopic_keywords>
</inex_topic>

```

FIG. 1.10 – Exemple de requête de la collection 2006

corpus avant la reformulation définitive de la requête, et les champs *Description* et *Narrative*, explicités en langage naturel, indiquent les intentions de l'auteur [67]. La formulation des requêtes est étroitement liée à la tâche de recherche associée.

En 2006, ces types de requêtes ont été regroupés dans le seul type CO+S en rajoutant un nouveau champ *castitle*, donnant la forme structurée de la requête.

Les deux figures 1.9 et 1.10 présentent respectivement un exemple de requête de type CO et un exemple de requête de type CAS.

1.6.3 La tâche ad-hoc

INEX propose plusieurs tâches d'évaluation dont la principale est la tâche de recherche ad-hoc. Elle est considérée comme une simulation de l'utilisation d'une bibliothèque, où un ensemble statique de documents est interrogé avec des besoins utilisateurs. La tâche ad-hoc est à son tour composée de sous-tâches divisées selon soit :

- le **type de requêtes** : les requêtes peuvent à la fois contenir des conditions structurelles et d'autres portant sur le contenu. En réponse à la requête, des éléments peuvent être retrouvés à partir de la collection. La tâche ad-hoc a été divisée en 3 sous tâches en 2004 (CO, SCAS et VCAS), en 2005 elle est divisée en 5 sous-tâches (CO, SSCAS, VSCAS, SVCAS

et VVCAS) et en 2006, les tâches se sont limitées à CO et CO+S.

- la **stratégie de recherche**, c'est à dire le critère sur lequel est jugée la performance d'un système. On distingue trois sous-tâches : "*Fetch and Browse*", "*Thorough*" et "*Focused*". En 2006, une nouvelle tâche appelée "*Best in Context*" a été définie.

1.6.3.1 Tâche CO

La tâche CO (*Content Only Task*) a pour but de répondre avec des éléments/documents XML à des requêtes utilisateurs CO. Aucune indication de structure dans la requête ne peut aider les SRI à déterminer la granularité de l'information à renvoyer.

1.6.3.2 Tâche CAS

On distingue plusieurs sous-tâches :

- La tâche SCAS (*Strict Content And Structure task*) consiste à répondre avec des éléments/documents XML aux topics CAS de manière stricte, c'est à dire respectant toutes les conditions sur la structure et le contenu énoncés dans la requête. Le champ *Title* de la requête SCAS est basé sur une syntaxe XPath.
- La tâche VCAS (*Vague Content And Structure Task*) utilise elle aussi des requêtes CAS, mais pour lesquelles les participants peuvent répondre de manière vague, c'est à dire avec des éléments/documents qui satisfont globalement les requêtes. Le champ *Title* des requêtes VCAS est basé sur le langage NEXI [196].
- Dans la tâche VVCAS, les éléments supports³ et les éléments recherchés spécifiés dans la requête sont interprétés de manière vague. Les jugements de pertinence sont fait selon le champ *Narrative* de la requête.
- Dans la tâche SVCAS, le type d'élément recherché spécifié dans la requête doit être respecté dans l'ensemble des éléments pertinents.
- Dans la tâche VSCAS, les éléments pertinents doivent respecter les éléments supports spécifiés dans la requête.
- Dans la tâche SSCAS, les éléments pertinents doivent satisfaire strictement les éléments supports ainsi que l'élément recherché spécifié dans la requête.

³ Les éléments supports sont les éléments qui décrivent la structure de l'élément que l'utilisateur désire retrouver

1.6.3.3 Stratégies de recherche

Parmi les stratégies de recherche, on distingue :

- La tâche **Thorough** dans laquelle on suppose qu'un utilisateur préfère retrouver tous les éléments fortement pertinents.
- La tâche **Focused** dans laquelle on suppose qu'un utilisateur préfère ne pas avoir d'éléments imbriqués dans ses réponses.
- La tâche **Fetch and Browse** appelée aussi **All in Context**, qui consiste à classer les résultats par article ou document. L'évaluation concerne alors d'une part les documents et d'autre part le classement des éléments dans un document donné.
- La tâche **Best in Context** qui permet d'évaluer les meilleurs points d'entrée dans un article donné.

1.6.4 Autres tâches

La campagne d'évaluation INEX ne cesse d'intégrer des tâches autres que la tâche Ad-hoc. Dans ce qui suit, nous présentons les différentes tâches proposées au fil des années.

1.6.4.1 Traitement automatique du langage naturel

Dans cette tâche, les utilisateurs formulent leurs requêtes en langage naturel, sans avoir besoin d'apprendre un langage complexe. Les systèmes ne doivent exploiter que le champ description spécifié dans la requête [69].

1.6.4.2 Tâche Reformulation par réinjection de pertinence (Relevance Feedback)

La tâche *Relevance Feedback* a pour but de reformuler la requête initiale de l'utilisateur en se basant sur des jugements de pertinence afin d'améliorer des performances des systèmes de recherche [4]. Nous détaillons cette tâche à laquelle nous participons dans le chapitre suivant.

1.6.4.3 Tâche Hétérogène

La collection d'évaluation utilisée dans les différentes tâches d'INEX est composée de documents homogènes ayant la même DTD. Dans la réalité, les

documents proviennent souvent de différentes collections ne possédant pas la même DTD. Notamment avec l'apparition et l'utilisation des systèmes distribués, la tâche hétérogène s'avère un véritable challenge qui pose un certain nombre de défis :

- dans le cas d'une recherche orientée contenu, les approches utilisées utilise une DTD pour retourner des éléments formant des réponses raisonnables. Dans des collections hétérogènes, des nouvelles approches doivent être développées indépendamment des DTDs.
- dans le cas des requêtes de type CAS, s'ajoute le problème de faire correspondre des conditions structurelles appartenant à de différentes DTDs.

1.6.4.4 Fouille de données (Data mining)

Le but de la recherche d'information dans des documents XML est de renvoyer les éléments (partie de document) répondant aux besoins de l'utilisateur. Cependant avec la masse croissante d'informations disponibles, un nouveau challenge est défini qui permet de classifier et de regrouper les informations afin de permettre un accès direct aux besoins de chaque utilisateur.

1.6.4.5 Tâche interactive

Cette tâche définie en 2005 a pour but d'étudier le comportement utilisateur lors d'une recherche dans les documents XML et de développer des systèmes qui tiennent compte de l'environnement de l'utilisateur [69].

1.6.4.6 Tâche multimedia

Dans cette tâche, on s'intéresse à développer des systèmes de recherche non seulement dans les documents textes mais aussi contenant des images et des vidéos.

1.6.5 Jugements de pertinence

L'évaluation de pertinence des SRI passe par une première phase de validation des documents renvoyés par les SRI. Chaque élément/document est jugé à la main par les participants pour chaque requête en utilisant le système de jugement en ligne [144]. En 2002, une première échelle de pertinence à deux dimensions a été proposée, basée sur le degré de pertinence et la couverture des

éléments. Depuis 2003, ces deux dimensions ont été remplacées par la spécificité et l'exhaustivité. Pour chacune une échelle de 4 niveaux a été définie : pas exhaustif (resp. pas spécifique), marginalement exhaustif (resp. marginalement spécifique), assez exhaustif (resp. assez spécifique) et très exhaustif (resp. très spécifique).

En 2005 et 2006, l'exhaustivité est mesurée selon une échelle à 4 niveaux :

exhaustivité { e=2 exhaustivité élevée
 e=1 exhaustivité moyenne
 e=0 pas d'exhaustivité
 e=? élément trop petit

La spécificité quant à elle est mesurée dans un intervalle continu $[0,1]$ où $s=1$ représente un élément totalement spécifique.

Les mesures d'évaluation utilisées durant la campagne 2005 sont basées sur les mesures XCG et ep/gr [109]. Ces mesures sont calculées en tenant compte des 2 dimensions de pertinence (exhaustivité et spécificité) agrégées en une seule valeur. Deux types de fonction d'agrégation sont utilisées :

- une agrégation "stricte" pour évaluer si un SRI est capable de retrouver des éléments très spécifiques et très exhaustifs

$$f_{strict}(e, s) = \begin{cases} 1 & \text{si } e = 2 \text{ et } s = 1 \\ 0 & \text{sinon} \end{cases} \quad (1.18)$$

- une agrégation "généralisée" pour évaluer les éléments selon leur degré de pertinence

$$f_{generalisee}(e, s) = e * s \quad (1.19)$$

L'utilisation d'une échelle à deux dimensions est motivée par le besoin de refléter la pertinence relative d'un élément par rapport à ses descendants. Par exemple, un élément peut être plus exhaustif que chacun de ses descendants pris séparément puisqu'il couvre l'union des aspects discutés dans chacun. De la même manière, des éléments peuvent être plus spécifiques que leurs parents, car ces derniers couvrent plus de sujets, y compris des sujets non pertinents.

1.6.6 Mesures d'évaluation

Jusqu'à 2004, l'évaluation de pertinence des différents systèmes proposés par les participants utilise des méthodes basées sur les mesures de rappel et précision en tenant compte de la structure des documents XML et de la possible imbrication des résultats. Depuis INEX 2005, d'autres mesures ont été définies

pour permettre une évaluation plus appropriée des performances des systèmes de recherche en RI structurée [109] : le gain cumulé (xCG) et l'effort précision (ep).

La mesure xCG cumule les scores de pertinences des éléments de la liste des résultats. Etant donnée une liste triée d'éléments xCG dans laquelle les identifiants des éléments sont remplacés par leur score de pertinence, le gain cumulé au rang i , noté $xCG[i]$, est calculé comme la somme des pertinences jusqu'à ce rang :

$$xCG[i] = \sum_{j=1}^i xG[j] \quad (1.20)$$

Par exemple, soit $xG_i = \langle 2, 1, 0, 1, 0, 0 \rangle$ un vecteur de gain jusqu'au rang i . Le vecteur de gain cumulé sera $\langle 2, 3, 3, 4, 4, 4 \rangle$.

Pour chaque requête, on calcule un vecteur de gain idéal xCI à partir de la base de rappel, en cumulant les scores de pertinences des éléments triés par ordre décroissant. Le xCG peut alors être comparé au gain idéal. Le xCG normalisé ($nxCG$) est obtenu par :

$$nxCG[i] = \frac{xCG[i]}{xCI[i]} \quad (1.21)$$

Pour un rang donné i , le gain cumulé $nxCG[i]$ reflète le gain relatif de l'utilisateur accumulé jusqu'à ce rang, comparé à ce qu'il aurait dû atteindre si le système avait produit une liste triée optimale.

Par analogie au gain cumulé, on définit l'effort-précision ($ep(r)$)

$$ep(r) = \frac{e_{ideal}}{e_{run}} \quad (1.22)$$

où e_{ideal} est le rang pour lequel le gain cumulé est atteint par la courbe idéale. e_{run} est le rang pour lequel le gain cumulé est atteint par le système.

La valeur 1 correspond à une performance idéale, pour laquelle l'utilisateur effectue un minimum d'effort pour atteindre un niveau de gain donné.

L'effort-précision est calculé à des points de gain-rappel arbitraires, où le gain-rappel gr est la valeur du gain cumulé divisé par la valeur totale atteignable du gain cumulé :

$$gr[i] = \frac{xCG[i]}{xCI[n]} \quad (1.23)$$

avec n le nombre total de document pertinents.

L'effort-précision à une valeur donnée de gain-rappel mesure l'effort d'un utilisateur pour atteindre un gain relatif au gain total qu'il peut obtenir.

La moyenne non interpolée MAep (*Mean Average Effort Precision*) d'effort-précision est utilisée pour moyenniser les valeurs d'effort-précision pour chaque rang auquel un élément pertinent est renvoyé.

D'autres mesures ont été proposées mais ne sont pas utilisés pour l'évaluation

officielle d'INEX. Citons par exemple :

- **EPRUM** (*The Expected Precision Recall with User Model*) : c'est une extension de la mesure Rappel-Précision. Cette mesure définie par Piwowarski [145] est utilisée en général pour mesurer les tâches *Focused* et *Fetch and Browse*. En effet elle permet de mettre en évidence le comportement de l'utilisateur en définissant un modèle probabiliste d'utilisateur. La précision est définie comme le ratio du nombre minimal des rangs consultés par l'utilisateur dans une liste retournée par un système idéal et de celui d'un système évalué.
- **BEDP** utilisée pour l'évaluation de la tâche *Best In Context*. C'est la somme des scores de tous éléments excédant le point X divisée par le nombre de meilleurs points d'entrée.
- Les F_{score} (combinaison des taux de rappel et de précision) utilisés pour évaluer la tâche *All In Context* au niveau d'un document.
- La **précision généralisée** (resp. **rappel généralisé**) : ces mesures sont elles aussi utilisées dans la tâche *All In Context*. C'est la somme de F-scores (resp. nombre d'articles ayant une pertinence) qui précèdent un article divisé par le rang de cet article (resp. le nombre des articles ayant une pertinence)

1.7 Conclusion

Les documents semi-structurés, en permettant le balisage des contenus des documents, réactualisent la problématique de recherche d'information classique, et permettent ainsi de traiter l'information avec une granularité plus fine. Le but des SRI traitant des documents semi-structurés est alors d'identifier des parties des documents les plus pertinentes à une requête donnée.

Nous avons ainsi présenté les principales approches d'indexation et d'appariement développées en RIS. Nous avons également détaillé les nouveaux concepts d'évaluation des systèmes de recherche en RIS.

Nous avons de plus indiqué que la reformulation de requêtes est une phase importante en RI classique permettant l'amélioration des performances des SRI. Parmi les techniques les plus efficaces citons la reformulation de requêtes par réinjection de pertinence. Avant de détailler notre contribution en réinjection de pertinence en RI structurée, nous allons présenter dans le chapitre suivant les principales approches développées en reformulation de requêtes, et plus particulièrement en réinjection de pertinence.

Chapitre 2

Reformulation de Requêtes

2.1 Introduction

Les performances d'un SRI, mesurées en général par la double mesure rappel-précision, dépendent d'une part de l'efficacité du modèle de recherche mis en œuvre pour l'appariement des requêtes documents, et d'autre part des requêtes formulées par l'utilisateur. En effet, l'utilisateur formule son besoin en information par une requête composée de ses propres mots clés et le choix de chaque terme a une influence directe sur l'ensemble des documents restitués par le système. Le plus souvent, l'utilisateur formule ses requêtes avec des termes qui lui sont propres, mais qui ne correspondent pas forcément à ceux utilisés pour indexer les documents pertinents des collections interrogées. Pour sélectionner le maximum de documents pertinents tout en limitant le bruit¹, il faudrait alors que l'utilisateur puisse choisir les termes utilisés comme index. Cette tâche s'avère difficile dans la mesure où il est impossible de connaître le langage d'indexation utilisé et où le nombre de termes indexés est généralement très grand. De plus, l'indexation et en particulier son exhaustivité, a également une incidence directe sur la qualité des réponses du système de recherche. De ce fait, retrouver les informations pertinentes en utilisant seulement la requête initiale de l'utilisateur est une opération quasi-impossible.

De nombreux travaux visent à concevoir des SRI capables de s'adapter aux besoins de l'utilisateur. La reformulation de la requête est sans doute la piste la plus investie dans ce contexte.

La reformulation de requêtes, comme nous l'avons signalé dans le chapitre précédent, est un processus ayant pour objectif de générer une nouvelle requête

¹Le bruit est défini par les documents non pertinents retrouvés par le système de recherche.

plus adéquate que celle initialement formulée par l'utilisateur. Elle représente une forme de personnalisation à court terme. Cette reformulation permet de coordonner le langage de recherche, utilisé par l'utilisateur dans sa requête et le langage d'indexation. Par conséquent, elle limite le bruit et le silence² dus à un mauvais choix des termes d'indexation dans l'expression de la requête d'une part, et les lacunes du processus d'indexation d'autre part.

L'apparition des documents semi-structurés a, comme nous l'avons signalé dans le chapitre précédent, apporté la dimension structurelle qui remet à plat la reformulation classique. En effet les techniques de reformulation en RIS doivent tenir compte de la co-habitation des deux sources d'évidences, le contenu et la structure qui apparaissent aussi bien dans les collections de documents que dans les requêtes de l'utilisateur.

Dans ce chapitre, nous présentons quelques techniques utilisées pour l'amélioration des performances des SRI dans la section 2.2. Dans la section 2.3, nous présentons le processus général de la réinjection de pertinence ainsi que ses différentes applications. Nous présentons ensuite les nouveaux enjeux de la réinjection de pertinence en Recherche d'Information structurée ainsi que les modèles proposés dans la littérature dans la section 2.4. Nous finissons par présenter le mécanisme d'évaluation de la réinjection de pertinence dans la section 2.5.

2.2 Techniques pour l'amélioration des performances des systèmes de recherche

Comme nous l'avons mentionné dans l'introduction, la requête initiale seule est souvent insuffisante pour permettre la sélection de document répondant au besoin de l'utilisateur. De ce fait, plusieurs techniques ont été proposées pour améliorer les performances des SRI. Ces méthodes apportent des solutions aux deux principales questions :

1. Comment peut-on retrouver plus de documents pertinents vis à vis d'une requête donnée ?
2. Comment peut-on mieux exprimer la requête de l'utilisateur de manière à mieux répondre à son besoin ?

Les principales techniques d'amélioration des systèmes de recherche se divisent en deux principales voies (voir figure 2.1) :

² Le silence est défini par les documents pertinents non retrouvés.

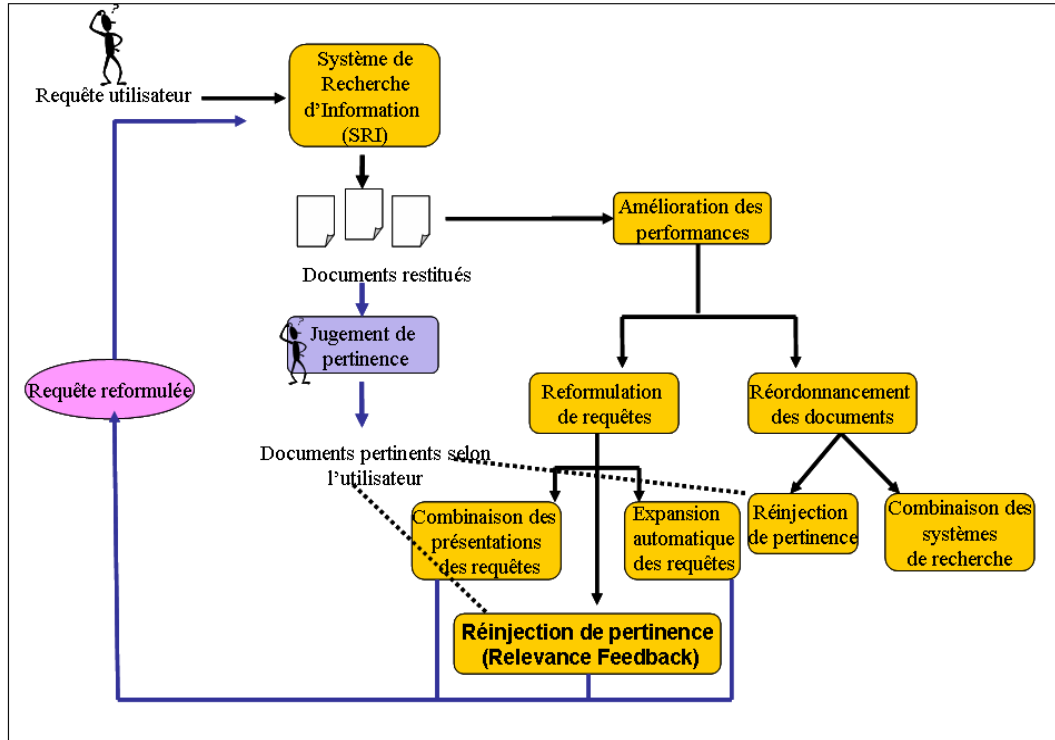


FIG. 2.1 – Le Processus général de l'amélioration de la recherche

- La première voie propose de réordonner les documents sans modifier la requête, soit en utilisant les jugements de pertinence des résultats déjà restitués [141] et en calculant de nouveaux scores pour les documents, soit en fusionnant les résultats de différents systèmes de recherche. Dans le premier cas, on parle de réinjection de pertinence pour le réordonnement et dans le second, on parle d'algorithmes multiples de recherche souvent utilisés dans les moteurs de recherche sur le web [181].
- La seconde voie propose de reformuler la requête initiale en y ajoutant de nouveaux termes. La reformulation peut se faire par expansion automatique de la requête, par combinaison de différentes présentations de la requête ou par réinjection de pertinence.

Nous présentons dans ce qui suit les principales techniques de reformulation de requêtes : la réinjection de pertinence sera détaillée dans la section 2.3.

2.2.1 Expansion et combinaison de requêtes

L'expansion directe de la requête consiste à rajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Plus précisément,

- au niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. On peut alors utiliser des ontologies linguistiques (citons par exemple *Wordnet* [137]).

On peut également ajouter à la requête des variantes morphologiques des termes employés par l'utilisateur. Le but de ce mécanisme est d'assurer la restitution des documents indexés par des variantes des termes composant la requête.

Les associations établies manuellement traduisent généralement des relations de synonymie et de hiérarchie. Les thésaurus construits manuellement sont un moyen efficace pour l'expansion de requête. Cependant, leur construction et la maintenance des informations sémantiques qu'ils contiennent sont coûteuses en temps et nécessitent le recours à des experts des domaines considérés. Pour cette raison, ils restent peu utilisés par les SRI.

- En ce qui concerne la seconde catégorie de ressources, elles sont construites en s'appuyant sur une analyse statistique des collections. Il s'agit de chercher des associations de termes afin d'ajouter des termes voisins à la requête. Il existe aussi d'autres méthodes entièrement automatiques telles que le calcul des liens contextuels entre termes [42] et la classification automatique de documents [39].

Les associations créées automatiquement sont généralement basées sur la cooccurrence des termes dans les documents. Les liens inter-termes renforcent la notion de pertinence des documents par rapport aux requêtes.

2.2.2 Combinaison de requêtes

Plusieurs approches de RI [181] utilisent une seule représentation de requête comparée à plusieurs représentations de document (algorithmes multiples de recherche). Il a été montré dans [118] qu'une recherche plus efficace peut être atteinte en exploitant des représentations multiples de requêtes ou d'algorithmes de recherche différents ou encore en utilisant différentes techniques de réinjection.

Une combinaison des représentations de requêtes peut augmenter le rappel d'une requête, tandis que la combinaison des algorithmes de recherche peut augmenter la précision. La base théorique de la combinaison des évidences a été présentée par Ingwersen [100, 101]. Il a en particulier montré que des représentations multiples du même objet, par exemple une requête, permettent une meilleure perception de l'objet qu'une seule bonne représentation. Cependant, il est important que chacune des sources d'évidences utilisées fournisse non seulement un point de vue différent sur l'objet, mais que ces points de vue aient différentes bases cognitives. Les représentations multiples d'une requête peuvent fournir différentes interprétations du besoin en information.

Une des approches de combinaison de multiples représentations de requêtes est par exemple proposée dans [16]. Elle consiste à calculer les scores des documents directement depuis la fonction d'appariement document-requête en utilisant le même système de recherche mais différentes versions de la requête. Ensuite, les résultats obtenus par chacune des versions sont combinés pour avoir une seule liste finale. Ces versions sont issues soit des expressions d'une même requête par des chercheurs différents, soit des présentations d'une même requête dans des langages différents.

Tamine et *al.* proposent dans [190] une technique de recherche d'information basée sur les algorithmes génétiques, plus précisément, elle propose d'utiliser une population de requêtes qui évolue à chaque étape de la recherche et tente de récupérer le maximum de documents pertinents.

2.3 Réinjection de pertinence

2.3.1 Motivation

Plusieurs travaux en RI ont été développés en considérant que les besoins de l'utilisateur sont fixes au cours d'une recherche d'information. Si cela peut être vrai dans certains cas, des études [112, 57, 187] prouvent que les besoins d'information devraient être considérés comme des entités passagères et évolutives plutôt que comme des demandes fixes. L'intervention de l'utilisateur au niveau de jugement de pertinence joue alors un rôle intéressant pour la reformulation de requêtes afin d'affiner la recherche.

2.3.2 Processus général de *RF*

Le processus de réinjection de pertinence, comme schématisé sur la figure 2.2, comporte principalement trois étapes : l'échantillonnage, l'extraction des évidences et la réécriture de la requête.

- L'échantillonnage : cette étape permet de construire un échantillon de documents à partir des éléments jugés par l'utilisateur. Cet échantillon est caractérisé par le nombre d'éléments jugés et le nombre d'éléments jugés pertinents.
- L'extraction des évidences est l'étape la plus importante, elle consiste en général à extraire les termes pertinents qui serviront à l'enrichissement de la requête initiale. Plusieurs approches ont été développées, la plus reconnue est celle de Rocchio [158] adaptée au modèle vectoriel.

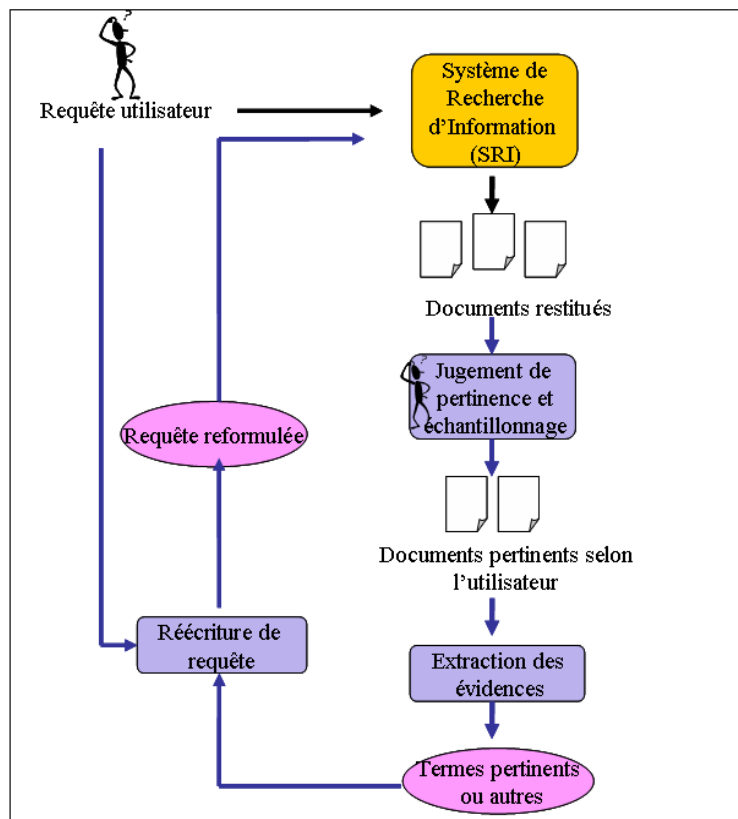


FIG. 2.2 – Le Processus général de la réinjection de pertinence

- La réécriture de la requête consiste à construire une nouvelle requête en combinant la requête initiale avec les informations extraites dans l'étape précédente.

Le processus général de la réinjection de pertinence peut être renouvelé plusieurs fois pour une même séance de recherche : on parle alors de la réinjection de pertinence à itérations multiples, dont les approches représentatives sont détaillées dans la section [2.3.6.2](#).

Considérons maintenant en détail les différentes phases du processus de réinjection de pertinence.

La phase d'échantillonnage ne présente pas de problématique spécifique. Le seul point abordé à ce niveau concerne le nombre d'éléments à évaluer pour pouvoir effectivement constituer un échantillon représentatif.

La problématique principale de la réinjection de pertinence réside dans les deux autres phases : l'extraction des termes (ils sont alors pondérés pour sélectionner les plus pertinents) et la réécriture de la requête avec repondération des termes. Dans la plupart des approches de la littérature, les deux phases sont effectuées avec des méthodes de pondération des termes similaires. Cependant certaines méthodes et particulièrement celles basées sur le modèle probabiliste, utilisent des méthodes de pondération différentes.

Dans la prochaine section nous proposons donc de détailler les méthodes d'extraction des termes. La reformulation de requêtes appliquée aux différents modèles de RI est ensuite décrite dans la section [2.3.4](#).

2.3.3 Méthodes d'extraction des termes

La reformulation de requête telle qu'elle a été initialement utilisée par Wu et Salton [208] consistait à ajouter tous les termes des documents pertinents retrouvés en réponse à la requête lors du processus de recherche. Cette méthode de sélection des termes peut être à l'origine de beaucoup de bruit (restitution de document non pertinents). En effet, les termes dans les premiers documents pertinents restitués ne sont pas tous significatifs. L'idée d'utiliser seulement une sélection de termes a été proposée par Harman [82]. La question est de savoir quels termes utiliser pour étendre la requête initiale de façon à améliorer le rappel et la précision du système.

L'approche présentée par Harman [82] consiste à sélectionner les dix pre-

miers documents et à identifier parmi ceux-ci les documents pertinents. Harman a utilisé différentes techniques pour ordonner les termes afin de choisir les vingt meilleurs termes de la liste. Il a été démontré que la technique utilisée pour le tri des termes pertinents a un large impact sur la performance. Dans plusieurs techniques de tri que l'auteur a définies, il utilise une mesure de bruit n_k calculée comme suit :

$$n_k = \sum_{i=1}^N \frac{tf_{ik}}{f_k} \log_2 \frac{f_k}{tf_{ik}} \quad (2.1)$$

Avec :

tf_{ik} le nombre d'apparition du terme k dans le document i ,
 f_k le nombre d'apparition du terme k dans la collection et
 N le nombre de termes dans la collection.

La technique a été étendue pour tenir compte du nombre de documents dans l'ensemble des documents pertinents contenant le terme k (p_k) et du nombre d'apparition du terme k dans l'ensemble des documents pertinents (rtf_k).

Harman a défini ainsi une autre mesure de bruit par rapport à l'ensemble des documents pertinents. Cette mesure est calculée comme suit :

$$rn_k = \sum_{i=1}^N p_k \frac{tf_{ik}}{rtf_k} \log_2 \frac{f_k}{tf_{ik}} \quad (2.2)$$

Dans [83], Harman a défini d'autres techniques de tri des termes. La technique qui conduit à de meilleurs résultats est basée sur une formule de pondération définie par Sparck-Jones et Robertson [156] :

$$W_{ij} = \log_2 \frac{p_{ij}(1 - q_{ij})}{q_{ij}(1 - p_{ij})} \quad (2.3)$$

Avec :

W_{ij} poids du terme i dans la requête j ,
 p_{ij} la probabilité que le terme i apparaisse dans les documents pertinents pour la requête j ,
 q_{ij} la probabilité que le terme i apparaisse dans les documents non pertinents pour la requête j .

La sélection des termes ayant une valeur de poids importante revient à sélectionner les termes caractéristiques des documents pertinents avec une faible probabilité d'apparition dans les documents non pertinents.

Harman [83] a également démontré que la meilleure méthode de sélection des termes issus des documents pertinents devient inefficace au-delà de 20 à 40 termes ajoutés.

Croft et *al.* [81] et Robertson et *al.* [157] ont adopté une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer à chaque terme un nombre traduisant sa valeur. Robertson propose la formule suivante pour calculer la valeur de sélection d'un terme :

$$selValue(i) = W_{ij} \times (P_i - U_i) \quad (2.4)$$

Avec :

W_{ij} défini dans l'équation 2.3,
 P_i la probabilité ($d_i = 1/D$ est pertinent); et U_i la probabilité ($d_i=0/D$ est non pertinent).

Les termes sont alors triés en fonction de leurs valeurs de pertinence puis sélectionnés en utilisant un seuil prédéfini.

Lundquist et *al.* ont étudié dans [127] une autre technique de tri des termes. Pour un terme k , les auteurs associent une valeur $p_k \times nidf$ où p_k est le nombre de documents dans l'ensemble des documents pertinents contenant le terme k , et $nidf$ est une fréquence absolue inverse normalisée utilisant la normalisation telle que définie par Singhal [183]. En utilisant la collection TIPSTER, Lundquist et *al.* [127] ont démontré que cette formule conduit à de bonnes performances. Par ailleurs, ils ont aussi démontré que l'utilisation des dix premiers termes (termes simples ou expressions) conduit à une amélioration de la précision moyenne de 31% par rapport à l'utilisation des cinquante premiers termes et vingt premières expressions.

Boughanem et *al.* [23] [21] ont quant à eux étudié la reformulation de requête sur un SRI basé sur l'approche connexionniste fondée sur les réseaux de neurones. Les termes ajoutés à la requête sont sélectionnés sur la base d'un seuil de cooccurrence avec les termes de la requête initiale. Ils ont conclu que la valeur idéale du seuil (c'est à dire la valeur permettant d'améliorer les résultats) varie de façon inversement proportionnelle à la taille de la base et à la taille moyenne des documents.

Buckley et *al.* ont démontré dans [28], que le taux de performance (Rappel-Précision) est davantage corrélé avec le nombre de termes ajoutés à la requête qu'avec le nombre de documents initialement retrouvés. Cette idée est traduite par l'équation suivante

$$RP(N) = A.log(N) + B.log(X) + C \quad (2.5)$$

Avec : $RP(N)$ la performance du système pour N documents restitués,
 N le nombre de documents restitués, et

X le nombre de termes ajoutés à la requête.

A, B, et C sont des constantes telles que $B \gg A > C$.

2.3.4 Principales approches de réinjection de pertinence en RI

2.3.4.1 Approche de Rocchio

La reformulation de requête a été introduite par Rocchio [158] dans le modèle vectoriel. Rocchio considère que la restitution des documents pertinents est liée à la notion de "requête optimale". Cette dernière est censée maximiser la différence entre le vecteur des documents pertinents et celui des documents non-pertinents.

Comme l'utilisateur n'est pas en mesure de soumettre une requête optimale, la réinjection de pertinence doit permettre de rapprocher le vecteur de la requête initiale du vecteur moyen des documents pertinents et de l'éloigner du vecteur moyen des documents non pertinents. Ceci est mis en œuvre par repondération des termes initiaux et ajout de nouveaux termes pondérés à la requête initiale. Les poids servent à la discrimination des documents pertinents des documents non pertinents. La formule originale de Rocchio est définie comme suit :

$$Q_1 = Q_0 + 1/n_r \sum_{i=1}^{n_r} R_i - 1/n_s \sum_{i=1}^{n_s} S_i \quad (2.6)$$

où

Q_0 est le vecteur de la requête initiale, Q_1 est le vecteur de la nouvelle requête, n_r est le nombre de documents pertinents, n_s le nombre de documents non pertinents, R_i est le vecteur du i^{eme} document pertinent et S_i le vecteur du i^{eme} document non pertinent.

Le nouveau vecteur de requête est le vecteur de la requête initiale plus les termes qui différencient au mieux les documents pertinents des documents non-pertinents. Une requête reformulée contient de nouveaux termes (extraits des documents jugés pertinents) associés à de nouveaux poids. Si le poids d'un terme de la requête décroît vers zéro ou au dessous de zéro, il est éliminé de l'ensemble des termes de la requête.

Une variante de cette formule a été examinée expérimentalement avec des résultats positifs sur le système de recherche SMART [158]. La petite taille de la collection de documents utilisée dans les expériences de Rocchio a engendré

certaines modifications dans la formule. Par exemple, un terme est seulement considéré s'il appartient à la requête initiale ou s'il apparaît plus dans les documents pertinents que dans les documents non-pertinents et dans plus que la moitié des documents pertinents. Ces modifications accentuent la difficulté d'aligner la théorie avec la pratique expérimentale.

Une autre modification apportée à cette formule qui permet de pondérer la contribution relative de la requête initiale, des documents pertinents et des documents non-pertinents dans le processus de RF . C'est la variante la plus répandue aujourd'hui (standard), elle est décrite dans l'équation suivante :

$$Q_1 = \alpha Q_0 + \beta/n_r \sum_{i=1}^{n_r} R_i - \gamma/n_s \sum_{i=1}^{n_s} S_i \quad (2.7)$$

où α , β et γ indiquent le degré d'effet de chaque composant sur le processus de réinjection de pertinence.

Ide [99] a étendu les expériences de réinjection de pertinence de SMART, en examinant différents aspects de RF . Il a par exemple étudié la restriction sur les documents jugés pertinents pour la réinjection de pertinence, en changeant le nombre de documents utilisés pour le processus de RF , et utilisé les documents non-pertinents. Il a également proposé une variante de la formule originale de Rocchio, en utilisant seulement le premier document non pertinent trouvé S_1 . La formule utilisée est la suivante :

$$Q_1 = Q_0 + \sum_{i=1}^{n_r} R_i - S_1 \quad (2.8)$$

Cette formule a été comparée à la formule originale de Rocchio. Bien que cette technique n'ait pas amélioré considérablement les résultats, elle était plus robuste, en permettant l'amélioration pour plus de requêtes.

2.3.4.2 Réinjection de pertinence dans le modèle probabiliste

Dans le modèle probabiliste développé par Robertson, Sparck Jones [156] et Van Rijsbergen [151], les documents et les requêtes questions sont également vu comme des vecteurs mais la mesure vectorielle de similarité est remplacée par une fonction probabiliste. On rappelle que le modèle probabiliste est basé sur la probabilité qu'un document soit pertinent à un utilisateur pour une requête donnée. Ce modèle est par essence même lié à la réinjection de pertinence, puisque ses paramètres sont estimés sur la base de la présence/absence des termes dans les documents pertinents et non pertinents.

Robertson et Sparck-Jones [156] utilisent la formule de pondération des termes suivante :

$$W_i = \log \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \quad (2.9)$$

W_i le poids du terme i , avec

$$p_i = P(t_i = 1/D \text{ est pertinent}) = \frac{r_i}{R}, \quad q_i = P(t_i = 1/D \text{ est non pertinent}) = \frac{n_i - r_i}{N - n_i}$$

où $t_i = 1$ si le terme i indexe le document, $t_i = 0$ sinon.

r_i le nombre de documents pertinents contenant le terme t_i ,

R le nombre de de documents pertinents pour la requête,

n_i le nombre de documents contenant le terme t_i et

N le nombre de documents dans la collection.

Les poids des termes ajoutés à la requête sont alors calculés selon la formule suivante :

$$W_i = \log \frac{r_i/R - r_i}{n_i - r_i / (N - n_i) - (R - r_i)} \quad (2.10)$$

Harman [83] a montré que l'utilisation de la formule de Sparck-Jones pour la repondération des termes, permet une augmentation de la précision de 25% sur la base Cranfield.

Croft [44] a défini une méthodologie de re-pondération en utilisant une version révisée de la formule de pondération de Sparck-Jones. Plus précisément, la recherche initiale suit la fonction de pondération des termes suivante :

$$W_{ijk} = (C + idf_i) \cdot f_{ik} \quad (2.11)$$

Avec

C une constante, f_{ik} la fréquence du terme t_i dans le document k , idf_i la fréquence absolue du terme t_i dans la collection et j la requête.

Pour re-pondérer des termes par réinjection de pertinence, Croft se base sur la formule de Robertson. La formule de re-pondération est la suivante :

$$W_{ijk} = \left[C + \log \frac{p_{ij}(1 - q_{ij})}{q_{ij}(1 - p_{ij})} \right] \cdot f_{ik} \quad (2.12)$$

Avec

W_{ijk} le poids du terme t_i dans la requête j et le document k ,

$$p_{ij} = \frac{r_i+0.5}{R+1.0} \text{ si } r_i > 0, p_{ij} = 0.01 \text{ si } r_i = 0,$$

$$q_{ij} = \frac{n_i-r_i+0.5}{N-R+1.0} \text{ si } r_i > 0, q_{ij} = 0.01 \text{ si } r_i = 0,$$

$$f_{ik} = K + (1 - K) \cdot \frac{freq_{ik}}{\max(freq_k)}$$

où $freq_{ik}$ est la fréquence du terme t_i dans le document k , $\max(freq_k)$ est le maximum des fréquences des termes dans le document k et C, K sont des constantes.

2.3.4.3 Réinjection de pertinence dans le modèle inférentiel

De Compos et *al.* ont utilisé dans [34] les réseaux bayésiens pour la recherche d'information. Les relations d'inférence traduisent soit les relations terme-document soit les relations terme-terme.

La réinjection de pertinence est basée sur la propagation des messages de type $\lambda(T) = \{0, 1\}$ (resp. $\lambda(T) = \{1, 0\}$) des nœuds des documents vers les termes pour exprimer la relation de pertinence (resp. de non pertinence) du terme. La pertinence d'un terme est ensuite calculée par la combinaison des messages provenant de différents documents. Les auteurs ont proposé 4 types de message :

1. un message de type $\lambda(T) = \{0, 1\}$ vers tous les termes t_i qui indexent les documents pertinents. Cette approche n'a pas donné d'amélioration puisque tous les termes des documents D_j pertinents sont considérés comme les termes de la requête initiale.
2. un message de type $\lambda_{D_j}(T_i) = \{p(D_j|\bar{t}_i), p(D_j|t_i)\}$ est propagé vers les termes, avec la probabilité de $t_i|t_j$ égale à 1 si $t_i = t_j$ sinon égale à 0.

$$p(D_j|t_i) = \alpha_j \sum_{T_k \in D_j} tf_{jk} \cdot idf_k^2 \cdot p(T_k|t_i)$$

α_j est une constante de normalisation calculée comme suit :

$$\alpha_j = 1 / \sqrt{\sum_{T_i \in D_j} tf_{ji} \cdot idf_i^2}$$

3. un troisième type de message qui tient compte de la requête initiale Q est de type $\lambda_{D_j}(T_i) = \{p(D_j/\bar{t}_i, Q), p(D_j/t_i, Q)\}$
4. Dans un quatrième type de message, la requête initiale joue le rôle d'un document pertinent : $\lambda_{D_j}(T_i) = \{p(D_j, Q|\bar{t}_i), p(D_j, Q|t_i)\}$

Les auteurs ont proposé deux méthodes pour la combinaison des messages reçus par un terme T_i provenant de S documents pertinents :

1. Une première qui est le produit direct des valeurs de $\lambda(T_i)$:

$$\lambda(T_i) = \prod_{j=1}^S \lambda_{D_j}(T_i) \quad (2.13)$$

2. Pour mieux tenir compte de la nature de requête la 2^{eme} formule est la suivante :

$$\lambda'(T_i) = \alpha + (1 - \alpha)\lambda(T_i) \quad (2.14)$$

Où α est le rapport du nombre des documents pertinents retrouvés sur le nombre de tous les documents pertinents.

On trouvera une autre application du modèle inférentiel à la réinjection de pertinence dans [81]. Les auteurs ont estimé la probabilité de pertinence d'un terme en fonction de son occurrence dans les documents pertinents. Les performances obtenues sont comparables à celles observées dans le modèle vectoriel.

2.3.4.4 Autres propositions

Chacun des modèles vectoriel et probabiliste suppose l'indépendance entre les termes. En d'autres termes, la présence d'un terme dans un document n'influe pas sur la probabilité de l'existence d'un autre terme dans le même document. Bien que cette hypothèse simplifiée facilite la construction de systèmes de recherche assez performants, l'indépendance des termes n'est pas fondée. En effet, les mots sont reliés par leur utilisation et des expressions, et leurs occurrences dans les documents peuvent refléter des relations sémantiques fondamentales entre les termes.

Des auteurs tels que Spiegel et Bennet [186] ont suggéré dès 1964 que cette dépendance de l'information peut être employée pour extraire d'autres termes pour l'extension de la requête. On distingue trois investigations sur la dépendance de l'information :

- Van Rijsbergen, et al. [152] ont proposé un arbre (MST) composé de nœuds représentant les termes et reliés par des arcs qui représentent les similarités entre deux termes. Cette similarité est estimée selon la mesure d'association basée sur la distribution des probabilités des deux termes. L'extension de la requête consiste à rajouter tous les termes directement liés aux termes de la requête initiale. L'ensemble des termes sera par la suite pondéré selon la formule de Robertson [156]. Les résultats ont montré une efficacité relative de cette approche.
- Smeaton et Van Rijsbergen [184] se sont concentrés sur trois méthodes pour l'extension de la requête : l'approche de MST de Van Rijsbergen

et *al.*, l'approche basée sur les voisins les plus proches (NN) (termes qui sont statistiquement les plus similaires aux termes de la requête) et l'extension d'une liste de termes extraits des documents jugés pertinents. Les résultats de ces expériences sont largement négatifs. Une des causes de la dégradation est le nombre de termes rajoutés à la requête initiale. Smeaton et Van Rijsbergen signalent que la difficulté d'estimer des probabilités est la raison principale de cet échec.

- Dans [19], Bhatia a également présenté un modèle d'arbres de dépendance pour l'extension de la requête en intégrant des informations spécifiques de l'utilisateur. Bhatia suggère que l'approche d'arbre de dépendance peut être améliorée non seulement par une sélection plus stricte mais aussi en pondérant les termes de la requête selon les préférences de l'utilisateur. Bien que la construction des arbres soit inspirée de la similitude statistique, elle ne considère pas la similitude conceptuelle. La solution présentée demande d'obtenir de l'utilisateur les relations entre les concepts présents dans les documents. Ceci peut être utilisé pour développer un nouvel arbre qui reflète plus exactement des relations conceptuelles personnalisées selon l'utilisateur. Un arbre de dépendance devrait être construit pour chaque utilisateur.

Une approche alternative exploitant la dépendance des termes consiste à grouper des termes reliés avec des termes d'extension de la requête. Ceci peut être réalisé sans information de pertinence (en utilisant seulement l'information statistique sur la similarité des termes) ou avec de l'information de pertinence (en utilisant une combinaison de la dépendance d'information dans une collection et l'information pertinente pour choisir les termes d'extension). Ces deux méthodes se fondent typiquement sur des méthodes de co-occurrence des termes utilisées dans la littérature et n'ont pas généralement fourni de résultats convaincants [142].

Les méthodes décrites précédemment qui intègrent la dépendance des termes n'ont pas permis une amélioration des performances des systèmes de recherche [152], [19], [184]. Ceci peut être dû aux limitations informatiques pour calculer et stocker l'information de la dépendance. Bien que les méthodes d'indépendance des termes telles que celles basées sur le modèle probabiliste semblent simplifiées et n'expriment pas explicitement la dépendance des termes pertinents, elles permettent implicitement d'exprimer un certain degré de co-occurrence des termes. C'est-à-dire, même si les méthodes d'indépendance des termes ne calculent pas de valeurs explicites de co-occurrence, on estime que les termes dans la liste d'extension ont un degré supérieur à la co-occurrence moyenne des termes. Ceci peut être expliqué par le fait que les bons discriminateurs de pertinence sont les termes qui apparaissent plus fréquemment dans les documents pertinents que dans les documents non pertinents. L'utilisation efficace de la

co-occurrence des termes reste une question ouverte en recherche d'information.

2.3.5 Reformulation par réinjection de pertinence négative

D'après Ruthven et Lalmas [161], la majorité des techniques proposées en *RF* est basée sur la différence entre le contenu des documents pertinents et celui des documents non pertinents. Ces derniers se rapportent à deux groupes de documents :

1. ceux qui ont été jugés non pertinents explicitement par l'utilisateur ;
2. ceux qui n'ont pas été jugés par l'utilisateur. Ces documents sont soit non sélectionnés, l'utilisateur ne les a pas jugés, soit l'utilisateur les a rejetés implicitement sans fournir une évaluation de pertinence.

La différence entre ces deux groupes de documents non pertinents n'est pas exprimée dans les modèles probabiliste et vectoriel.

La *RF* utilisant le groupe des documents jugés explicitement non pertinents est appelée *RF* négative. D'après [161], cette dernière est considérée comme problématique pour trois raisons principales :

1. L'implantation : La *RF* négative présente une difficulté au niveau du traitement des informations négatives par le système. Une pratique courante en RI est de supprimer les termes ayant un poids négatif. Ces termes permettent plutôt la recherche de documents non pertinents que de documents pertinents. La Réinjection de pertinence négative peut être utilisée pour indiquer les termes devant avoir un poids négatif.

Dans ce contexte, Belkin et *al.*, dans une étude de la participation de l'utilisateur dans la réinjection de pertinence [15, 13, 14, 17], proposent un modèle alternatif. Leur hypothèse est qu'un terme appartenant à un document pertinent ou à un document non pertinent peut être intéressant puisqu'il permet d'augmenter le nombre de documents pertinents (s'il appartient à un document pertinent) ou de diminuer le nombre de documents non pertinents (dans le cas contraire).

Le but de la réinjection de pertinence négative abordé par Sumner et al. [188], était la suppression des documents non pertinents précédemment vus par l'utilisateur mais pouvant réapparaître dans la liste des résultats s'ils répondent à la nouvelle requête. Les expérimentations dans [15] montrent des résultats comparables pour la réinjection de pertinence positive et la réinjection négative, mais laissent entendre des améliorations potentielles en utilisant une combinaison des deux. Les expérimentations dans [13] ont montré que bien que les utilisateurs puissent utiliser la réinjection de pertinence négative, l'amélioration

des performances n'est pas significative.

2. Clarté : Il est difficile de spécifier les conditions dans lesquelles un utilisateur doit considérer un document non pertinent. En effet, un document est considéré non pertinent s'il ne contient absolument aucune information pertinente, s'il ne contient aucune information liée aux besoins de l'utilisateur, s'il contient l'information liée au thème en question mais pas l'information pertinente, si le document n'est pas assez pertinent, etc. La question est quand un utilisateur devrait-il juger un document non pertinent ?

Ce problème existe aussi dans le cas de la réinjection de pertinence positive mais pour deux raisons, la problématique est plus centrale dans le cas de réinjection de pertinence négative. D'abord, comme prouvé par les expérimentations de Belkin et al. [13], les effets de la réinjection de pertinence négative ne sont pas clairs pour les utilisateurs. Dans le cas de la réinjection de pertinence positive, le genre de documents recherchés, ainsi que les changements effectués par le système apparaissent avec plus de clarté, contrairement à la réinjection négative pour laquelle l'utilisateur ne peut pas voir quels documents ont été supprimés.

Deuxièmement, le jugement de non pertinence est une tâche plus délicate que le jugement de pertinence [161]. Dans la pratique, la pertinence et la non pertinence ne sont pas des notions opposées. En général, un utilisateur qui juge un document pertinent donne souvent des raisons détaillées, mais les raisons de la non-pertinence sont susceptibles d'être basées sur ce qui manque dans le document, plutôt que sur ce qui est présent.

3. Rentabilité : Bien que les techniques de *RF* puissent améliorer une recherche, les utilisateurs ne font pas toujours d'évaluation de pertinence. Ceci peut être dû à un manque de conscience de la part de l'utilisateur de l'utilité de la réinjection de pertinence. La rentabilité des évaluations peut avoir un effet sur la façon dont probablement les utilisateurs doivent évaluer. Plus les modalités d'évaluation sont compliquées moins les utilisateurs évaluent la pertinence, ce qui est le cas de l'évaluation de la non pertinence.

2.3.6 Autres formes de Réinjection de pertinence

2.3.6.1 Réinjection automatique de pertinence

La réinjection de pertinence décrite jusque là est basée sur les jugements de l'utilisateur. Une approche alternative, connue sous le nom de *pseudo-réinjection* ou *blind Relevance Feedback*, utilise des techniques de réinjection automatique

à l’aveugle pour construire une nouvelle requête. Plus précisément, le système de recherche restitue un ensemble de documents répondant à la requête initiale. Ainsi au lieu de juger explicitement les documents, on suppose que les k premiers documents comme étant pertinents (documents pseudo-pertinents). On peut également considérer les documents qui sont restitués en fin de liste comme non pertinents. L’idée de base derrière la pseudo réinjection de pertinence est qu’une itération de réinjection basée sur les documents les plus similaires à la requête initiale de l’utilisateur pourrait donner une meilleure restitution des documents.

Cette technique a été développée la première fois par Croft & Harper [45], en tant qu’un moyen d’estimation des probabilités dans le modèle probabiliste pour une première recherche. Depuis, cette technique a été largement étudiée pour améliorer les classements des documents en particulier dans le cadre de TREC [203]. Croft Harper ont également indiqué que cette méthode peut avoir des impacts négatifs. En effet si les documents considérés pour la réinjection contiennent peu d’informations pertinentes ou aucune, la réinjection ajoutera des termes à la requête initiale qui sont ”pauvres” à détecter la pertinence, et par conséquent pour la recherche des documents pertinents.

La réinjection automatique peut être bénéfique si les requêtes initiales permettent de retrouver des documents pertinents, dans le cas contraire elle provoque une dégradation des performances. Des chercheurs comme Mitra et *al.* [138] et Buckley et *al.*, [29], ont essayé avec un certain succès de surmonter ce problème en améliorant le taux de précision dans les k meilleurs documents, c’est ce qu’on nomme habituellement la ”haute précision”. D’autres groupes de recherche comme Efthimiadis et Biron, [55], Robertson et *al.*, [154] et Lee [118] se sont concentrés sur l’amélioration des techniques de réinjection afin de détecter les meilleurs termes à ajouter ainsi que sur le calcul de leurs poids.

Il est prouvé dans la majorité des travaux que la réinjection automatique présente une solution pratique pour l’amélioration des performances de la recherche en ligne sous un certain nombre de conditions. En particulier, c’est une technique très utile pour améliorer la recherche quand il s’agit de requêtes courtes ou de requêtes qui ne permettent pas de restituer assez de documents pertinents. Les améliorations observées en particulier dans le cadre de TREC sont faibles [103].

Pour répondre aux limites de cette technique, il est nécessaire de faire intervenir l’utilisateur dans le processus de réinjection de pertinence. Dans une section ultérieure, nous détaillons une approche qui permet la modification de la requête utilisateur d’une manière interactive.

2.3.6.2 Réinjection de pertinence à itérations multiples

Dans une série d'articles traitant des besoins d'information, Campbell a abordé la notion du besoin dynamique [31, 32, 33] à travers la notion de la "pertinence ostensive". L'idée derrière la pertinence ostensive est que des documents jugés pertinents dans une itération courante de *RF* présentent des indicateurs plus intéressants que ceux retrouvés dans des itérations précédentes. Cependant, les documents pertinents ne sont pas considérés d'égale importance mais d'importance variable. Dans [33], Campbell et Van Rijsbergen ont étendu le modèle probabiliste en intégrant un terme de "vieillesse" pour la pondération des termes pertinents. Ce concept permet de savoir si le document auquel appartient le terme est récemment jugé pertinent ou jugé dans des itérations antérieures. Dans [32], des expérimentations préliminaires de cette approche ont montré que la pondération ostensive peut améliorer les résultats en moins d'itérations de recherche que les approches non-ostensives. Ruthven et al. ont montré également que la pondération ostensive est bénéfique pour l'extension de la requête [162].

2.3.6.3 Extension interactive de requêtes

Dans le cas des méthodes d'extension automatique des requêtes décrites précédemment, les termes sont extraits à partir des documents et ajoutés en totalité à la requête. Une alternative est de permettre aux utilisateurs de choisir les termes pouvant être ajoutés : on parle d'*Expansion Interactive des Requêtes* (EIR) [82]. L'utilisateur qui est le mieux placé pour déterminer la pertinence, a alors plus de contrôle sur les termes qui seront ajoutés à la requête. Cette technique est défendue par le fait que l'utilisateur peut mieux sélectionner les termes pertinents que le système.

Bien que les systèmes aient accès à l'information statistique interne qui leur permet de choisir de bons termes discriminatoires, les utilisateurs peuvent prendre une décision plus perspicace de la pertinence. La question est alors comment concevoir un système d'extension de requêtes interactif pour traduire les avantages potentiels de l'EIR afin d'améliorer les performances des systèmes de recherche. Il y a plusieurs problématiques associées à ce problème, qui concernent en général les interfaces dynamiques.

2.3.6.4 Combinaison d'algorithmes de réinjection de pertinence

Une autre application de la réinjection de pertinence est la combinaison des résultats de différentes méthodes de réinjection. Ceci pourrait impliquer de combiner les classements donnés par les différentes méthodes de réinjection sur

les mêmes évaluations originales de requête et de pertinence, ou la combinaison des requêtes modifiées selon plusieurs méthodes de réinjection. Cette approche a été expérimentée et validée par Lee dans [118].

D'après Ruthven et Lalmas [161], la combinaison des évidences est une technique puissante pour la réinjection de pertinence, cependant, la majorité de techniques évaluées ont prouvé que cette combinaison est une technique très variable pour la recherche initiale : elle permet d'améliorer la performance pour quelques requêtes mais aussi de la dégrader pour d'autres. En outre, il est également très difficile de prévoir quelles sont les évidences à combiner pour différentes collections ou requêtes.

2.4 Réinjection de pertinence en RIS

2.4.1 Problématiques de la réinjection de pertinence en RIS

La nature des documents semi-structurés, comportant du texte et des informations structurales, réactualise la problématique de la RI classique (plein texte) en général et de la reformulation de requêtes en particulier.

L'objectif de la *RF* en RIS est d'enrichir la requête initiale (comme en RI traditionnelle) afin de mieux exprimer les besoins de l'utilisateur.

Comme nous l'avons déjà vu dans l'introduction générale, plusieurs questions se posent dans ce contexte. Elles portent principalement sur la manière de prendre en compte le contenu et structure lors de la reformulation de requête. Nous résumons ici les principales :

- En RI classique, l'unité documentaire jugée et donc à partir de laquelle les termes sont extraits, est le document entier. Les méthodes proposées ont montré leur intérêt en termes de rappel-précision [158], [156]. Or dans le contexte de la RIS, l'unité documentaire peut avoir différentes formes. Elle peut être le document entier ou tout élément du document. Une adaptation simpliste des méthodes de la RI classique à la RI structurée consisterait à extraire les termes pertinents à partir des éléments de différentes granularités jugés pertinents par l'utilisateur. Cette adaptation simpliste est-elle en adéquation avec la RI structurée ? Comment tenir compte du fait que les éléments peuvent être imbriqués les uns dans les autres ? Permet-elle effectivement d'améliorer les performances de la recherche ? Au lieu de sélectionner indifféremment tous les éléments pertinents pour l'extraction des termes, doit-on au contraire prendre en compte les sémantiques différentes des éléments (par exemple, *paragraphe*, *titre*,

section) ?

- La reformulation de requêtes s’est intéressée à enrichir la requête initiale par extraction et réinjection des termes pertinents, mais qu’en est-il de la dimension structurelle ? Est-il intéressant d’enrichir une requête avec des contraintes structurelles ?
- Une dernière question concernant le processus de la reformulation est la réécriture de la requête. D’une manière générale, on aura à rajouter des termes pertinents et/ou des structures pertinentes à des requêtes structurées et non structurées. La question est comment intégrer ces deux évidences dans la requête initiale ? Comment pondérer les termes ? Doit-on re-pondérer les termes originaux ? Comment rajouter des structures à des requêtes déjà structurées ? A quels groupes de mots-clés doit-on ajouter des conditions structurelles ?

2.4.2 Principales approches de la réinjection de pertinence en RIS

On classe les différentes approches développées en *RF* dans les documents structurés selon deux principales voies :

- des approches **orientées contenu** qui se basent sur l’extraction, la pondération et la réinjection des termes pertinents dans la requête initiale,
- d’autres approches **orientées contexte** qui consistent à extraire des informations concernant le contexte des éléments jugés pertinents comme le document dans lequel ils se trouvent, le nom des balises, les balises des ancêtres, des descendants, la taille, etc.

2.4.2.1 Approches orientées contenu

La majorité des approches proposées dans ce cadre ont simplement adopté l’algorithme de Rocchio à la recherche d’information structurée. Ceci consiste de manière générale à extraire les termes à partir des éléments jugés pertinents et les pondérer. On trouve ces adaptations dans plusieurs travaux que nous décrivons brièvement dans ce qui suit.

Réinjection de pertinence à la Rocchio

En 2004, Mass et Mandelbrod ont développé une première approche orientée

contenu [131] appliquée à leur système de recherche basé sur le modèle vectoriel étendu développé en 2002 [133]. La réinjection de pertinence ne concerne que les requêtes composées de simples mots clés. Comme nous l'avons mentionné dans le chapitre précédent leur système de base consiste en une recherche indépendante au niveau de chaque index³.

Les auteurs ont proposé un algorithme [132] basé sur la formule de Rocchio [158], qui est appliqué à chaque type d'éléments. C'est une adaptation exacte de l'algorithme de Rocchio (section 2.3.4.1) : au lieu de considérer le document entier, on considère les éléments. D'après les résultats obtenus dans la campagne d'évaluation INEX 2005 [69], les améliorations ne dépassent pas 5%.

Crouch et al. [47] ont eux aussi appliqué l'algorithme de Rocchio sur leur système de recherche basé sur la propagation de pertinence. Ces travaux ont principalement évalué l'intérêt de prendre en compte une seule dimension de pertinence pour sélectionner les éléments pertinents, en l'occurrence l'exhaustivité. Plus précisément, seuls les paragraphes ayant une valeur d'exhaustivité élevée sont utilisés dans le processus de la réinjection. Dans le cadre de INEX 2004 [67], les améliorations n'ont pas dépassé un taux de < 4%.

Sigurbjörnsson et al. [180] ont eux aussi utilisé l'algorithme de Rocchio pour la réinjection de pertinence aveugle sur leur modèle de recherche basé sur la combinaison des modèles de langage de l'élément, de document et de la collection (voir chapitre précédent, section 1.2.2.5). Ils ont évalué l'extraction des termes à partir de l'index *article* ainsi de l'index des éléments. Les résultats obtenus dépendent des requêtes et ne permettent pas de tirer des conclusions claires.

Autres propositions pour la réinjection de pertinence

On trouve également dans la littérature des stratégies de sélection de termes ad-hoc comme celle proposée par Geva [72]. Elle consiste à d'extraire les dix premiers éléments, d'éliminer ensuite les mots qui ocurrent plus de 50000 fois dans la collection et ceux qui ont un taux d'occurrence de moins de 20% dans les éléments retournés. Le reste des mots est classé par ordre décroissant selon leurs fréquences, les 5 premiers sont rajoutés à la requête. Les évaluations ne montrent aucune amélioration des résultats.

³L'index est composé de plusieurs sous-index où chacun représente l'ensemble des éléments des documents ayant un type unique, comme par exemple *paragraphe*, *section*,...

2.4.2.2 Approches orientées contexte

On appelle contexte toute information décrivant les éléments pertinents non pas du point de vue contenu (mots clés) mais par rapport au contexte dans lequel ils apparaissent : balise, taille, ancêtre, descendant, etc.

On retrouve l'utilisation de cette notion en réinjection de pertinence dans le cadre des travaux de Sigurbjörnsson et *al.* [135], [136] en particulier dans le système de recherche TIJAH [180] basé sur un modèle de langage.

Plus précisément, ces auteurs ont proposé d'utiliser le contexte implicite des éléments jugés pertinents pour améliorer leurs performances. La reformulation est réalisée en deux étapes :

- Extraction du contexte des éléments les plus pertinents.
- Reformulation de la requête orientée contenu en réinjectant les informations contextuelles déjà extraites.

Pour extraire l'information contextuelle, les auteurs ont utilisé les noms des journaux⁴ auxquels appartiennent les éléments jugés pertinents parmi les 20 premiers retrouvés par le système, les noms de ces éléments ainsi que leur taille. Ils ont considéré que si un élément est évalué comme pertinent pour une requête donnée, le journal auquel il appartient est susceptible de contenir des informations semblables. Le nom de la balise XML dans lequel se trouve l'élément pertinent est également utilisé dans ce processus dans le but de privilégier dans la liste des résultats trouvés, les éléments ayant le même nom de balise. le troisième indicateur utilisé est la taille des éléments. L'idée de tenir compte de la taille d'un élément est basée sur le fait que les éléments ayant une taille rapprochée à celle des éléments pertinents sont probablement désirés davantage par l'utilisateur.

La priorité d'un journal est calculée selon la formule suivante :

$$P(J) = a + b \cdot \frac{\sum_{r \in \text{top}_{20} \subseteq J} E_r}{3 \cdot |\{r \in \text{top}_{20} | E_r > 0\}|} + (1 - a - b) \cdot \frac{|J \supseteq \text{top}_{20}|}{20} \quad (2.15)$$

avec E_r est la valeur d'exhaustivité de l'élément r appartenant au vingt premiers éléments issu d'un journal J et a, b des poids accordés à l'importance de l'information. Cette équation n'utilise que l'exhaustivité, elle traduit le fait que plus un journal est exhaustif plus il contient d'éléments pertinents.

Pour le calcul de priorité d'un élément, on tient compte de l'exhaustivité

⁴Les expérimentations sont réalisées sur la base d'INEX 2004 où la collection est composée d'articles de journaux IEEE.

ainsi que de la spécificité pour traduire si l'élément est trop grand ou trop petit. La priorité d'un élément est exprimée par la formule suivante :

$$P(e) = a + b \cdot \frac{\sum_{r \in \text{top}_{20} \subseteq e} E_r + S_r}{6 \cdot |\{r \in \text{top}_{20} | E_r \cdot S_r > 0\}|} + (1 - a - b) \cdot \frac{|e \in \text{top}_{20}|}{20} \quad (2.16)$$

avec S_r la valeur de spécificité de l'élément r appartenant au vingt premiers éléments.

La taille désirée est estimée selon une combinaison des tailles des 20 meilleurs éléments en fonction de leurs valeurs d'exhaustivité et de spécificité.

Pour la réécriture de la requête de type CO, les auteurs ont proposé des requêtes contenant des mots clés et les balises dans lesquelles l'utilisateur souhaite trouver l'information. Un poids $P(e)$ déjà calculé est associé aux contraintes structurelles (balises) qui sont reliées par l'opérateur booléen "or". Plusieurs combinaisons de contexte ont été évaluées. Les améliorations sont comparables et ne dépassent pas 6%.

Réinjection de pertinence orientée contexte structurel

Dans [175], Schenkel *et al.* ont proposé une technique permettant d'étendre la requête initiale de type CO en requêtes structurées, en intégrant le contexte structurel des éléments. Ce dernier est composé d'un ensemble de champs décrivant l'ancêtre, les descendants ainsi que le contenu de chaque élément. Les auteurs ont alors défini 4 classes de caractéristiques à partir des éléments jugés pertinents :

- Les termes composant le contenu de l'élément (classe C),
- Le couple balise-terme dans les descendants de l'élément (classe D).
- Le couple balise-terme dans les ancêtres de l'élément (classe A).
- Le couple balise-terme dans les descendants des ancêtres de l'élément (classe AD).

Tous les candidats de chaque classe sont pondérés par un score $W_{RSJ}(c)$ calculé selon la formule de Robertson :

$$W_{RSJ}(c) = \log \frac{r_c + 0.5}{R - r_c + 0.5} + \log \frac{E - ef_c - R + r_c + 0.5}{ef_c - r_c + 0.5} \quad (2.17)$$

Avec r_c est le nombre d'éléments pertinents qui contiennent le terme candidat c , R le nombre d'éléments pertinents, E le nombre d'éléments dans la collection, et ef_c la fréquence d'élément pour un candidat (nombre d'éléments où le candidat apparaît).

Pour un candidat c , la valeur de RSV est calculée comme suit :

$$RSV(c) = W_{RSJ}(c).(p - q) \quad (2.18)$$

Où $p = r_c/R$ est la probabilité d'occurrence d'un candidat dans l'ensemble des éléments pertinents et q est la probabilité d'occurrence d'un candidat dans l'ensemble des éléments non pertinents.

Les auteurs ignorent par la suite les candidats qui apparaissent seulement dans les descendants des éléments. Le reste des candidats est trié en fonction de leurs valeurs RSV. Les candidats ayant les meilleurs poids sont par la suite sélectionnés. Les candidats des classes A et AD doivent avoir le même ancêtre pour être sélectionnés.

La requête étendue est reformulée comme suit (en langage NEXI [196]) :

```
//balise-ancêtre [contraintes d'A+AD] // * [mots clés initiaux +contraintes de C+D].
```

Par exemple, si la requête initiale est composée d'un simple mot clé "XML" et on considère les candidats : A (ancêtre, article, 'IR'), AD (article, bib, 'index') et D (descendant, p, 'index'), la requête étendue est :

```
//article [about(. , 'IR') and about (//bib, 'index')] // *[about (. , 'XML') and about(//p, 'index')].
```

Les auteurs ont ensuite procédé de manière à attribuer les meilleurs scores (RSV(c)) aux candidats des classes C et D tout en restant inférieurs aux scores des mots clés initiaux et à pénaliser les scores des candidats des classes A et AD en les multipliant par une constante $\beta = 0.2$.

Les expérimentations montrent que les meilleurs résultats sont obtenus par la combinaison de toutes les caractéristiques et que les évaluations selon le INEX 2006 montrent des améliorations de l'ordre de 25%.

On remarque que cette approche permet d'enrichir les requêtes par des couples balise-terme mais n'exprime pas explicitement de relations entre les termes et les structures et que les termes sont extraits de manière indépendante les uns des autres.

Une alternative de l'utilisation du contexte est appliquée pour le réordonnement des résultats [175]. La réinjection de pertinence est alors utilisée pour évaluer le nouveau score des éléments. Ce calcul est effectué en fonction des caractéristiques contextuelles extraites à partir des 20 premiers éléments jugés. Ce score sera par la suite ajouté au score initial de l'élément.

Les auteurs ont défini alors des classes descriptives :

- Les termes composant le contenu de l'élément (classe C)

- Les couple balise-termes dans le document de l'élément (classe D) qui peut renseigner sur la nature des éléments qui peuvent appartenir au document, et
- Les dérivés du chemin des éléments (classe P) : préfixe du chemin, sous chemin, suffixe du chemin, chemin entier ...

Pour chaque classe, on calcule le poids des candidats selon la formule de Rocchio. Pour chaque élément du résultat de base, on calcule un score pour chaque classe dans un espace vectoriel où chaque dimension correspond à un candidat qui se produit dans au moins un élément des vingt meilleurs éléments. Les scores correspondant à chaque classe sont alors calculés comme le cosinus du vecteur composé des k meilleurs candidats. Chacun des scores est normalisé dans l'intervalle $[-1.0, 1.0]$. Le score final de l'élément est la somme de ses scores calculés ajoutée au score initial.

Les expérimentations ont montré que les meilleurs résultats sont obtenus en ne considérant que la classe décrivant le document (D) et la classe des dérivés du chemin (P). Cependant, en suivant le protocole d'INEX, les améliorations ne dépassent pas 2%.

2.4.3 Bilan

On constate que l'ensemble des approches orientées contenu proposées consistent à enrichir une requête initiale en rajoutant des termes pertinents. Ces termes sont sélectionnés en fonction de mesures statistiques basées sur les fréquences des termes dans les éléments pertinents. Dans le cas du modèle vectoriel étendu, les index correspondent à des types d'éléments prédéfinis, ce qui rend cette approche contraignante puisqu'elle dépend d'un type de DTD bien défini. On remarque également qu'en aucun cas la sémantique des éléments n'a été prise en compte : les termes sont sélectionnés indépendamment du type des éléments pertinents pris en compte.

Les approches orientées contexte permettent d'enrichir les requêtes par le contexte des éléments pertinents indépendamment des termes. On rajoute en général des préférences sur le contexte mais en aucun cas on ne spécifie la structure d'élément recherché, c'est-à-dire une structure exacte dans lequel on retrouve tous les éléments pertinents.

Enfin, les deux types d'approches (orientée contenu et orientée contexte) sont appliquées indépendamment, alors qu'il serait intéressant de voir ce que donnerait leur combinaison.

2.5 Évaluation de la reformulation de requêtes

2.5.1 Différentes stratégies d'évaluation de la reformulation

Dès le début des années 70, Chang *et al.* [38] ont démontré que l'évaluation des algorithmes de *RF* pose certains problèmes pour le rappel et la précision. Étant donné que la réinjection de pertinence utilise l'information extraite à partir des documents jugés pertinents, il est évident qu'un des effets principaux de la *RF* est de pousser les documents jugés pertinents au dessus de leur rangs initiaux. Ce ré-ordonnement améliorera artificiellement les valeurs de rappel précision. Ceci rend difficile l'examen de l'impact de la réinjection de pertinence sur la restitution des documents pertinents. Chang *et al.* [38] ont donc étudié trois solutions pour mettre en évidence les impacts invisibles de la réinjection de pertinence.

– Rang résiduel :

Cette technique consiste à éliminer du résultat final, les documents qui sont utilisés pour le jugement de pertinence. Ceci inclura les documents pertinents et non pertinents. Après la réinjection de pertinence, les taux de rappel précision sont calculés sur le résultat (résiduel) restant.

L'avantage de cette méthode est qu'elle considère seulement l'effet de la réinjection sur les documents pertinents restitués. L'inconvénient de cette stratégie est que, à chaque itération de réinjection, les valeurs de Rappel|Précision peuvent être basées sur différents nombres de requêtes. Ceci est dû au fait que des documents pertinents sont éliminés de la collection. Si tous documents pertinents pour une requête donnée sont supprimés, alors la requête ne peut plus être utilisée dans des itérations suivantes puisqu'il n'y a aucun document pertinent pour calculer les valeurs de rappel-précision.

Cette méthode est seulement appropriée à un nombre restreint d'itérations de réinjection, autrement le nombre de documents pertinents dans la collection résiduelle peut devenir relativement petit et peu représentatif de l'ensemble des documents pertinents.

En outre les performances avant/après reformulation ne sont pas réellement comparables, puisqu'elles sont effectuées sur des collections différentes. Pour avoir deux types d'ordonnement différents mais directement comparables, Salton [166] a utilisé la collection résiduelle avant et après la réinjection.

– "Freezing"

La méthode connue sous le nom de "freezing" [161] est basée sur les rangs des documents dans la liste des résultats et elle existe sous deux formes : "blo-

cage entier” et ”blocage modifié”. Dans le cas du blocage entier, les rangs des n meilleurs documents jugés pour la réinjection sont bloqués. Les documents restants sont retirés et les valeurs Rappel/Précision sont calculées pour l’ensemble des documents. Comme les seuls documents à changer de rangs sont ceux qui succèdent les n meilleurs documents, aucun changement de Rappel|Précision ne se produit avant le rang n .

Dans le cas du ”blocage modifié”, les rangs sont bloqués au rang du dernier document jugé pertinent.

L’inconvénient des approches de ”freezing” est qu’à chaque itération de réinjection une proportion plus intéressante de documents pertinents est bloquée. Ceci signifie que les n meilleurs documents bloqués contribue plus au taux de rappel-précision aux itérations postérieures de la réinjection. Bien que la réinjection puisse fonctionner mieux à ces itérations postérieures, elle peut sembler moyennement efficace à cause d’une contribution plus élevée des documents bloqués (i.e. les améliorations ne concernent qu’une partie de plus en plus petite de l’ensemble des résultats).

Dans ce qui précède nous avons mentionné que la méthode du rang résiduel force à éliminer les requêtes pour lesquelles tous les documents pertinents avaient été trouvés. Pour les méthodes de ”freezing”, une fois que tous documents pertinents pour une requête donnée, ont été trouvés, les taux de rappel-précision peuvent encore être calculés. Cependant ces taux ne changeront pas une fois que tous documents pertinents ont été bloqués. Intuitivement ceci semble correct : une fois que nous avons trouvé tous les documents pertinents pour une requête donnée, la réinjection n’améliore pas ou n’empire pas l’efficacité de récupération des documents pertinents.

– groupes d’essai et de test.

Dans cette technique [161], la collection de documents est aléatoirement coupée en deux collections : le groupe d’essai et le groupe de test. La reformulation de requête est effectuée par réinjection de pertinence sur le groupe d’essai et la nouvelle requête est alors exécutée dans le groupe de test. Les taux de Rappel|précision sont évalués seulement au niveau du groupe de test, il n’y a donc aucun effet de rang. Des requêtes successives peuvent être lancées sur le groupe de test pour évaluer des requêtes reformulées sur une collection de documents qui peut être considérée complète, contrairement de la méthode de rang résiduel.

À la différence des méthodes de ”freezing”, tous les documents pertinents dans le groupe de test sont libres de se déplacer dans la liste triée des documents. Ceci signifie que les taux de rappel-précision, avant et après refor-

mulation de requête, sont directement comparables. La difficulté avec cette méthode d'évaluation est de dédoubler la collection. Il est facile de dédoubler aléatoirement une collection de document (par exemple en mettant tous les documents pairs dans le groupe d'essai et tous les documents impairs dans le groupe de test). Cependant, cette distribution n'assurera pas le fait que les documents pertinents sont également dédoublés entre les deux collections. En aucun cas, on ne peut s'assurer que les documents pertinents dans le groupe d'essai sont représentatifs de ceux dans le groupe de test. D'autres facteurs tels que la longueur des documents ou la distribution des termes d'index peuvent également être importants pour la méthode de réinjection examinée, et on ne peut pas également s'assurer que la distribution des termes est dédoublée entre les deux collections.

Chacune de ces méthodes a des avantages et des inconvénients mais toutes sont des méthodes standards pour évaluer des algorithmes de réinjection de pertinence. Cependant, elles comparent seulement l'exécution des algorithmes dans des conditions idéales [112, 57, 187].

Un point final concernant les mesures d'évaluation de la réinjection de pertinence est qu'elles peuvent ne pas être directement comparables : chaque mesure peut donner différents résultats selon la façon dont les résultats sont comparés et sur quels facteurs la recherche est effectuée.

En conclusion, les mesures d'évaluation calculent différents aspects de réinjection : la stratégie de "freezing" mesure l'efficacité cumulative, le rang résiduel mesure l'efficacité de rechercher seulement les documents pertinents restants et le groupe d'essai et de test mesure la performance relative des requêtes reformulées produites à chaque itération.

2.5.2 Évaluation selon la campagne d'évaluation INEX

Le protocole de la campagne d'évaluation INEX 2005 et 2006 [5, 68] consiste à considérer le jugement de pertinence des 20 premiers éléments retournés par le système de base pour les requêtes CO et les requêtes de type CAS. Seule la stratégie de recherche "Thorough" (recherche de tous les éléments pertinents) est utilisée dans la tâche de *RF*. Le processus de réinjection de pertinence peut être appliqué en plusieurs itérations pour une requête donnée. Il n'y a aucune restriction sur le nombre d'itérations.

Un run⁵ de *RF* est établi comme suit : on utilise les jugements de pertinence des 20 premiers éléments du résultat de la recherche initiale. Les éléments jugés sont alors bloqués avec leur rang original et le reste des éléments sont triés à la suite des 20 premiers éléments. Si on applique plusieurs itérations de *RF*, pour

⁵Dans le jargon de la RI, on appelle run l'ensemble des résultats d'un système donné pour un jeu de requêtes données.

chaque itération i , les éléments jugés sont "bloqués" (gardent les mêmes positions que les runs de base) de la position $(i-1) * 20$ jusqu'à ce qu'à la position $i * 20 - 1$. Le reste des éléments pourvus des éléments jugés est trié à partir de la position $n * 20 - 1$, avec n le nombre d'itérations.

En 2006, les organisateurs ont proposé de varier les expérimentations avec différentes stratégies de post-réinjection pour éliminer l'influence des éléments dont la pertinence est connue sur les résultats, parmi elles plusieurs variantes de la méthode du rang résiduel ou du "freezing" sur n éléments, n étant à fixer. Seuls les résultats utilisant la stratégie de "freezing" à 20 éléments sont évalués officiellement.

Afin de pouvoir comparer les requêtes reformulées, on a proposé d'indiquer la requête reformulée utilisée après réinjection de pertinence. Le format pour cette requête reformulée suit le langage de requête NEXI avec les poids additionnels et facultatifs pour les termes, par exemple,

```
//article [about (. , 0.5*XML 0.75*database -0.3*index)]
```

Pour évaluer les améliorations apportées par le processus de réinjection de pertinence on a défini la valeur absolue d'amélioration (AA ou AI (*Absolute Improvement*)) calculée comme suit :

$$Me(RF_{run}) - Me(base_{run}) \quad (2.19)$$

et l'amélioration relative (AR ou RI (*Relative Improvement*)) calculée comme suit :

$$Me(RF_{run}) - Me(base_{run}) / Me(RF_{run}) \quad (2.20)$$

où $Me(RF_{run})$ (resp. $Me(Base_{run})$) est la mesure considérée pour les résultats après réinjection (resp. des résultats de base).

En 2006, seule la fonction généralisée d'agrégation est considérée.

2.6 Conclusion

La reformulation de requêtes est une phase importante du processus de recherche d'information. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin. Cette technique peut être appliquée automatiquement ou d'une façon interactive, c'est à dire avec l'intervention de l'utilisateur.

La nature des documents semi-structurés ainsi que les requêtes a conduit à de nouvelles problématiques spécifiques à la reformulation de requêtes dans la recherche d'information structurée.

Les approches proposées dans ce contexte se divisent en deux principaux types :

les approches orientées contenu dont le but est d'enrichir le contenu des requêtes en y ajoutant des termes pertinents comme en RI classique et les approches orientées contexte qui permettent d'enrichir la requête initiale en spécifiant le contexte dans lequel apparaissent les éléments pertinents. Nous avons montré tout au long de ce chapitre que les approches proposées jusque là ne permettent pas d'aboutir à des améliorations significatives des résultats exceptée celle qui considère l'aspect structurel dans le contexte.

Plusieurs points peuvent expliquer ces résultats :

- dans les approches orientées contenu, les termes sont sélectionnés indépendamment du type des éléments pertinents pris en compte,
- les approches orientées contexte ne permettent pas de spécifier la structure des éléments recherchés,
- les deux types d'approches sont appliquées séparément, alors que la combinaison d'évidence a souvent montré son intérêt en RI.

Dans la suite du document, nous présentons notre contribution pour la réinjection de pertinence ou RIS. Nos propositions visent à répondre aux différentes problématiques de la réinjection de pertinence en recherche d'information structurée, et tentent d'apporter des solutions aux limites énoncées ci-dessus.

Deuxième partie

Nouvelles Approches pour la Reformulation de requêtes en Recherche d'Information Structurée

Chapitre 3

Reformulation de requêtes par réinjection de contenu et de structures

3.1 Introduction

La reformulation de requêtes en Recherche d'Information structurée par réinjection de pertinence ne concerne plus que les mots clés (cas de la RI classique) mais aussi d'autres sources d'évidence qui permettent de spécifier l'élément recherché. Comme nous l'avons mentionné dans le chapitre précédent, ces sources peuvent décrire le contexte des éléments pertinents (descendants, ancêtres, taille, ...).

Dans ce chapitre, nous proposons de nouvelles approches de réinjection de pertinence en utilisant différentes sources d'évidence. En effet, nous proposons d'enrichir le contenu de la requête initiale par des termes pertinents sélectionnés selon leur distribution dans les éléments pertinents et non pertinents ainsi que leur proximité vis-à-vis des termes de la requête initiale.

Une autre source d'évidence que nous allons aussi utiliser est l'information structurelle que nous traduisons par la notion de *structure pertinente*.

Nous proposons également de faire cohabiter les deux sources d'évidence contenu et structure dans une approche *combinée*.

Ce chapitre est structuré de la manière suivante. Nous présentons tout d'abord notre motivation dans la section 3.2, ensuite nous décrivons dans la section 3.3 l'approche orientée contenu. La section 3.4 est consacrée à l'approche orientée structure. Enfin, l'approche combinée sera détaillée dans la section 3.5.

3.2 Motivation

Nous avons présenté dans le chapitre précédent les différentes approches proposées pour la réinjection de pertinence dans la recherche d'information structurée. Certaines permettent d'étendre la requête au niveau du contenu et ont généralement utilisé l'algorithme de Rocchio [158] et d'autres ont proposé d'utiliser le contexte des éléments pertinents.

Nous avons également montré qu'une simple adaptation de l'algorithme de Rocchio dans le contexte de la recherche d'information structurée ne conduit pas à une amélioration significative des résultats [95], [173], [47], [132]. De ce fait, nous pensons que les évidences classiques ($tf * idf$) utilisées pour l'identification des termes pertinents et leur pondération doivent être revues dans le cas des documents semi-structurés puisqu'on traite non plus des documents entiers mais des parties des documents. Il faudrait trouver et intégrer d'autres indicateurs adéquats aux documents semi-structurés.

Dans le cas des approches orientées contexte, la réinjection de pertinence consiste à prendre en compte le contexte dans lequel apparaissent les éléments pertinents. Elles concernent précisément la prise en compte des ancêtres, des descendants, du nom du journal dans lesquels apparaissent des éléments pertinents, ainsi que leurs taille. Les résultats obtenus restent non significatifs (taux d'amélioration $< 5\%$). Seule l'approche de réinjection du contexte structurel a montré son intérêt [175]. Ces approches restent étroitement liées à un type de DTD.

De plus, la majorité des approches proposées en réinjection de pertinence, ont abordé une seule source d'évidence à la fois soit contenu soit contexte. Il existe peu voire pas d'approches ayant combiner les deux sources pour identifier des éventuelles relations de pertinences entre termes et structures. De plus, dans aucune des approches proposées on ne retrouve une modélisation des relations directes entre les données textuelles et les données structurelles.

Les objectifs de nos travaux sont alors les suivants :

- utiliser d'autres sources d'évidence, indicateurs, pour sélectionner et pondérer les termes pertinents [93],
- proposer une approche pour la réinjection de la structure [94], [88], [96],
- enfin étudier l'impact de la combinaison des deux sources d'évidence (contenu et structure) [90] [91], [92], [89] pour enrichir la requête initiale et répondre en particulier à la question de dépendance contextuelle

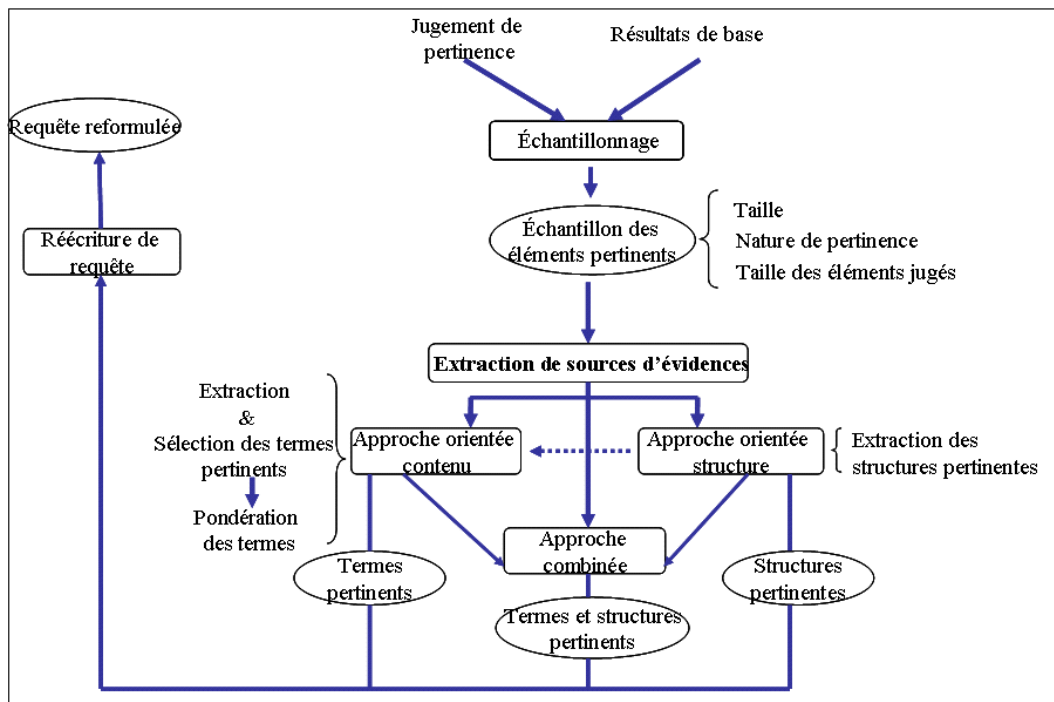


FIG. 3.1 – Mécanisme de reformulation

qui peut exister entre les termes pertinents et les structures pertinentes.

Nos investigations concernent donc plusieurs points comme ceci est résumé dans la figure 3.1. Le point de départ des approches que nous proposons est évidemment la liste des éléments jugés pertinents par l'utilisateur. En outre, la démarche que nous adoptons pour ce processus de reformulation de requête est composée essentiellement de 3 étapes :

1. La première étape : **Echantillonnage**. Cette étape est commune à toutes les approches proposées, elle consiste à construire un échantillon d'éléments à partir des résultats de base et des jugements de pertinence. Un échantillon est caractérisé par sa taille, le nombre d'éléments pertinents qu'il contient et la nature de la pertinence considérée (degré d'exhaustivité et de spécificité). Nous n'avons pas effectué d'investigations ni de propositions théoriques particulières dans cette étape. C'est une étape qui a été investie principalement lors de l'expérimentation de nos approches. Nous discuterons donc dans le chapitre consacré à l'expérimentation les critères à considérer pour la construction d'un échantillon.
2. La seconde étape : **Extraction** des sources d'évidence. Dans cette étape, nous proposons trois types d'approches pour extraire des informations à partir de l'échantillon défini dans l'étape précédente :

- la première orientée contenu dans laquelle on enrichit le contenu de la requête par des termes pertinents,
 - la seconde permet d’extraire des structures pertinentes,
 - la troisième combine les deux sources : termes pertinents et structures pertinentes.
3. La dernière étape : **Réécriture de la requête**. Cette étape dépend de l’approche utilisée ainsi que du type de requêtes (structurées et non structurées). Elle permet de réinjecter les données sélectionnées de l’étape précédente pour aboutir à une nouvelle requête qui peut être exécutée par le système de recherche.

Comme nous l’avons mentionné ci-dessus, les étapes que nous venons de décrire concernent les 3 approches proposées : orientée contenu, orientée structure et combinée. Ainsi, au lieu de présenter chacune de ces étapes d’une manière séparée, nous avons préféré les présenter dans chacune des approches.

3.3 Approche orientée Contenu

Notre approche orientée contenu se déroule en trois étapes : l’extraction et la sélection des termes pertinents, la pondération des termes de la requête et enfin la réécriture de la requête.

3.3.1 Extraction et Sélection des termes pertinents

Nous avons abordé le problème d’extraction de termes en utilisant différents indicateurs de pertinence. Ceci a conduit à trois stratégies. Nous proposons tout d’abord une stratégie de base utilisant uniquement la distribution des termes dans les éléments pertinents, puis les stratégies intégrant d’autres sources notamment le contexte des termes et l’information issue de la pertinence négative.

3.3.1.1 Stratégie de base : Sélection par probabilité de pertinence

Nous avons assimilé le problème d’extraction des termes pertinents à un problème probabiliste. En effet, l’extraction des termes pertinents est conditionnée par leur présence dans les éléments jugés pertinents. L’idée derrière cette approche est que plus un terme figure dans des éléments jugés pertinents plus sa probabilité de pertinence, c’est à dire sa capacité d’exprimer le besoin de l’utilisateur, est importante.

Une manière simple de mesurer cette probabilité est d'utiliser le maximum de vraisemblance. On considère alors la pertinence d'un terme t_j comme un événement probabiliste. Ceci est une traduction simple de la formule de Robertson [156]. Ainsi, la probabilité qu'un terme soit pertinent pour une requête est définie comme suit :

$$P(t_j/R) = |p_{ej}|/|E^p| \quad (3.1)$$

où

R est l'évènement pertinence. $|p_{ej}|$ est le nombre d'éléments pertinents dans lesquels apparaît le terme t_j et $|E^p|$ est la taille de l'ensemble le nombre d'éléments pertinents (E^p).

Cette équation attribue des probabilités indépendamment des fréquences des termes dans les éléments pertinents. De ce fait, tous les termes des éléments jugés pertinents auront un poids > 0 . Cette formule présente des limites dans le cas où les termes occurrent dans le même nombre d'éléments pertinents mais avec des fréquences différentes. Dans ce cas les termes peuvent être de degrés d'importance différents alors qu'ils ont le même poids.

Exemple :

si considère un terme t_1 qui occure dans 2 éléments jugés pertinents dans un ensemble composé de 3 éléments pertinents, avec des fréquences 3 et 5 et un terme t_2 qui occure également dans 2 éléments jugés pertinents avec des fréquences 1 et 2, bien qu'il nous parait que le terme t_1 est plus représentant de l'élément pertinent que le terme t_2 , ces derniers ont le même poids = 0,66.

Nous essayons alors d'affiner le choix des termes pertinents dans l'étape suivante : sélection basée sur le contexte des termes pertinents.

3.3.1.2 Stratégie basée sur le contexte

Comme nous l'avons signalé, l'extraction des termes en considérant uniquement leurs distribution dans les éléments pertinents est insuffisante pour déterminer leur degré de pertinence. En effet, les valeurs de probabilité calculées ne sont pas assez discriminantes. Comme la pertinence est définie selon deux dimensions l'exhaustivité et la spécificité, notre problème revient alors à trouver les termes qui décrivent des éléments à la fois spécifiques et exhaustifs. La notion d'exhaustivité est traduite par la distribution des termes dans les éléments pertinents que nous avons vue dans la stratégie de base. La spécificité peut être traduite en considérant les termes proches de ceux de la requête. En d'autres termes, si un terme se trouve souvent aux alentours des termes de la requête, il y a une forte chance que ce terme soit lié sémantiquement

à ceux de la requête. Intuitivement, ce terme pourrait être un bon candidat pour l'enrichissement de la requête initiale. Une manière simple de traduire cette proximité surfacique entre les termes des éléments pertinents et ceux de la requête est d'utiliser la notion de *contexte* d'un terme. Cette notion a été préalablement utilisée en RI classique [163], nous l'avons adapté à notre contexte. C'est une mesure qui tient compte des termes de la requête pour pondérer les termes extraits d'un élément jugé pertinent. Elle est basée sur les distances entre les termes d'un élément et ceux de la requête. Elle est définie dans l'équation suivante :

$$\text{context}^{e_i}(t_j) = (\text{distribution}^{e_i}(q) - \text{min}^{e_i}(t_j)) / \text{distribution}^{e_i}(q) \quad (3.2)$$

$$\text{min}^{e_i}(t_j) = \min_{t_j \neq t_k} |(\text{position}^{e_i}(t_j) - \text{position}^{e_i}(t_k))| \quad (3.3)$$

$$\text{distribution}^{e_i}(q) = \text{length}(e_i) / \text{occurrences}^{e_i}(q) \quad (3.4)$$

où

$\text{distribution}^{e_i}(q)$ est la distribution de tous les termes de la requête dans l'élément e_i , avec $\text{length}(e_i)$ la taille de l'élément e_i moins les termes de la requête et $\text{occurrence}^{e_i}(q)$ le nombre d'occurrences des termes de la requête q dans l'élément e_i

$\text{min}_{e_i}(t_j)$ est la différence minimale de positions entre n'importe quelle occurrence du terme t_j et un autre terme t_k de la requête, avec $\text{position}^{e_i}(t_j)$ la position du terme t_j dans e_i .

Cette notion permet de mesurer le degré d'appartenance d'un terme au contexte d'une requête donnée. Elle sert à mettre en valeur les termes exprimant à la fois l'exhaustivité et la spécificité. En effet, la formule peut être interprétée de deux manières différentes :

1. Une interprétation directe de cette mesure permet de constater qu'on obtient un contexte élevé pour un terme t_j dans le cas où $\text{min}^{e_i}(t_j)$ (distance entre t_j et le terme t_k de la requête) est faible (i.e $\text{min}^{e_i}(t_j)$ tend vers 0). Ce qui traduit la spécificité du terme par rapport à la requête.

Exemple :

Un utilisateur exprime le besoin suivant :

” **recherche d'information** ”

Soit l'élément suivant jugé pertinent :

” Abrégée en RI ou IR (Information Retrieval en anglais), la **recherche d'information** est la science qui consiste à **rechercher l'information** dans des documents - les documents eux-mêmes ou les méta données qui décrivent les documents ..., dans des bases de données -L'informatique a permis le

développement d'outils pour traiter l'**information** et établir la représentation des documents au moment de leur indexation, ainsi que pour **rechercher l'information ...**"

La distribution de la requête est le nombre des termes de la requête figurant dans le paragraphe, ici : 7 par lequel on divise la taille du paragraphe excepté des termes de la requête soit 28, pour faciliter les calculs (on ne considère pas les prépositions, les propositions, les articles, etc.). Le minimum est quant à lui égal à 1. Si on calcule le score du terme "document" par exemple :

$$\text{Context}(t_j) = (4 - 1)/4 = 0.75.$$

2. Une deuxième interprétation vient du fait que si le $\min^{ei}(t_j)$ est faible, le contexte reste faible tant que $\text{distribution}^{ei}(q)$ est faible (i.e. se rapproche de la valeur du $\min^{ei}(t_j)$). Ceci traduit le cas où les termes de l'élément considéré sont en majorité ceux de la requête. Les termes de cet élément n'appartenant pas à la requête ne représentent pas en général une description de l'information recherchée et par conséquent, ils ne peuvent pas être efficaces pour pointer sur des éléments exhaustifs.

Exemple : soit la même requête que précédemment :

"recherche d'information"

Si on considère l'élément suivant :

" Association francophone en **Recherche d'Information** et Applications (ARIA) "

Cet élément bien qu'il ne réponde pas directement au besoin de l'utilisateur, est jugé pertinent puisqu'il renseigne sur l'association du domaine sans présenter aucune information sur le processus de la RI.

Si on calcule le score du terme "francophone" :

la distribution de la requête dans cet élément est $3/2=1.5$, le minimum =1.

$$\text{Context}(t_j) = (1.5 - 1)/1.5 = 0.33.$$

On remarque que ce terme a un poids bien inférieur à celui du terme "document" dans l'exemple précédent ($0.33 < 0.75$). En effet, il appartient un élément caractérisé par une faible distribution des termes de la requête initiale.

En conclusion la mesure contexte répond à nos besoins pour exprimer la pertinence d'un terme. Nous allons alors combiner le poids déjà calculé par la probabilité conditionnelle avec le contexte du terme calculé dans l'ensemble des éléments pertinents. Le poids d'un terme appelé Poids Contextuel (PC) sera calculé suivant l'équation 3.5 suivante combinant la probabilité et le contexte :

$$PC(t_j) = P(t_j/E^p) \times \sum_{i=1}^{|p_{ej}|} context_j^{e_i} \quad (3.5)$$

où

$|p_{ej}|$ est l'ensemble des éléments pertinents contenant le terme t_j .

Cette équation conçue pour la sélection de termes pertinents en considérant la réinjection de pertinence positive. Dans le prochain paragraphe, nous proposons de considérer la pertinence négative pour l'extraction et la sélection des évidences.

3.3.1.3 Prise en compte de la pertinence négative

Comme nous l'avons déjà mentionné dans le chapitre précédent, la reformulation de requête peut également prendre en compte des éléments jugés non pertinents [158].

D'une manière générale, l'effet de la réinjection négative est de diminuer l'importance des termes qui ont un effet négatif sur la recherche ou de les supprimer. Ces termes sont extraits des éléments jugés non pertinents.

D'après Ruthven *et al.* [161], les éléments jugés non pertinents ne sont pas bien définis, ce qui explique le fait que la réinjection négative n'a pas d'effet important, en RI en termes de performances (i.e. dans Rocchio, le coefficient γ de la réinjection négative est beaucoup plus faible que le coefficient β de la réinjection positive).

Intuitivement, nous croyons que la prise en compte de la réinjection de pertinence négative permettrait un meilleur taux de précision. Nous proposons pour cela de calculer pour chaque terme un facteur bruit [168]. Un terme présente du bruit s'il occure autant de fois dans les éléments pertinents que dans les éléments non pertinents. Ce facteur bruit est calculé comme suit :

$$Bruit(t_j) = \sum_{i=1}^{np} \frac{tf_j^{e_i}}{ttf_j} \log \frac{ttf_j}{tf_j^{e_i}} \quad (3.6)$$

où

$tf_j^{e_i}$ est la fréquence du terme t_j dans l'élément e_i , np est le nombre d'éléments non pertinents et ttf_j est la fréquence totale du terme t_j dans les éléments non pertinents.

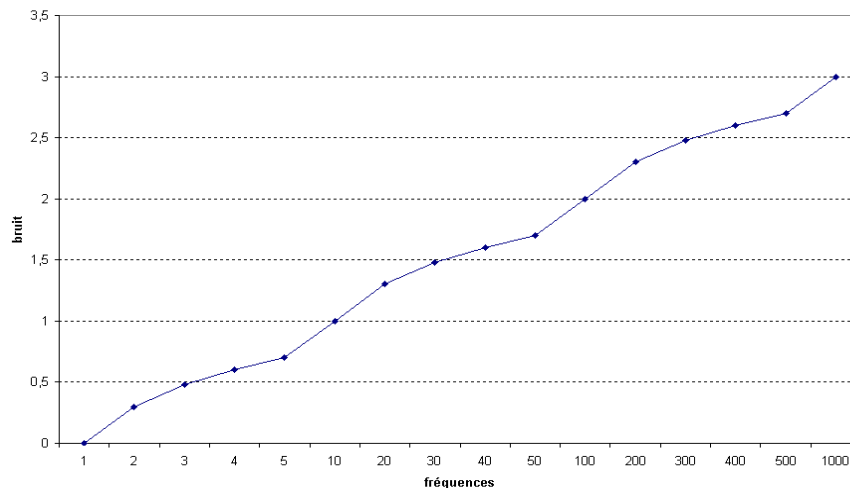


FIG. 3.2 – Variation du bruit en fonction de fréquences

Le $Bruit(t_j)$ défini dans [168] est basé sur la fréquence du terme dans le document. Dans notre cas nous considérons les fréquences des termes au niveau des éléments jugés non pertinents.

Si on étudie la fonction de bruit de plus près, on constate qu'elle ne peut être nulle que s'il s'agit d'un cas particulier. $Bruit(t_j) = 0$ correspond au cas où le terme n'apparaît que dans un seul élément ($ttf_j = tf_j^e$). Le reste des cas correspond à des valeurs qui sont soit entre 0 et 1 soit supérieures à 1. Le premier cas ($0 < Bruit(t_j) < 1$), correspond à une distribution variée dans les différents éléments jugés non pertinents : c'est le cas où le terme présente relativement du bruit.

Le deuxième cas est celui où le terme occure d'une manière régulière dans tous les éléments jugés non pertinents. Les variations du bruit en fonction des fréquences sont illustrées dans la figure 3.2 : plus le nombre d'éléments non pertinents dans lequel apparaît un terme augmente, plus le bruit augmente. Il est au dessus de 1 ce qui correspond à un terme bruité. Remarquons que cette fonction ne peut pas avoir des valeurs strictement négatives puisque $ttf_j \geq tf_j^e$. Selon les valeurs possibles du bruit, ce facteur permet de discriminer davantage les poids des termes pour la sélection.

Exemples :

1. soit un terme t_1 qui occure dans 3 éléments jugés non pertinents avec les fréquences suivantes : 2, 4 et 10. Le bruit est calculé comme suit :

$$bruit(t_1) = 2/16 * \log(16/2) + 4/16 * \log(16/4) + 10/16 * \log(16/10) = 0.38$$

2. soit un terme t_2 qui occure dans 20 éléments jugés non pertinents avec une même fréquence égale à 5. Le bruit est calculé comme suit :

$$\text{bruit}(t_1) = 20 * 5/100 * \log(100/5) = 1.3$$

Nous proposons alors de diminuer les poids des termes qui présentent un bruit élevé (c'est à dire $\text{bruit} > 1$) et d'augmenter ceux des termes moins bruités (ayant un bruit entre 0 et 1).

Nous proposons alors d'intégrer le bruit dans le poids contextuel. Ainsi le poids ajusté (PA) combinant le facteur bruit et le poids des termes sélectionnés selon l'équation 3.5. Le poids ajusté permet de prendre en compte à la fois de la pertinence positive exprimée à travers le poids contextuel et la pertinence négative à travers le facteur bruit.

$$PA(t_j) = (P_nC(t_i))^{\text{Bruit}(t_j)} \quad (3.7)$$

Nous avons appliqué cette fonction en puissance des poids contextuels déjà calculés normalisés $P_nC(t_j)$ dans l'intervalle $[0, 1]$. La normalisation est effectuée de la manière suivante :

$$P_nC(t_j) = \frac{PC(t_j)}{\max_{t_i \in TS}(PC(t_i))} \quad (3.8)$$

Où

$PC(t_j)$ le poids contextuel du terme t_j (équation 3.5), $P_nC(t_j)$ le poids contextuel normalisé du terme t_j , TS l'ensemble des termes sélectionnés.

Ceci nous conduit à un poids maximal si le terme n'apparaît que dans un seul élément non pertinent ou dans aucun élément, le score final est égal à 1. Un terme bruité ($\text{bruit} > 1$) aura un poids final inférieur à celui déjà calculé. Pour le cas où les valeurs se situent entre 0 et 1, les poids seront élevés tout en restant inférieurs à 1.

Les deux équations conçues pour la sélection de termes pertinents (poids contextuel et poids ajusté), peuvent servir également à la pondération des termes de la requête reformulée. Par ailleurs, d'autres sources d'évidence peuvent aussi être utilisées pour la pondération des termes de la nouvelle requête. Ceci fait l'objet de la section suivante.

3.3.2 Pondération des termes de la requête

La pondération concerne d'une part les termes sélectionnés et d'autre part les termes de la requête initiale. Nous proposons deux solutions différentes :

- La première consiste à pondérer les termes de la requête reformulée selon le poids contextuel (avec ou sans prise en compte du bruit).
- Dans la seconde hypothèse, nous séparons la phase de la sélection de celle de la pondération. En fait, les termes de la requête finale sont sélectionnés selon le poids contextuel, puis il sont pondérés en utilisant une formule de type *tf.idf.ief* que nous avons proposée dans le modèle XFIRM que nous dériverons dans le chapitre 4. Cette formule reflète l'importance d'un terme dans les éléments ainsi que dans les documents de la collection. Ainsi les termes de la requête, y compris les termes originaux, seront pondérés selon l'équation suivante :

$$Pds(t_j) = tf(t_j) \times idf(t_j) \times ief(t_j) \quad (3.9)$$

Nous avons alors considéré la fréquence du terme dans la collection $tf(t_j)$. Pour conserver la dualité de la pertinence (exhaustivité et spécificité), le facteur $tf(t_j)$ est ainsi multiplié par le facteur idf , ainsi que par son analogue ief défini pour exprimer la spécificité d'un terme dans l'ensemble des éléments de la collection :

$$ief(t_j) = \log \left(\frac{|E|}{|e_j|} + 1 \right) \quad (3.10)$$

Avec $|e_j|$ le nombre d'éléments dans lequel occure le terme t_j et $|E|$ le nombre d'éléments dans la collection.

Dans les deux solutions proposées, les poids attribués aux termes à réinjecter seront normalisés entre 0 et 1.

3.3.3 Réécriture de la requête

La réécriture de la requête est l'étape finale de la reformulation, elle permet de mettre en place la requête qui sera transmise au système de recherche. La question est comment prendre en compte les termes de la requête initiale Q dans la réécriture de la nouvelle requête Q' ?

D'une manière générale, la nouvelle requête est formulée comme suit :

$$Q' = \alpha.Q + \beta.TS \quad (3.11)$$

où :

Q est la requête initiale composée d'un ensemble de k couples (t_j, w_{jq}) , $j \in [1, k]$, w_{jq} est le poids du terme t_j de la requête initiale. TS représente l'ensemble des couples (t_j, w_{jTS}) terme pertinent t_j

associé à son poids w_{jTS} , triés selon leurs poids. Les poids sont calculés selon l'équation 3.5, l'équation 3.7 ou l'équation 3.9

Une alternative possible de la réécriture 3.11 est de rajouter uniquement à la requête initiale les nouveaux termes :

$$Q' = Q + TS' \quad (3.12)$$

avec

$$TS' = \{TS - Q\}$$

Exemples : requête orientée contenu

Nous considérons la requête 202 de la collection de test INEX 2005 :

"ontologies case study"

Nous supposons que nous sélectionnons 3 termes pertinents dont un existe déjà dans la requête. Les termes sélectionnés avec leurs poids associés sont les suivants : (*graph*, 1) (*concept*, 0.6) et (*ontology*, 0.8) Nous supposons que les termes initiaux de la requête ont un poids égale à 1.

La requête finale sera alors comme suit :

"ontologies,1 case,1 study,1 graph,1 concept,0.6 ontology,0.8"

avec $\alpha = \beta = 1$

Si on considère la seconde alternative, la requête finale sera comme suit où le mot-clé "ontology" apparaît une seule fois :

"ontologies,1 case,1 study,1 graph,1 concept,0.6"

Exemples : requête structurée

Ce type de requête est composée de sous requêtes dont chacune représente une contrainte structurelle et un ensemble de mots clés. Dans notre approche pour éviter d'une part les redondance dans la requête et pour simplifier d'autre part la recherche, nous allons procéder de manière à enrichir la sous requête cible. La sous requête cible est explicitement identifiée par le terme *ec* (utilisé dans notre système XFIRM décrit dans le chapitre 4). Le terme *ec* désigne l'élément désiré par l'utilisateur. Nous appliquons alors le même principe que précédemment. Les termes appartenant aux autres sous requêtes auront une pondération maximale =1.

Nous considérons comme exemple la requête 202 de la collection de test INEX 2005 :

" article[ontologies] // ec : sec[ontologies case study] "

On cherche une section sur "ontologies case study" descendante d'un article sur les "ontologies".

Si on considère les termes de l'exemple précédent à réinjecter, les nouvelles requêtes seront comme suit : seule la sous requête cible (*sec[ontologies case study]*) sera modifiée.

" article[ontologies,1] // ec : sec[ontologies,1 case,1 study,1 graph,1 concept,0.6 ontology,0.8] "

avec $\alpha = \beta = 1$.

Selon la seconde alternative, la requête finale sera comme suit :

" article[ontologies,1] // ec : sec[ontologies,1 case,1 study,1 graph,1 concept,0.6] "

3.3.4 Conclusion

L'approche que nous avons proposée pour extraire les termes pertinents à partir des éléments pertinents est différente des approches proposées dans la littérature. Cette différence se situe dans la stratégie proposée pour extraire les termes pertinents. Cette stratégie combine plusieurs indicateurs : la distribution des termes dans les éléments pertinents, la proximité contextuelle de ces termes vis à vis des termes de la requête initiale et enfin le bruit qu'ils peuvent engendrer selon leur présence dans les éléments non pertinents.

Nous allons introduire la seconde source d'évidence dans la section suivante.

3.4 Réinjection de la structure

Le processus de réinjection de pertinence que nous avons étudié tout au long des chapitres précédents concerne principalement l'ajout des termes dans les requêtes. Notre objectif dans cette section est d'étudier l'intérêt de reformuler une requête en réinjectant une contrainte structurelle. L'intuition que nous avons derrière cette démarche est la suivante :

nous pensons que les informations pertinentes recherchées par un utilisateur ont de fortes chances de se retrouver dans des éléments de même type (même type

de balise). L'idée est alors d'arriver à identifier ces balises à partir des éléments jugés pertinents par l'utilisateur puis reconstruire une nouvelle requête en y injectant ces balises sous forme de contraintes structurelles.

Pour simplifier, nous allons commencer par introduire la notion de structure pertinente. Avant de rentrer dans le détail de l'approche et afin de lever toutes les ambiguïtés dans notre discours, nous donnons quelques définitions utiles pour la suite.

Définitions :

Nous rappelons brièvement les notions de l'élément et de chemins :

- Un *élément* toute partie qui représente un sous-arbre de l'arbre d'un document XML. Un élément est représenté par un nœud. Un nœud est caractérisé par le nom d'une balise.
- Le *chemin* (*path*) de l'élément est l'ensemble des nœuds séparant le nœud de l'élément de la racine.

Nous considérons une structure comme une forme simplifiée du chemin, composée d'un ensemble de balises.

La *distance* entre deux balises d'une même structure est le nombre de balises qui les séparent. Si nous considérons une structure S (composée de n balises) : $S = B_1/B_2/.../B_n$, la distance (d) entre la balise B_i et la balise B_n est calculée comme suit :

$$d(B_n, B_i) = (n - i), n \geq i$$

Une **séquence** de balises est une partie d'une structure.

3.4.1 La notion de structure pertinente

La première question qui se pose dans cette approche concerne l'existence même de cette notion de **structure pertinente**, c'est-à-dire celle susceptible de contenir des informations pertinentes, et qu'est ce qu'elle représente exactement.

Pour répondre à cette question, nous avons analysé les collections de test de INEX 2005 et INEX 2006. Cette analyse consiste à regarder de près la nature des réponses pertinentes à une requête donnée.

En particulier, nous avons compté le nombre de types de structures dans lesquelles peuvent se retrouver les éléments pertinents pour une requête donnée. Les types de structures correspondent dans ce cas à la dernière balise comme par exemple la balise p de la structure $/article/sec/ss1/p$. Nous nous sommes

servis principalement des jugements de pertinence fournis par la campagne d'évaluation pour chaque requête. Nous n'avons considéré que les éléments strictement pertinents puisque le but est de fixer les besoins de l'utilisateur en structures.

Le tableau 3.1 présente pour chacune des collections 2005 (composée de 28 requêtes jugée) et 2006 (composée de 114 requêtes jugées), la moyenne du nombre d'éléments strictement pertinents sur l'ensemble des requêtes (MEP), la moyenne du nombre de balises distinctes dans lesquelles se trouvent les éléments pertinents (MSEP) et le nombre total de balises différentes des éléments pertinents (NSEP) pour toutes les requêtes. D'après ce tableau pour une requêtes

TAB. 3.1 – Propriétés des jugement de pertinence

	MEP	MSEP	NSEP
INEX 2005	31.22	4.67	27
INEX 2006	323.86	8.3	37

donnée de la collection 2005, la moyenne d'éléments pertinents est de 31.22. Ces éléments ont en moyenne 4.67 structures différentes parmi les 27 de la collection. Dans le cas de la collection 2006, la moyenne d'éléments pertinents pour une requête donnée est de 323.86. Ces éléments ont en moyenne 8.3 structures différentes parmi les 37 de la collection. Nous avons ensuite compté pour chaque balise le nombre de fois qu'elle apparaît dans les éléments pertinents ($\#(balise_i)q$). Ces balises sont ensuite triées par ordre décroissant de ce nombre ($\#(balise_i)q$).

Afin de mieux rendre compte de ces nombres, le tableau 3.2 liste le ratio (%) entre ($\#(balise_i)q$) et le nombre total d'éléments pertinents.

Dans a colonne *1 struct*, on ne considère qu'une seule balise, *2 struct*. On considère les 2 premières pour une requête et *3 struct*, les trois premières, etc. Nous avons calculé une moyenne, un min et un max sur l'ensemble des requêtes pour les 2 collections considérées.

TAB. 3.2 – Répartition des éléments pertinents en fonction des types de structures - INEX 2005-2006

	1 struct	2 struct	3 struct	4 struct	5 struct
2005					
Moyenne	64%	85%	93%	96%	98%
Min	23%	44%	60%	73%	89%
Max	100%	100%	100%	100%	100%
2006					
Moyenne	70%	84%	91%	95%	97%
Min	31%	56%	73%	84%	89%
Max	96%	100%	100%	100%	100%

Nous constatons que quelle que soit la collection (INEX 2005 ou INEX 2006) les éléments se partitionnent généralement dans un ensemble bien défini de type

de structures. Nous notons en examinant la moyenne, que l'on arrive à plus de 90% des éléments pertinents (93% pour la collection INEX 2005 et 91% pour le collection de INEX 2006) en considérant trois balises pour une requête donnée, sachant que le nombre total des types de structures (balises) caractérisant des éléments pertinents est respectivement 27 et 37 dans les collections 2005 et 2006.

Nous pouvons donc conclure qu'il existe bien des structures pertinentes pour chaque requête. Nous définissons par la suite la notion de structure générique qui traduit la pertinence des structures.

3.4.2 Extraction de la structure pertinente

Une structure pertinente est une structure dans laquelle on retrouve des informations à la fois exhaustives et spécifiques. Nous définissons tout d'abord le concept de structure générique comme suit :

On appelle *structure générique* une structure qui peut être commune à un grand nombre d'éléments pertinents.

Exemple :

Si l'on considère que pour une requête donnée, nous avons 3 éléments jugés pertinents ayant les structures S_k , S_l et S_m suivantes :

$$\begin{aligned} S_k & /article/bdy/sec/ss1, \\ S_l & /article/bdy/sec/ss1/ss2 \text{ et} \\ S_m & /article/bdy. \end{aligned}$$

On remarque que S_m est une structure commune aux deux autres structures. Si on considère ces structures sous forme d'un arbre, la distance entre les 2 structures S_m et S_l est de 3.

Pour extraire alors la structure générique, nous allons procéder de manière à retrouver une structure qui représente une branche commune entre la majorité des structures auxquelles appartiennent les éléments pertinents tout en tenant compte des distances entre les structures.

Nous revenons brièvement sur cette notion de distance définie dans la section précédente. Le tableau 3.3 liste les différentes distances entre les structures de l'exemple. On constate que la structure S_k a la somme des distances la séparant des autres structures, la moins élevée =3 par rapport aux sommes obtenues par

TAB. 3.3 – Récapitulation des différences de distance entre les structures

	S_k	S_l	S_m	Somme
S_k	-	1	2	3
S_l	1	-	3	4
S_m	2	3	-	5

S_m (4) et S_l (5) les structures S_m et S_l . La structure générique dans ce cas est S_k .

Nous proposons dans ce qui suit l’algorithme d’extraction des structures génériques appelé SCA (*Smallest Common Ancestor*).

Plusieurs approches en RI structurée orientées bases de données ont utilisé la notion d’ancêtre communs pour répondre à la fois aux contraintes structurelles et textuelles. Nous présentons dans ce qu’il suit quelques algorithmes de recherche d’ancêtre commun ainsi notre algorithme d’extraction des structures génériques dans le cas des documents homogènes, ayant une même DTD.

3.4.3 Extraction de structures pertinentes dans des documents homogènes

3.4.3.1 Algorithmes de recherche des ancêtres communs

Il existe une panoplie d’algorithmes permettant la recherche des ancêtres communs on y trouve en particulier :

- L’algorithme LCA : Les auteurs de [177] ont proposé l’algorithme LCA (*Lowest Common Ancestor*) pour la recherche dans les documents XML par mots clés. Cet algorithme permet de sélectionner le plus petit sous arbre contenant tous les mots clés. Le LCA est utilisé pour la reconstruction des B-Arbres [205]. On retrouve d’autres dérivés de cet algorithme comme le SLCA et le MLCAS. La recherche selon LCA est assez stricte : toutes les contraintes doivent être satisfaites ce qui ne convient pas à la définition de structure générique : elle peut ne représenter un tronçon commun avec certaines structures.
- SLCA : La notion de SLCA (*Smallest Lowest Common Ancestor*) a été proposée par Xu et Papakonstantinou dans [209] pour pallier le problème de redondance des sous-arbres, de LCA. La fonction SLCA permet au système de recherche et de ne retourner que le nœud le plus spécifique et non pas un ensemble des nœuds redondants. Les auteurs ont alors im-

planté deux algorithmes de recherche basés sur SLCA : *Indexed Lookup Eager* (appliqué dans la cas où la fréquence des mots clés varie significativement) et *Scan Eager* (dans le cas contraire) testés dans le système de recherche X-KSearch (XML Keywords Search).

Une autre extension a été proposée par Sun *et al.* de [189] (*Multiway SLCA*) pour répondre aux requêtes comportant des opérateurs booléens de type OR et AND.

- Le MLCA (*Meaningful Lowest Common Ancestor*) [123] est lui aussi un dérivé de LCA. Le MLCA est un plus petit nœud commun de deux autres nœuds de deux types différents. MLCA ne peut pas avoir un descendant pour lequel il peut être un ancêtre commun avec les deux types de nœuds. Le MLCAS est un dérivé de la fonction MLCA qui ne renvoie que les nœuds répondant aux contraintes avec leur plus petit ancêtre commun MLCA. En d'autres termes, c'est une structure qui ne comporte pas d'informations inutiles pour la requête.

Ces différents algorithmes sont appliqués au niveau d'un seul arbre d'un document XML. Leur but est d'extraire le plus petit ancêtre commun qui satisfait toutes les contraintes textuelles et structurelles. Nous nous intéressons dans notre cas aux structures qui satisfont le maximum, pas forcément tous les éléments jugés pertinents (dans ce cas on parle de l'ancêtre commun qui peut refléter l'exhaustivité de l'information) mais qui ne perdent pas l'aspect spécifique renseigné par la structure des éléments.

Nous allons en fait extraire un ancêtre commun d'une manière plus flexible grâce à un nouveau algorithme appelé SCA (Smallest Common Ancestor) et nous appelons cet ancêtre une structure générique.

3.4.3.2 L'algorithme SCA (Smallest Common Ancestor)

Nous considérons les paramètres suivants :

- E^p l'ensemble des éléments pertinents jugés par l'utilisateur,
- e_i^p le i^{eme} élément pertinent $\in E^p$,
- e_i^p est caractérisé par un chemin XPath simplifié c_i (exemple : `/article/bdy/section`) et un poids w_i (initialisé à 1 au début de l'algorithme),
- $c.first$ et $c.last$ respectivement la première et la dernière balise du chemin c ,
- $head(c)$ une fonction permettant de réduire le chemin c en lui attribuant celui du parent (c.à.d. supprimant la dernière balise de la structure). Par exemple, $head(/article/bdy/section) = /article/bdy$.

Notre algorithme (voir tableau 3.4) consiste à comparer la structure de chaque élément pertinent avec le reste des structures des éléments jugés pertinents. Pour chaque $(e_i^p, e_j^p)_{i \neq j} \in E^p \times E^p$, nous appliquons l'algorithme *SCA* qui permet d'extraire le chemin du plus petit ancêtre commun entre e_i^p et e_j^p . Le chemin sera par la suite ajouté à un ensemble des Structures Communes noté *SC*.

<pre> SCA(e_i^p, e_j^p) Début $e_i^p(c_i, w_i); e_j^p(c_j, w_j)$ $SC = \emptyset$ si $c_i.first = c_j.first$, alors si $c_i.last = c_j.last$, alors si $\exists e_k^p(c_k, w_k) \in SC$ $c_k = c_i$ alors $w_k \leftarrow w_k + w_j$ sinon $w_i \leftarrow w_i + w_j$ $SC \leftarrow c_i$ sinon si $head(c_j) \neq null$, alors $c_j' \leftarrow head(c_j)$ $w_j' \leftarrow w_j/2$ $SCA(e_i^p(c_i, w_i), e_j^p(c_j', w_j'))$ sinon $SCA(e_j^p, e_i^p)$ Fin </pre>

TAB. 3.4 – Algorithme d'extraction de la structure générique.

La structure générique choisie est celle ayant le score le plus élevé.

3.4.3.3 Exemple d'application de l'algorithme SCA

On considère pour une requête donnée trois éléments jugés pertinents e_1^r , e_2^r et e_3^r auxquels correspondent les structures (nous traitons dans ce cas les structures comme des chemins) :

S_1 /article/bdy/sec/ss1
 S_2 /article/bdy/sec/ss1/ss2
 S_3 /article/bdy.

Nous décrivons à travers les figures suivantes les différentes étapes de la recherche d'une structure générique. On affecte un poids unique à toutes les structures sont w_1 , w_2 et w_3 . Dans notre application, ce poids est une constante,

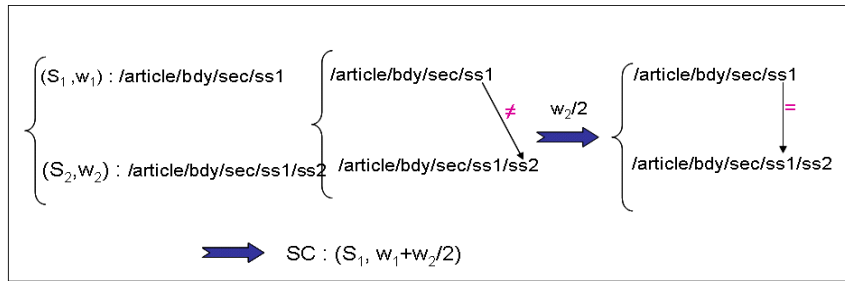


FIG. 3.3 – Recherche d’une structure générique :A

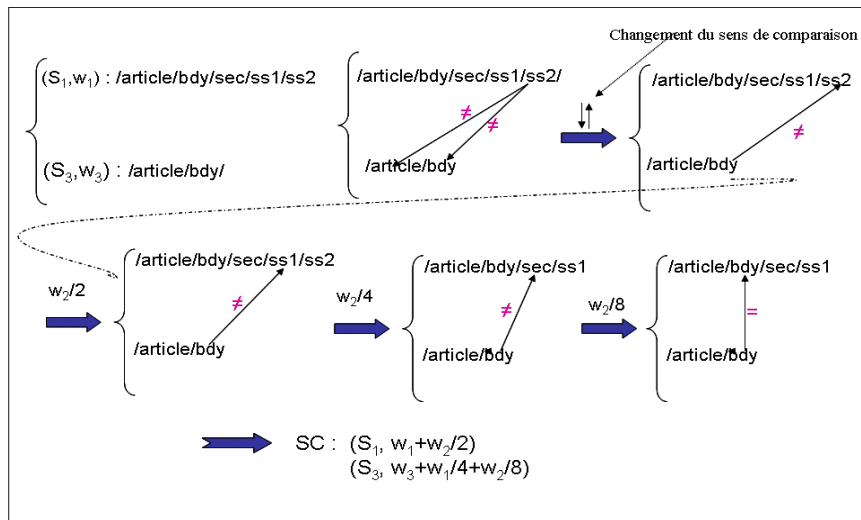


FIG. 3.4 – Recherche d’une structure générique : C

que l’on pourra par exemple prendre égale à 1. Soit l’ensemble SC initialement vide dans lequel on rajoutera les structures génériques.

Nous considérons que la comparaison des premières balises comme étape 0 de l’algorithme.

La figure 3.3 présente la première étape de notre recherche de structure générique ; elle consiste à comparer la dernière balise de la structure S_1 à la dernière balise de la structure S_2 . Dans ce cas les balises sont différentes, donc on passe au niveau supérieur de la structure S_2 dont le score devient $w_2/2$. La dernière balise devient $ss1$ ce qui correspond à la dernière balise de S_1 . Par conséquent, la structure S_1 sera ajoutée à l’ensemble SC avec le score $w_1 + w_2/2$

On compare ensuite (Figure 3.5) les deux structures S_1 et S_3 . La comparaison de la dernière balise de S_1 avec les balises de la structure S_3 n’aboutit à aucun résultat (figure 3.5), on passe au ”matching” dans le sens inverse.

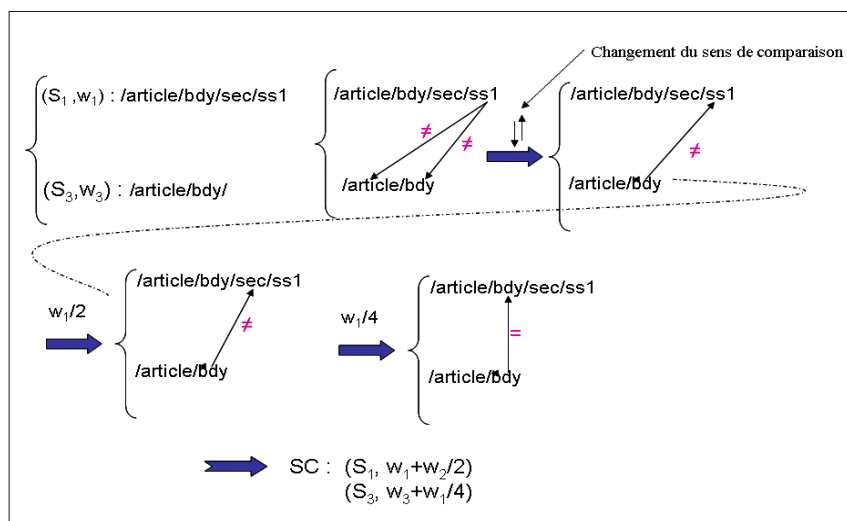


FIG. 3.5 – Recherche d’une structure générique : B

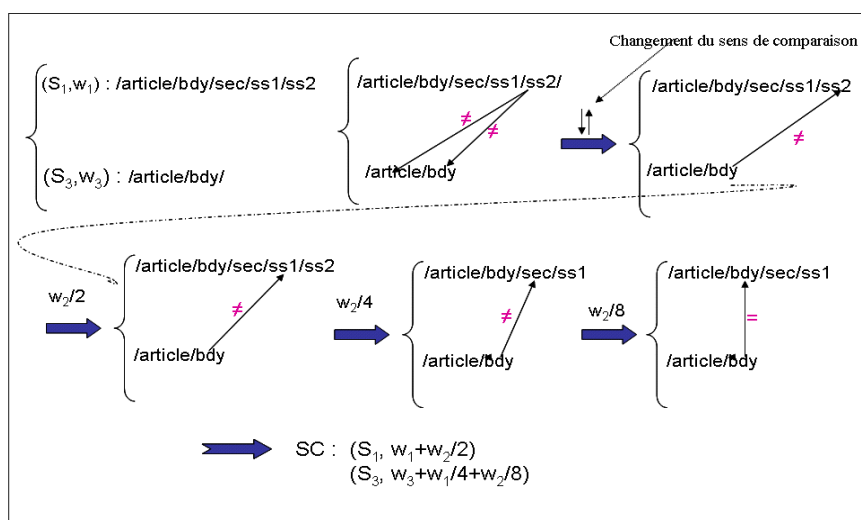


FIG. 3.6 – Recherche d’une structure générique : C

On arrive à trouver la structure commune entre S_3 et S_1 au bout de 2 itérations. Le score de la structure de S_1 est divisée par 2^2 et la structure `/article/bdy` est rajoutée à l'ensemble SC .

On passe ensuite à la comparaison des structures S_2 et S_3 (figure 3.6). On applique le même principe que précédemment, le premier sens de comparaison ne donne pas de résultat, on passe à la sous étape suivante.

Pour retrouver l'ancêtre commun, on effectue trois fois les passages au niveau plus haut. Par conséquent, le score de S_2 est divisé par 2^3 . On remarque que la structure `/article/bdy` existe déjà dans l'ensemble SC . Son score est alors incrémenté dans SC de $w_2/8$.

La structure générique sera celle qui a le plus élevé score parmi l'ensemble SC . Dans cet exemple on sélectionne la structure S_1 (`/article/bdy/sec/ss1`).

L'algorithme proposé concerne principalement les collection homogènes dans lesquelles tous les documents ont la même DTD.

Remarque :

Intuitivement, on pourrait penser que la structure générique est `/article/bdy`. Ce n'est pas le cas car en fait notre algorithme ne cherche pas l'ancêtre commun seulement, mais aussi celui qui a la plus petite distance qui le sépare des autres structures, d'où le résultat.

En réalité, les documents peuvent provenir de différentes sources n'ayant pas la même DTD, d'où la question : Comment peut-on traiter l'hétérogénéité des collections ?

3.4.4 Extraction des structures pertinentes dans des documents hétérogènes

Nous proposons dans cette section d'étendre l'algorithme SCA pour prendre en compte l'hétérogénéité des structures. Nous entendons par structures hétérogènes celles qui décrivent des éléments de documents ayant différentes DTDs.

Exemple : pour une requête donnée, deux éléments jugés pertinents peuvent avoir les deux structures suivantes : `A/B/C` et `E/C/D`. On constate que dans ce cas il n'existe pas une séquence de balises commune depuis les racines des deux structures. Nous définissons tout d'abord la notion de *classe de structures*

et classe de structures génériques.

Une classe de structures est un ensemble de structures ayant la même balise finale.

Nous nous intéressons ici principalement au type de l'élément répondant au besoin de l'utilisateur (le type d'élément est spécifié par la dernière balise d'une structure). Une classe de structure de type A notée $C(X)$ est définie comme suit :

$$C(X) = \{S | S.last = X\}$$

où S est une structure.

A partir de la notion de classe de structure, nous définissons la notion de classe de structures génériques.

Une classe de structures génériques est une classe de structure ayant le plus grand nombre d'éléments.

$$TC = argmax_{X \in \Omega} (|C(X)|)$$

Ω : l'ensemble de classe de structures extraites des éléments pertinents

$|C(X)|$: le nombre d'éléments pertinents ayant des structures appartenant à $C(X)$.

Pour pouvoir trier les classes de structures génériques ayant le même nombre TC , on assigne à chaque classe un score calculé en fonction de l'occurrence de la balise caractérisant la classe dans les structures S_i des éléments pertinents tout en tenant compte de la distance qui la sépare $S_i.last$.

Le processus d'extraction de la structure générique est comme suit :

1. La première étape consiste à construire les classes de structure à partir de l'ensemble des éléments jugés pertinents.
Exemple : Si on considère les quatre structures suivantes : $A/B/C$, $E/C/D$, $H/I/C$ et $A/E/F$. Les classes de structures sont : $C(C) = \{A/B/C, H/I/C\}$, $C(D) = \{E/C/D\}$ et $C(F) = \{A/E/F\}$.
Les classes seront triées selon la valeur de TC .
La classe de structure générique est celle qui a la valeur la plus élevée de TC . dans l'exemple ça sera $C(C)$.
S'il existe plusieurs classes ayant la même valeur de TC , on passe à l'étape suivante.

2. La deuxième étape consiste à calculer les scores des classes ayant la même valeur de TC .

$$Weight(X) = \sum_{\forall S_i \in E^p / X \in S_i} S_i / (d(X, S_i) + 1) \quad (3.13)$$

Où :

S_i est une structure d'un élément appartenant à l'ensemble des éléments pertinents (E^p), dans laquelle apparaît la balise X
 $d(X, S_i)$ la distance qui sépare la balise considérée et la dernière balise de la structure S_i .

Exemple : distances calculées pour la classe C de l'exemple.

$$d(C, A/B/C) = 0, d(C, E/C/D) = 1 \text{ et } d(C, H/I/C) = 0$$

$$Weight(C) = 1 + 1/2 + 1 = 2,5$$

A l'issu, la classe de structures générique est celle ayant le meilleur $Weight(C)$.

Jusqu'à présent, nous avons considéré que la dernière balise d'une structure à intégrer. Une autre alternative à envisager est de considérer tout le chemin d'un élément recherché, c'est à dire spécifier les différentes balises qui constituent la structure génériques.

Pour ce faire, nous calculons les poids des balises intérieures de chaque structure des éléments pertinents. Ceci nous ramène dans le cas de l'exemple à calculer le poids des balises A, B, C, D, E, F, I et H . Nous construisons ensuite le graphe des nœuds (voir figure 3.7). Les structures sont présentées dans un graphe orienté composé des nœuds représentant des balises pondérées selon la formule 3.13 et les arcs représentent les chemins extraits des structures des éléments pertinents. On assigne ensuite à chaque structure S_i un score :

$$score(S_i) = \sum_{\forall X \in S_i} weight(X)$$

La structure générique est alors celle qui a le score le plus élevé. Ce processus permet de sélectionner le chemin partagé par le maximum d'éléments pertinents.

Ainsi, si on regarde la figure 3.7, nous remarquons que le chemin ayant le score le plus important de la classe C est : $A/E/C$, la somme est $2/3 + 5/6 + 2.5 = 4$.

On remarque que le chemin de cette structure ne correspond à aucun chemin ou sous-chemin des structures des éléments jugés pertinents. De ce fait, il y a

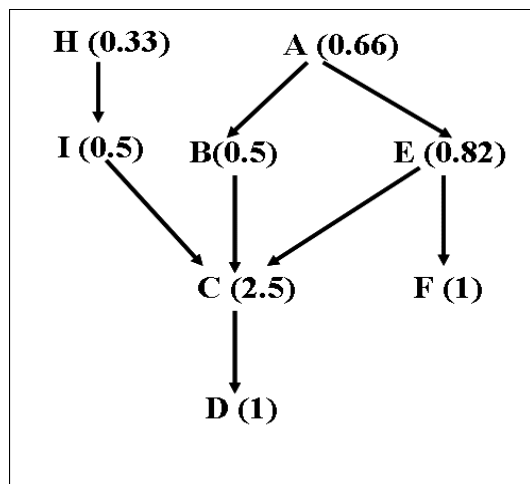


FIG. 3.7 – Présentation des structures dans un graphe orienté

une probabilité faible qu'on puisse retrouver des éléments qui satisfont cette structure. D'autres possibilités sont alors à envisager pour exprimer le chemin de la structure. Ces possibilités sont définies de manière plus flexible. C'est à dire on peut exprimer un chemin en ignorant quelques balises intermédiaires entre la racine et la dernière balise. On l'appelle chemin générique.

Exemple de chemin indéterminé : `//A//C`.

Pour garder le sens des structures génériques, les balises seront celles qui sont partagées au maximum par l'ensemble des chemins des éléments pertinents en d'autres termes celles qui correspondent aux nœuds ayant les scores les plus élevés.

Dans les deux processus utilisant l'algorithme SCA et la classification nous extrayons des structures complètes. Étant donné que l'algorithme SCA concerne le cas des documents homogènes (même DTD), nous réinjectons seulement la dernière balise de la structure (forme simplifiée). Dans le cas des document hétérogènes, la réinjection peut concerner soit la dernière balise (forme simplifiée) de la structure pertinente, soit la structure complète (spécifiant toutes les balises intermédiaire) ou son chemin générique (en éliminant quelques balises intermédiaires).

3.4.5 Réécriture de la requête

La réécriture des requêtes concerne les deux types de requêtes (structurées et non structurées). Pour les reformuler, on sélectionne les structures ayant les scores les plus élevés. Ces structures seront utilisées sous deux formes différentes

(la réinjection peut concerner une ou plusieurs structures pertinentes) :

- une forme simplifiée qui correspond à l’ajout à la requête initiale de la dernière balise de la structure sélectionnée. En cas de plusieurs structures, la requête est composée de plusieurs sous-requêtes dont chacune spécifie une contrainte structurelle.
- une forme complexe qui correspond à l’ajout à la requête initiale du (des) chemin(s) de la (des) structure(s) sélectionnée(s).

Considérons l’exemple des deux types de requêtes auxquels on réinjecte une structure pertinente S qui peut être sous forme simplifiée (balise) ou sous forme d’un chemin :

1. Soit $R1 = t_1, t_2, \dots, t_n$ une requête composée de n mots clés. La requête reformulée par réinjection de S sera :

$$R1' = \text{élément cible} : S[t_1, t_2, \dots, t_n]$$

2. Soit $R2$ une requête structurée avec $CS1$ et $CS2$ sont les anciennes contraintes structurelles et $CS1$ est l’élément cible :

$$R2 = \text{élément cible} : CS1[t_1, t_2, \dots, t_i] // CS2[t_{i+1}, t_{i+2}, \dots, t_n].$$

La nouvelle requête sera de la forme :

$$R2' = \text{élément cible} : CS1[t_1, t_2, \dots, t_i] // CS2[t_{i+1}, t_{i+2}, \dots, t_n] \text{ OR } S[t_1, t_2, \dots, t_i].$$

où OR est l’opérateur booléen pour exprimer la disjonction des sous requêtes. Le même opérateur est utilisé pour relier les sous requêtes après la réinjection de plusieurs structures pertinentes.

Dans le cas de la réinjection d’une forme complexe la structure S sera remplacée par son chemin. Pour généraliser cette écriture, nous définissons la grammaire présentée dans le tableau 3.5.

Exemple

Dans ce qui suit, les requêtes sont formulées selon le langage de requête XFIRM [169]. On distingue les deux types de requêtes (structurées et non structurées).

- Soit la requête initiale de type CO : Q ”reformulation des requêtes en recherche d’information”. La nouvelle requête sera reformulée par ajout d’une structure $ss1$ et donc de type CAS :

$$\text{”ec} : // \text{ss1} [\text{”reformulation des requêtes en recherche d’information”}] \text{”}.$$

<p>Soient R1 la réécriture d'une requête non structurée et R2 la réécriture d'une requête structurée :</p> <p>R1 := <élément cible><contrainte structurelle><c1><Requête initiale1><c2><Suite Expressions Structurées>*</p> <p>R2 := <Requête initiale2> <Suite Structures>*</p> <p>Requête initiale1 := requête non structurée</p> <p>Requête initiale2 := requête structurée</p> <p>Suite Expressions Structurées := <Opérateur><contrainte structurelle><c1><Requête Initiale><c2></p> <p>Suite Structures := <Opérateur><contrainte structurelle><c1><MC><c2></p> <p>Opérateur := "OR"</p> <p>élément cible := "//ec : "</p> <p>contrainte structurelle := nom de la balise extraite—chemin de la balise</p> <p>MC := mots clés de l'élément cible de la requête initiale</p> <p>c1 := "["</p> <p>c2 := "]"</p>

TAB. 3.5 – Grammaire de la réécriture des requêtes par injection de structure.

Si on considère plusieurs structures génériques S_i à réinjecter ($i \in \{2, \dots, n\}$), la requête finale sera de la forme :

"ec :// S₁ [reformulation des requêtes en recherche d'information"] OR S₂ [reformulation des requêtes en recherche d'information"] OR ... OR S_n [reformulation des requêtes en recherche d'information]".

– Soit une requête structurée de type CAS :

"article [recherche d'information] //ec :paragraphe [reformulation des requêtes]"

La requête reformulée par réinjection d'une structure *ss1* sera :

"// article [recherche d'information] ec : //paragraphe [reformulation des requêtes]" OR ss1 [reformulation des requêtes]"

L'opérateur Booléen "OR" exprime une éventuelle contrainte structurelle. On traite de la même façon le cas de réinjection de plusieurs structures génériques, la requête finale sera sous le format :

"// article [recherche d'information] ec : //paragraphe [reformulation des requêtes]" OR S₁ [reformulation des requêtes] OR S₂ [reformulation des requêtes] OR ... OR S_n [reformulation des requêtes]".

A ce niveau, nous avons présenté les deux différentes approches orientée contenu et orientée structure. Une question qui se pose maintenant porte sur

l'intérêt des combiner ces deux approches ?

Dans la prochaine section nous allons présenter les différentes stratégies de combinaison que nous avons envisagées.

3.5 Approche Combinée

Nous proposons dans cette section de combiner les sources d'évidence contenu et structure extraites des éléments pertinents afin d'exprimer des relation contextuelles qui peuvent exister entre elles. Cette combinaison peut se faire de trois manières différentes :

1. une combinaison naïve,
2. une combinaison avec dépendance contextuelle,
3. une combinaison flexible.

3.5.1 Combinaison naïve

Une première forme de combinaison, qualifiée de naïve, consiste à rajouter à la requête initiale à la fois les termes pertinents et les structures pertinentes. Les processus ayant permis l'extraction des termes et des structures pertinentes sont indépendants.

Cette combinaison est effectuée au niveau de la réécriture. On distingue les deux types de requêtes :

- La requête de type CO sera reformulée en ajoutant les termes extraits comme pertinents aux termes originaux de la requête. L'ensemble des termes (termes originaux de la requête + termes pertinents sélectionnés) sera conditionné par la structure pertinente extraite à partir des éléments pertinents selon l'approche orientée structure.

Formellement, soit $R1 = t_1, t_2, \dots, t_n$ une requête composée de n mots clés. La requête reformulée par réinjection d'une structure S et de 3 termes pertinents t_k, t_l et t_m sera :

$R1'$ =élément cible : $S [t_1, w_1 t_2, w_2 \dots t_n, w_n, t_k, w_k t_l, w_l t_m, w_m]$ où w_i est le poids correspondant à chaque terme calculé selon l'approche orientée contenu.

- Dans le cas d'une requête structurée, la nouvelle structure à réinjecter sera une condition sur les termes de l'élément cible de la requête initiale auxquels on ajoute les termes pertinents sélectionnés. L'ensemble représente une sous requête qui sera coordonnée avec la requête initiale avec l'opérateur booléen OR.

Formellement, soient CS_1 et CS_2 sont les anciennes contraintes structurales avec CS_1 est l'élément cible :

$R2$ =élément cible : $CS_1 [t_1, t_2, \dots, t_i] // CS_2[t_{i+1}, t_{i+2}, \dots, t_n]$. La nouvelle requête par réinjection d'une structure S et de 3 termes pertinents t_k , t_l et t_m sera de la forme :

$R2'$ =élément cible : $CS_1 [t_1, w_1 t_2, w_2 \dots t_i, w_i] // CS_2[t_{i+1}, w_{i+1} t_{i+2}, w_{i+2} \dots t_n, w_n]$

OR $S [t_1, w_1 t_2, w_2 \dots t_i, w_i, t_k, w_k t_l, w_l t_m, w_m]$.

où OR est l'opérateur booléen pour exprimer la conjonction des sous requêtes.

En général, la réécriture des deux requêtes suit la grammaire suivante (voir tableau 3.9).

<p>Soient R1 la réécriture d'une requête non structurée et R2 la réécriture d'une requête structurée :</p> <p>R1 : :<élément cible><contrainte structurelle><c1><Requête initiale1><Mots Clés><c2><Suite Expressions Structurées>*</p> <p>R2 : := <Requête initiale2><Suite Structures>*</p> <p>Requête initiale1 : := requête non structurée</p> <p>Requête initiale2 : := requête structurée</p> <p>Suite Expressions Structurées : :=<Opérateur_i><contrainte structurelle><c1><Requête Initiale><Mots Clés><c2></p> <p>Suite Structures : := <Opérateur><contrainte structurelle><c1><Mots clés Cibles><Mots Clés><c2></p> <p>Opérateur : := "OR"</p> <p>élément cible : := " // ec : "</p> <p>contrainte structurelle : := nom de la balise extraite</p> <p>Mots Clés : := mots clés sélectionnés</p> <p>Mots Clés Cibles : := mots clés appartenant à l'élément cible de la requête initiale</p> <p>c1 : := "["</p> <p>c2 : := "]"</p>
--

TAB. 3.6 – Grammaire de la réécriture des requêtes par injection des structures et des mots clés.

Exemple :

Soit la requête 202 de la collection de test INEX 2005 :

Requête CO

$R1$ = " *ontologies case study* "

et requête CO+S¹

$R2$ = " *article[ontologies] // ec : sec[ontologies case study]* "

sous forme de requête structurée où le terme *ec* marque la sous requête cible c'est-à-dire le type d'éléments désiré par l'utilisateur.

¹Nous rappelons qu'une requête de type CO+S est une requête comportant des mots clés + une contrainte structurelle qui sera traitée d'une manière vague

On considère que les termes à réinjecter sont : (graph, 1), (concept, 0.6) et que les termes de la requête initiale ("ontologies case study") sont pondérés respectivement par 1, 0.3 et 0.4. On considère également "paragraph" comme structure pertinente à réinjecter. Les requêtes reformulées R1' et R2' seront :

– Requête CO :

$R1' = \text{"ec : //paragraph [ontologie,1 case,0.3 study,0.4 graph,1 concept,0.6]"}$

– Requête CO+S :

$R2' = \text{" article[ontologies,1] // ec : sec[ontologies,1 case,0.3 study,0.4] OR paragraph [ontologies,1 case,0.3 study,0.4 graph,1 concept,0.6] "}$

3.5.2 Combinaison avec dépendance contextuelle

Jusqu'à présent, nous avons considéré les termes pertinents indépendamment des structure pertinentes.

Une hypothèse envisageable est de considérer que l'importance des termes dépende des structures dans lesquelles ils apparaissent : leur contexte. L'intuition derrière cette hypothèse est qu'il est possible qu'il ait un lien entre les termes pertinents et les structures pertinentes. En XML, les balises ont un rôle syntaxique pour structurer le document mais porte également une sémantique. On entend par sémantique d'un élément la balise qui le décrit, par exemple s'il s'agit d'un *article*, d'un *paragraphe*, d'une *référence* ou d'*url*, il est peu envisageable de mettre une balise "section" pour un titre de l'article.

Si l'on considère qu'une adresse *url* peut présenter un élément pertinent pour l'utilisateur, généralement, les termes extraits de cet élément ne sont pas pertinents. D'où l'idée d'étudier la sémantique des éléments pertinents.

Notre objectif est alors de répondre à la question déjà posée : doit-on tenir compte de la sémantique des éléments pour l'extraction des termes pertinents ?

On peut procéder d'une manière inverse. Il s'agit d'extraire des structures pertinentes en tenant compte des termes déjà extraits (Ce cas peut être un cas particulier de notre processus dans lequel on considère que tous les termes des éléments pertinents sont pertinents). De ce fait, nous nous intéressons à l'extraction des termes pertinents en fonction des structures pertinentes. Nous distinguons dans ce cas deux approches différentes :

1. Une première est d'affecter un degré d'importance prédéfini aux différentes structures de manière à considérer par exemple que pour deux termes différents ayant les mêmes caractéristiques, pertinence, contexte et bruit, le fait qu'un terme appartienne à un titre peut sembler sémantiquement plus intéressant que celui appartenant à une section ou une référence.

Cette préférence reste assez subjective puisque c'est un jugement qui dépend essentiellement de la requête : s'il s'agit d'une requête portant sur une référence il sera plus intéressant d'affecter un poids plus élevé au terme extrait d'une *référence* que d'un *titre* ou d'une *section*.

2. Une seconde approche est d'augmenter le poids de pertinence d'un terme en fonction du type des structures qui sont déjà sélectionnées pertinentes selon l'approche orientée structure. La pertinence d'un terme dépend alors de la nature de la structure à laquelle il appartient.

Nous procédons alors de manière à restreindre l'ensemble des éléments à partir desquels on extrait les termes pertinents. Si un élément ne fait pas partie des structures pertinentes, il ne sera pas considéré dans la phase d'extraction des termes. Dans ce cas la formule d'extraction déjà présentée dans le paragraphe 3.3.1.1 devient la suivante :

$$P(t_i|E_{res}^p) = p_{e_{res}}/|E_{res}^p| \quad (3.14)$$

Où

E_{res}^p est l'ensemble restreint des éléments pertinents possédant des structures pertinentes.

$p_{e_{res}}$ est le nombre des éléments pertinents appartenant à E_{res}^p contenant le terme t_i .

On remarque que cette restriction peut affiner l'extraction des termes de manière à ne considérer que ceux appartenant à des structures pertinentes. Cette approche en revanche ne donne pas de poids relatifs aux termes appartenant à deux structures pertinentes différentes. On peut dire que ce sont des poids binaires : 1, s'ils appartiennent aux structures pertinentes, 0 sinon.

Cette technique peut être affinée davantage en attribuant des scores aux structures pertinentes qui seront combinés avec le score du terme à calculer. Ce score traduit le degré d'importance d'une structure d'un élément pertinent. Les scores des structures seront les poids calculés dans la phase de l'extraction des structures génériques. Nous considérons par la suite l'ensemble des couples (structure, score) pour l'extraction des termes. Exemple : (paragraphe, 0.7), (référence, 0.4).

La nouvelle formule pour l'extraction des termes pertinents est alors la suivante :

$$Poids(t_i, E_{res}^p) = score(SG) \times p_{e_{res}}/|E_{res}^p| \quad (3.15)$$

Où le $score(SG)$ est le score de la structure générique calculé selon l'algorithme SCA décrit dans le paragraphe 3.4.3.2.

On applique la même grammaire que pour la combinaison naïve (paragraphe

3.5.1) pour la réécriture des requêtes en réinjectant les termes et les structures sélectionnés.

3.5.3 Combinaison flexible

L'approche ci-dessus peut être considérée comme stricte : la relation de dépendance exclut tout terme n'appartenant pas aux structures génériques. Les termes seront donc pénalisés. Pour pallier ce problème, nous proposons une combinaison flexible des deux sources d'évidence. On cherche alors à calculer le poids d'un terme en fonction des éléments dans lesquels il apparaît. Considérons une liste de termes et une liste de structures génériques, la combinaison flexible consiste à distribuer (répartir) les termes de manière à faire apparaître chacun d'eux dans le type d'éléments qui le concerne (où il apparaît) pour formuler une requête du type :

"article [recherche d'information] //ec : paragraphe [reformulation des requêtes]"

qui peut remplacer une requête du type :

"ec : article [recherche d'information et reformulation des requêtes]".

Le processus de distribution est réalisé comme suit :

Considérons les 3 termes pertinents : t_i , t_j et t_k et les 3 structures pertinentes A, B et C. Nous supposons que les occurrences de chaque terme dans chaque structures sont comme décrites dans le tableau 3.7 :

TAB. 3.7 – Distribution des termes dans les structures génériques.

	A	B	C	Nombre d'éléments pertinents
t_i	2	5	3	10
t_j	6	3	0	9
t_k	0	0	2	2

L'idée est de calculer la distribution d'un terme dans l'ensemble des structures pertinentes, c'est à dire de quelle manière est distribuée sa fréquence totale dans les différents types d'éléments. Pour un terme donné on calcule la somme de ses occurrences dans les éléments ayant le même type X divisée par sa fréquence totale. Ce facteur est appelé partition $Part(t_i, X)$.

$$Part(t_i, X) = \frac{\sum_{j=1}^N Occ(t_i, e_j)}{\sum_{k=1}^M Occ(t_i, e_k)} \quad (3.16)$$

avec

N est le nombre des éléments (e_j) ayant une structure pertinente de type X dans lesquels occure le terme t_i ,

M est l'ensemble des éléments pertinents dans lequel occure t_i et

$Occ(t_i, e_j)$ est le nombre d'occurrence du terme t_i dans l'élément e_j

D'après le tableau 3.7, les distributions du terme t_i dans les différentes structures (A, B et C) sont les suivantes : 2/10, 5/10 et 3/10. On remarque alors que le terme t_i occure plus fréquemment dans des éléments de type B. Autrement dit, il existe une relation plus solide entre le terme t_i et la structure B.

La distribution peut nous renseigner sur des relations entre termes et structures mais n'est pas assez discriminante. En effet si on considère qu'un terme occure autant de fois dans deux types d'éléments différents ayant des tailles différentes, il est évident que la relation entre le terme et le type de plus petite taille est plus intéressante que celle avec des éléments de plus grande taille. Pour avoir alors des relations discriminantes nous avons tenu compte de la taille des éléments. La formule qui traduit la relation entre le terme t_i et structure X est alors comme suit :

$$Rel(t_i, X) = \frac{\sum_{j=1}^N Occ(t_i, e_j)/(|e_j|)}{\sum_{k=1}^M Occ(t_i, e_k)} \quad (3.17)$$

Où :

$|e_j|$ la taille (nombre de termes) de l'élément e_j .

On suppose que la taille de chacun des éléments de type A, B et C est respectivement, 30, 15 et 12. La matrice terme structure sera comme illustré dans le tableau 3.8.

TAB. 3.8 – Les relations termes pertinents-structures génériques.

	A	B	C
t_i	2/300	5/150	3/120
t_j	6/270	3/135	0
t_k	0	0	2/24

Nous proposons alors d'intégrer cette relation sémantique au niveau de la pondération des termes de la requête. Nous proposons de calculer un nouveau poids pour chaque terme dans chaque type d'élément. La formule de pondération est alors la suivante :

$$Poids(t_i, A) = W(t_i) \times \frac{\sum_{j=1}^N Occ(t_i, e_j)/(|e_j|)}{\sum_{k=1}^M Occ(t_i, e_k)} \quad (3.18)$$

Avec $W(t_i)$ est le poids du terme calculé selon une des fonction de podération déjà présentées (équation 3.5 et 3.9). Cette méthode est intéressante lorsqu'elle

concerne la réinjection de plus qu'une structure pertinente. En effet dans le cas d'une seule structure, tous les mots-clés partagent une même contrainte structurelle.

La réécriture des requêtes garde le même principe que précédemment. Cependant, les mots clés sont répartis dans les sous requêtes, dans lesquelles on spécifie les différentes contraintes structurelles, de manière à ce que si un terme pertinent n'appartient pas à un type d'élément (structure) il ne fera pas partie des mots clés de la sous-requête utilisant ce type d'élément. Cette réécriture est représentée dans la grammaire suivante :

Soient R1 la réécriture d'une requête non structurée et R2 la réécriture d'une requête structurée :

R1 : :<élément cible><contrainte structurelle><c1><Requête initiale1><Mots Clés ><c2><Suite Expressions Structurées>*

R2 : := <Requête initiale2><Suite Structures>*

Requête initiale1 : := requête non structurée

Requête initiale2 : := requête structurée

Suite Expressions Structurées : :=<Opérateur><contrainte structurelle><c1><Requête Initiale><Mots Clés><c2>

Suite Structures : := <Opérateur><contrainte structurelle><c1><Mots clés Cibles><Mots Clés><c2>

Opérateur : := "OR"

élément cible : := " //ec : "

contrainte structurelle : := nom de la balise extraite

Mots Clés : := mots clés sélectionnés appartenant à la condition structurelle spécifiée

Mots Clés Cibles : := mots clés appartenant à l'élément cible de la requête initiale et à la condition structurelle spécifiée

c1 : := "["

c2 : := "]"

TAB. 3.9 – Grammaire de la réécriture des requêtes par injection flexible des structures et des mots clés.

Exemple :

On considère les termes t_i , t_j et t_k et les trois structures A, B et C à réinjecter dans deux requêtes :

- CO R1 = t_1, t_2, \dots, t_n composé de n mots clés et structurée
- CO+S R2 = $ec : S[t_1, t_2, \dots, t_n]$ composée d'un élément cible de type S et de n mots clés.

Les requêtes reformulée seront comme suit :

R1' = $ec : A [(t_1, w_{1a}), \dots, (t_n, w_{na}), (t_i, w_{ia}), (t_j, w_{ja})]$
 OR $B [(t_1, w_{1b}), \dots, (t_n, w_{nb}), (t_i, w_{ib}), (t_j, w_{jb})]$ OR
 $C [(t_1, w_{1c}), \dots, (t_n, w_{nc}), (t_i, w_{ic}), (t_k, w_{kc})]$

R2' = $ec : S[t_1, 1 t_2, 1 t_n, 1]$ OR $A [(t_1, w_{1a}), \dots, (t_n, w_{na}), (t_i, w_{ia}), (t_j, w_{ja})]$
 OR $B [(t_1, w_{1b}), \dots, (t_n, w_{nb}), (t_i, w_{ib}), (t_j, w_{jb})]$
 OR $C [(t_1, w_{1c}), \dots, (t_n, w_{nc}), (t_i, w_{ic}), (t_k, w_{kc})]$

Où $w_{ia}, w_{ja}, w_{ib} \dots$ and w_{kc} sont les poids correspondant à chaque terme sélectionné dans chaque type d'élément. w_{1a}, \dots, w_{na} sont les poids des termes de la requête originale dans l'élément de type A, w_{1b}, \dots, w_{nb} sont les poids des termes de la requête originale dans l'élément de type B et w_{1c}, \dots, w_{nc} sont les poids des termes de la requête originale dans l'élément de type C.

Lorsque le terme n'apparaît pas dans un type d'élément son poids est égal à zéro.

Nous supposons que tous les mots clés originaux occurrent dans tous les types d'éléments.

Exemple :

Pour illustrer ce point, nous allons supposer les conditions résumées dans le tableau 3.10 : sachant que la requête initiale est la requête 202 de la collection de test INEX 2005 :

R1 = " ontologies case study" sous forme de requête non structurée et

R2 = " article[ontologies] // ec : sec[ontologies case study] " sous forme d'une requête structurée,

où le terme *ec* marque la sous requête cible c'est-à-dire désirée par l'utilisateur.

On considère que les termes à réinjecter sont *graph*, *concept*, *semantic*, que les structure à réinjecter sont *paragraphe*, *titre*, *section* et on garde les même relations illustrées dans le tableau 3.8.

TAB. 3.10 – Distribution des termes dans les structures génériques.

	paragraphe	titre	sous-section
ontologies	0.7	0.9	0.5
case	0.3	0.05	0.1
study	0.4	0.1	0.2
graph	0.06	0.3	0.25
concept	0.2	0.2	0
semantic	0	0	0.16

Les requêtes finale sont R1' et R2' :

- **R1'** = "ec : //paragraph [ontologie,0.7 case,0.3 study,0.4 graph,0.06 concept,0.2] OR titre [ontologie,0.9 case,0.05 study,0.1 graph,0.3 concept,0.2] OR sous-section [ontologie,0.5 case,0.1 study,0.2 graph,0.25 semantic,0.16]"
- **R2'** = " article[ontologies,1] // ec : sec[ontologies,1 case,1 study,1] OR paragraph [ontologie,0.7 case,0.3 study,0.4 graph,0.06 concept,0.2] OR titre [ontologie,0.9 case,0.05 study,0.1 graph,0.3 concept,0.2] OR sous-section [ontologie,0.5 case,0.1 study,0.2 graph,0.25 semantic,0.16]"

3.6 Conclusion

Dans ce chapitre, nous avons présenté des approches de réinjection de pertinence qui répondent aux caractéristiques de la RI structurée. Nous avons alors proposé trois différentes approches :

1. Une approche orientée contenu qui permet d'enrichir la requête initiale en réinjectant des mots clés. Ces mots clés sont tout d'abord extraits à partir des éléments jugés pertinents en utilisant différents indicateurs : la distribution des termes dans les éléments pertinents, la proximité de ces termes vis-à-vis ceux de la requête initiale et enfin le bruit qu'ils peuvent engendrer s'ils sont fréquents dans les éléments non pertinents.
2. Une approche orientée structure qui permet de réinjecter une ou plusieurs structures pertinentes. Nous avons alors commencé par montrer empiriquement l'existence de la notion de structures pertinentes. Nous avons ensuite défini un algorithme (SCA) pour l'extraction des structures pertinentes appelées ici structures génériques. Nous avons également proposé une solution d'extraction des structures génériques dans des collections de documents hétérogènes, c'est ce que nous avons appelée classe de structures.
3. Une troisième approche combine les deux premières afin d'enrichir la requête initiale en y ajoutant à la fois des termes et des structures pertinents. Nous avons présenté alors trois formes de combinaison : une naïve qui s'applique au niveau de la réécriture de la requête, une seconde qui traduit la dépendance entre les deux approches et une troisième plus flexible qui permet d'exprimer des relations "sémantiques" entre les termes et les structures. Cette dernière répartit les termes pertinents dans les structures pertinentes adéquates.

Le dernier chapitre est consacré à la phase de mise à l'épreuve de nos propositions sur les collections INEX.

Chapitre 4

Evaluations & Expérimentations

4.1 Introduction

Pour évaluer les différentes approches proposées dans cette thèse, nous nous appuyons sur les collections de test fournies dans le contexte d'INEX.

Dans ce chapitre, nous nous intéressons à la construction de l'échantillon d'éléments renvoyés par le système à considérer pour l'extraction des évidences (termes +structures) ainsi qu'à l'étude qualitative des différentes approches présentées dans le chapitre précédent. Nous commençons par introduire notre plateforme d'évaluation dans la section 4.2, à savoir le système de recherche XFIRM, les collections de test INEX ainsi que les différentes stratégies d'évaluation utilisées dans nos expérimentations. Dans la section 4.3, nous évaluons l'impact du nombre d'éléments jugés sur les performances. Nous évaluons ensuite les performances des différentes approches proposées : l'approche orientée contenu (section 4.4), l'approche orientée structure (section 4.5) et l'approche combinée (sections 4.6). Nous présentons dans la section 4.7, l'impact du type de jugements de pertinence et des résultats de base¹ sur les performances de nos approches. Nous finissons dans la section 4.8 par présenter nos expérimentations sur la réinjection à itérations multiples ainsi que sur la réinjection aveugle.

¹Les résultats de base sont donnés par le système de recherche XFIRM

4.2 Plateforme pour l'évaluation

4.2.1 Le système de recherche XFIRM

Nous avons évalué nos approches en utilisant le système de recherche XFIRM [170]. Ce système est basé sur une méthode de propagation de pertinence. Des valeurs de pertinence sont d'abord calculées pour les différents noeuds feuilles (c'est à dire les noeuds contenant du texte). Ces valeurs sont par la suite propagées et agrégées vers les noeuds ancêtres.

Nous détaillons le modèle dans ce qui suit, à savoir la méthode d'évaluation de pertinence des noeuds feuilles, ainsi que l'évaluation de la pertinence des noeuds ancêtres dans le cadre des requêtes non structurées et structurées.

4.2.1.1 Évaluation de pertinence des noeuds feuilles

Si on considère un noeud feuille (nf) et une requête q composée de n mots clés (c'est à dire une requête de type CO dans la terminologie d'INEX), la valeur de pertinence du noeud feuille sera calculée en utilisant la fonction de similarité $RSV(q, nf)$ suivante [143] :

$$RSV(q, nf) = \sum_{j=1}^n w_j^q \times w_j^{nf} \quad (4.1)$$

Où :

$w_j^q = tf_j^q$ est le poids du terme j dans la requête q et $w_j^{nf} = tf_j^{nf} \times idf_j \times ief_j$ est le poids du terme j dans le noeud feuille nf .

Cette pondération permet d'exprimer à la fois l'importance des termes dans la collection de noeuds feuilles et la collection de documents.

tf_j^q et tf_j^{nf} sont respectivement la fréquence du terme j dans la requête q et dans le noeud feuille nf ,

$idf_j = \log(|D|/(|d_j| + 1)) + 1$, avec $|D|$ le nombre total des documents dans la collection, $|d_j|$ le nombre de documents contenant le terme j ,

et ief_j est la fréquence inverse d'élément du terme j , c'est à dire $\log(|NF|/|nf_j| + 1) + 1$, où $|nf_j|$ est le nombre de noeuds feuille contenant le terme j et $|NF|$ est le nombre total de noeuds feuilles dans la collection.

La valeur de pertinence d'un noeud interne n (différent d'un noeud feuille) est calculée différemment par propagation de pertinence selon le type de requête (structurée ou non structurée).

4.2.1.2 Propagation de pertinence dans une requête non structurée

On attribue pour chaque noeud de l'arbre du document une valeur de pertinence calculée en fonction des valeurs de pertinence des noeuds feuilles qui lui appartiennent. La valeur de pertinence d'un noeud interne n , définie par $RSV(q, n)$, est calculée comme suit :

$$RSV(q, n) = |F_n^p| \sum_{k=1..F_n} \alpha^{dist(n, nf_k)-1} * RSV(q, nf_k) \quad (4.2)$$

Où

les nf_k sont les noeuds feuilles descendants du noeud n , $dist(n, nf_k)$ est la distance entre le noeud n et le noeud feuille nf_k dans l'arbre du document (c'est à dire le nombre d'arcs nécessaires pour atteindre le noeud n en partant du noeud feuille nf_k),

$|F_n^p|$ est le nombre de noeuds feuilles descendants du noeud n ayant un score différent de zéro,

F_n est le nombre total de noeuds feuilles descendants de n , et $\alpha \in [0..1]$.

On peut également intégrer dans la mesure du score la pertinence que l'on accorde au document entier. On parle alors de pertinence contextuelle. La valeur de pertinence d'un noeud interne est défini alors comme suit :

$$p_n = \rho * |F_n^p| \sum_{k=1..N} \alpha^{dist(n, nf_k)-1} * RSV(q, nf_k) + (1 - \rho) * p_{racine} \quad (4.3)$$

avec p_{racine} la pertinence du noeud *racine* du document, calculée d'après l'équation 4.2. $\rho \in [0..1]$ est le paramètre servant de pivot et permettant d'ajuster l'importance de la pertinence du noeud racine.

4.2.1.3 Propagation de pertinence dans une requête structurée

Les requêtes structurées les plus précises se présentent sous le format suivant :

$$Q = RE//ec : RE//RE$$

où RE est une requête élémentaire de la forme :

$$RE = tg_1[t_{11}, t_{12}, \dots, t_{1n}] OR tg_2[t_{21}, t_{22}, \dots, t_{2n}] OR \dots$$

avec tg_i un nom de balise qui représentant une contrainte structurelle.

Les // entre les requêtes RE permettent d'exprimer des contraintes hiérarchiques et ec : permet de désigner les éléments cibles de la requête c'est à dire les éléments devant être renvoyés à l'utilisateur.

L'évaluation des requêtes est réalisée à travers les étapes suivantes :

1. Les requêtes sont décomposées en sous-requêtes élémentaires SRE , ayant la forme : $SRE = tg[q]$, où $q = t_1, \dots, t_n$ présente une contrainte sur le contenu composée de simples mots clés.
2. Les valeurs de pertinence sont par la suite calculées entre les noeuds feuilles et les conditions portant sur le contenu des sous requêtes élémentaires.
3. Les valeurs de pertinence sont propagées dans l'arbre du document afin de répondre aux contraintes structurelle des sous-requêtes élémentaires.
4. Les requêtes élémentaires RE sont ensuite évaluées en appliquant l'opérateur OR entre les sous requêtes élémentaires. Le résultat d'une RE est un ensemble de noeuds et les pertinences associées.
5. Pour évaluer les conditions de hiérarchie de la requête originale, les ensembles résultats des requêtes élémentaires sont combinées grâce à l'opérateur non-commutatif Δ défini ci-dessous. Cet opérateur permet de propager les pertinences des noeuds résultats de différentes RE vers les noeuds résultats de la requête élémentaire désignant les éléments cibles. Cette propagation est uniquement effectuée si les conditions hiérarchiques de la requête sont vérifiées dans les documents.

Soient deux ensembles de paires (noeud, pertinence) $R_i = (n, p_n)$ et $R_{i+1} = (m, p_m)$

$$R_i \Delta R_{i+1} = (n, p'_n) \quad (4.4)$$

avec

$$p'_n = \begin{cases} p_n + prop_{ag}(dist(m, n), p_n, p_m) & \text{si } n \in R_i \text{ est Ancetre de } m \in R_{i+1} \\ p_n & \text{sinon} \end{cases} \quad (4.5)$$

Où $prop_{ag}(dist(m, n), p_n, p_m) \rightarrow p'_n$ permet d'agréger les pertinences p_m du noeud m et p_n du noeud n en fonction de la distance qui sépare les deux noeuds, pour obtenir la nouvelle pertinence p'_n du noeud n .

Dans la troisième étape, la valeur de pertinence p_n d'un noeud n pour une sous requête élémentaire $SRE = tg[q]$ est calculée selon la formule suivante :

$$p_n = \begin{cases} \sum_{nf_k \in F_n} \alpha^{dist(n, nf_k)-1} * RSV(q, nf_k) & \text{si } n \in construct(tg) \\ 0 & \text{sinon} \end{cases} \quad (4.6)$$

où le résultat de la fonction $construct(tg)$ est l'ensemble des noeuds ayant tg comme nom de balise, et $RSV(q, nf_k)$ est calculée dans l'étape 2 avec l'équation 4.1. La fonction $construct(tg)$ utilise un index *Dictionnaire* qui présente pour une balise tg donnée les balises qui lui sont considérées équivalentes.

Pour évaluer des requêtes structurées en considérant les contraintes structurées comme vagues, on utilise l'index *Dictionnaire* composé d'équivalences étendues. Par exemple, un noeud de type *section* peut être considéré équivalent aux noeuds de type *paragraphe* ou aussi *body*. Ce dictionnaire est construit manuellement.

On trouvera plus de détails sur le modèle XFIRM dans [170].

4.2.2 Rappel sur les collections de test

Nous nous basons pour l'évaluation des performances sur la collection de test fournie dans le cadre de la campagne d'évaluation INEX (*INitiative for the Evaluation of XML Retrieval*).

Dans le premier chapitre, nous avons présenté les différentes tâches et mesures proposées depuis la mise en place de cette campagne. Dans cette section, nous nous focalisons sur les stratégies et les mesures appliquées dans la tâche *Relevance Feedback* en 2005 et 2006.

4.2.2.1 Collection de documents

Les collections de documents diffèrent entre 2005 et 2006 :

- En 2005, la collection présente une extension de la collection 2004 composée d'articles scientifiques provenant de la *IEEE Computer Society*, balisés au format XML. Elle comporte environ 17000 articles publiés de 1995 à 2004 provenant de 21 magazines ou revues différents ayant une taille totale d'environ 1,3 gigaoctets. En moyenne, un article contient 1532 noeuds XML, où la profondeur moyenne d'un noeud est 6.9. La collection contient au total 8 millions de noeuds et 180 balises différentes.
- En 2006, la collection est composée de 659388 documents en anglais issus de l'encyclopédie en ligne de Wikipedia [51] pour une taille totale d'environ 5 gigaoctets. Elle contient environ 5000 balises différentes. En moyenne un article contient 161.35 noeuds XML, où la profondeur moyenne d'un élément est 6.72.

4.2.2.2 Topics

Les topics, c'est à dire les thèmes à partir desquels les requêtes sont construites, sont créés par les différents participants et doivent être représentatifs des demandes de l'utilisateur sur la collection. On distingue deux types de requêtes :

- Les CO (*Content Only*) : ce sont des requêtes composées de simples mots clés. Les mots clés de la requête peuvent être éventuellement groupés sous forme d'expressions et précédés par les opérateurs '+' (signifiant que le terme est obligatoire) ou '-' (signifiant que le terme ne doit pas apparaître dans les éléments renvoyés à l'utilisateur).
- Les CAS (*Content And Structure*) : ces requêtes contiennent des contraintes

sur la structure des documents.

Les tâches de recherche proposées pour chaque type de requêtes en 2005 diffèrent de celles de 2006. Nous distinguons en ce qui concerne la tâche de *Relevance Feedback* :

- En 2005, des requêtes de type CO (40 requêtes dont 28 ont été jugées) ont été traitées dans une tâche CO dans laquelle on considère uniquement leur contenu (*titre* composé de mots clés) et dans une tâche CO+S dans laquelle on considère le même contenu que la requête CO avec une contrainte structurelle vague. Pour les requêtes structurées CAS (47 requêtes dont 12 ont été jugées), on n'a considéré que la tâche VVCAS².
- En 2006, un seul ensemble de requêtes a été proposé. Il a été traité selon deux tâches : CO et CO+S. Les requêtes sont au nombre de 125 dont 114 ont été jugées.

Dans la tâche *Relevance Feedback* seule la stratégie *Thorough* de recherche est appliquée pour les différents types de requêtes. On rappelle que dans la stratégie *Thorough*, on suppose qu'un utilisateur préfère retrouver tous les élément fortement pertinents.

4.2.2.3 Jugements de pertinence

Comme nous l'avons déjà introduit dans le premier chapitre, le jugement de pertinence en RI structurée concerne 2 dimensions : l'exhaustivité et la spécificité. En 2005, l'exhaustivité est mesurée selon une échelle à 4 niveaux :

exhaustivité { e=2 exhaustivité élevée
 e=1 exhaustivité moyenne
 e=0 pas d'exhaustivité
 e=? élément trop petit

En 2006, tous les éléments jugés pertinents sont d'exhaustivité=2.

La spécificité est mesurée dans un intervalle continu [0,1] où s=1 représente un élément totalement spécifique.

Pour obtenir des résultats de performance, les 2 dimensions de pertinence (exhaustivité et spécificité) sont agrégées en une seule valeur. Deux types de fonction d'agrégation sont utilisées :

²On rappelle que dans la tâche VVCAS le jugement de pertinence est fait selon le champ narrative de la requête, c'est à dire la contrainte structurelle n'est pas prise en compte.

- une agrégation "stricte" pour évaluer si un SRI est capable de retrouver des éléments très spécifiques et très exhaustifs

$$f_{strict}(e, s) = \begin{cases} 1 & \text{si } e = 2 \text{ et } s = 1 \\ 0 & \text{sinon} \end{cases} \quad (4.7)$$

- une agrégation "généralisée" pour évaluer les éléments selon leur degré de pertinence

$$f_{generalisee}(e, s) = e * s \quad (4.8)$$

4.2.2.4 Mesures d'évaluation

Les mêmes mesures ont été utilisées dans INEX 2005 et INEX 2006. Elles reposent sur deux principales mesures que nous allons utiliser durant nos expérimentations :

- La mesure xCG (le gain cumulé) : on utilise la forme normalisée $nxCG[i]$. Pour un rang donné i , le gain cumulé $nxCG[i]$ reflète le gain relatif de l'utilisateur accumulé jusqu'à ce rang, comparé à ce qu'il aurait dû atteindre si le système avait produit une liste triée optimale. Le $MANXCG[i]$ est la moyenne des gains cumulés jusqu'au rang i .
- Par analogie au gain cumulé, on définit l'effort-précision ($ep(r)$)
L'effort-précision ($ep(r)$) est calculé à des points de gain-rappel arbitraires, où le gain-rappel gr est la valeur du gain cumulé divisé par la valeur totale atteignable du gain cumulé.
L'effort-précision à une valeur donnée de gain-rappel mesure l'effort d'un utilisateur pour atteindre un gain relatif au gain total qu'il peut obtenir. On utilise dans nos expérimentations la moyenne non interpolée MAep (*Mean Average Effort Precision*). Elle permet de moyenniser les valeurs d'effort-précision pour chaque rang auquel un élément pertinent est renvoyé.

Par analogie aux courbes de rappel précision en RI classique, nous utilisons dans ce contexte les courbes de variation d'effort-précision en fonction des points du gain-rappel.

Nous rappelons également que l'évaluation de la réinjection de pertinence est mesurée par l'amélioration relative AR définie dans la section 2.5.2 du second chapitre.

4.2.3 Stratégies d'évaluation

Pour mettre en relief l'impact de nos approches sur la restitution de nouveaux éléments pertinents, les différentes expérimentations que nous avons menées ont été évaluées selon la stratégie résiduelle. On rappelle que la stratégie résiduelle évalue les requêtes initiales et la reformulation sur une collection résiduelle ne contenant plus les éléments jugés.

Nous utiliserons la stratégie d'évaluation "freezing" adoptée pour les résultats officiels de la campagne d'évaluation INEX à la fin de nos expérimentations. Ces résultats sont présentés dans la section 4.9.

4.2.4 Résultats de base

Pour évaluer les performances de nos approches, nous devons tout d'abord fixer les résultats de base à partir desquels nous allons construire notre échantillon pour la reformulation. Ces résultats seront par la suite comparés à ceux obtenus après reformulation en appliquant les mêmes paramètres de recherche.

Pour ce faire, nous avons considéré les tâches de recherche CO, CO+S et VV-CAS de la collection 2005 et CO et CO+S de la collection 2006 pour retrouver la valeur du paramètre α du système de recherche XFIRM qui permet d'avoir les meilleurs résultats. α est une constante qui varie dans un intervalle $]0..1]$.

Pour juger les meilleures performances du système, nous privilégions les valeurs obtenues avec une agrégation stricte (les éléments strictement pertinents traduisent d'une manière exacte les besoins de l'utilisateur) ainsi que les MAep, présentant une mesure de performance globale sur l'ensemble des éléments retournés par le système de recherche (alors que les MANxCG présentent des performances dans des points précis de la liste des éléments restitués).

Nous résumons dans le tableau 4.1 les valeurs du paramètre α permettant d'obtenir les meilleurs résultats de base pour chaque type de requête de chaque collection. Nous présentons les mesures MAep et MANxCG[10] avec les deux fonctions d'agrégation (généralisée et stricte). Notons que les faibles valeurs de α permettant d'obtenir des résultats optimaux privilégient les éléments de petite taille dans la liste des résultats. Dans le cas des requêtes CO, le paramètre ρ est fixé à 0 (donnant les meilleurs résultats). Lorsque $\rho = 0$, seule la pertinence du document entre en compte pour le calcul de la pertinence des éléments.

Nous utiliserons pour relancer la recherche après reformulation les mêmes valeurs des constantes (α et ρ) que celles ayant permis d'obtenir les meilleurs résultats de base.

TAB. 4.1 – Résultats de base des collections 2005 et 2006.

	MA_nxCG [10] gen	MA_{ep} gen	MA_nxCG [10] stricte	MA_{ep} stricte	Rang/Nombre de partici- pants
CO 2005 ($\alpha = 0.2$)	0.1532	0.0457	0.0438	0.0156	29/55
CO+S 2005 ($\alpha = 0.1$)	0.2986	0.0568	0.1277	0.0316	3/33
VVCAS 2005 ($\alpha = 0.1$)	0.2532	0.0499	0.1189	0.0382	9/28
CO 2006 ($\alpha = 0.2$)	0.2356	0.0205	0.1319	0.0132	26/106
CO+S 2006 ($\alpha = 0.1$)	0.2016	0.0079	0.1749	0.0045	77/106

4.2.5 Démarche d'évaluation

La démarche que nous adoptons pour évaluer nos différentes propositions est la suivante. Nous allons commencer tout d'abord par évaluer l'impact de l'échantillon et du choix du nombre d'éléments jugés pour les différentes tâches des deux collections. Nous évaluons ensuite chacune des approches proposées. Les approches seront testées dans l'ordre de leur présentation dans le chapitre précédent : orientée contenu, orientée structure et approche combinée.

Afin de tirer des conclusions générale sur l'intérêt d'un point ou une proposition, nous commençons d'abord par observer les résultats obtenus sur les différentes tâches (CO, CO+S, VVCAS) pour les deux collections considérées, puis nous faisons un bilan global à partir de toutes les observations.

4.3 Échantillonnage

Un échantillon est principalement caractérisé par :

- le nombre d'éléments jugés,
- le nombre d'éléments jugés pertinents. Ce facteur influence directement les approches utilisées. En effet, si le nombre d'éléments pertinents est faible, voire nul, il sera difficile de déceler des termes pertinents ainsi que des structures pertinentes pouvant enrichir la requête initiale. Dans le cas où l'ensemble des éléments pertinents est vide, on ne peut pas appliquer la réinjection de pertinence.

- la nature du jugement de pertinence : c’est à dire si les éléments qui constituent l’échantillon sont jugés strictement (éléments très spécifiques et très exhaustifs) ou selon la fonction d’agrégation généralisée³.

Notre objectif est d’améliorer les performances du système en utilisant les mêmes conditions d’expérimentation (mêmes caractéristiques de l’échantillon). Nous nous intéressons en particulier à améliorer les performances du système de recherche en se basant sur des éléments strictement pertinents traduisant exactement le besoin de l’utilisateur. Nous ne considérons alors que les éléments jugés strictement pertinents pour la construction de l’échantillon. Dans les paragraphes suivants, nous allons évaluer l’impact du nombre d’éléments jugés ainsi que le nombre d’éléments jugés pertinents.

Nous utiliserons pour relancer la recherche après reformulation les mêmes valeurs des constantes (α et ρ) que celles ayant permis d’obtenir les meilleurs résultats de base. Nous avons appliqué :

- la reformulation orientée contenu (*ROC*) en réinjectant un seul terme pertinent. Pour la pondération des termes de la nouvelle requête, nous avons utilisé le poids *Probabiliste-Contextuel* (*Prob-Cont*) calculé selon l’équation 3.5. Les termes réinjectés n’apparaissent pas dans la requête initiale.
- la reformulation orientée structure (*ROS*) en réinjectant une seule structure générique extraite en appliquant l’algorithme SCA.
- la reformulation combinée naïve (*RCN*) en réinjectant un terme pertinent (on utilise les mêmes paramètres que ceux de *ROC*) et une structure générique extraite aussi en appliquant l’algorithme SCA.

D’autres valeurs ainsi que d’autres mode de combinaison seront testés par la suite dans les sections consacrées à l’évaluation de chacune des approches.

Nous rappelons aussi que les évaluations sont effectuées en appliquant la stratégie **résiduelle**. Pour permettre une meilleure lisibilité, les cas où nous n’obtenons pas d’améliorations seront représentés par des cellules vides (-).

4.3.1 Choix du nombre d’éléments jugés

D’après le protocole d’évaluation adopté par la campagne INEX, on considère les 20 premiers éléments retournés par le système de base pour construire l’échantillon et on utilise la stratégie de "freezing". Or nous supposons que le choix des 20 premiers éléments n’est pas forcément le meilleur pour tous les systèmes. En effet, si un système est assez performant, un nombre plus faible peut être efficace et dans le cas où le système ne retourne pas d’éléments perti-

³On rappelle que la fonction d’agrégation généralisée traduit le fait qu’un élément pertinent peut avoir différents degrés de spécificité et d’exhaustivité

nents dans les 20 premiers éléments, il sera plus important d'élargir l'ensemble des éléments à juger.

En réalité il n'existe pas de taille idéale pour construire l'échantillon. Ce dernier dépend directement :

- Des performances du système de base qui sont liées aux types des requêtes. En effet, les résultats diffèrent d'un type à un autre. On le constate par exemple dans le tableau récapitulatif des résultats des différents participants d'INEX. Par exemple en 2005, le système de recherche XFIRM se classe le 3^{ème} dans le cas des requêtes de type CO+S alors qu'il est à la 22^{ème} place dans le cas des requêtes de type CO. Par conséquent, le nombre d'éléments jugés pertinents n'est pas le même dans les n premiers résultats retournés par le système de recherche. On remarque également dans le cas de collection 2006, que malgré une meilleure expressivité des requêtes CO+S (comportant mots clés et structures), les résultats ne sont pas meilleurs que ceux obtenus par des requêtes CO (comportant que des mots clés)
- De la collection : on remarque dans le tableau 4.1 que pour le même type de requêtes, on ne retrouve pas les mêmes performances. Par exemple la MAep stricte des requêtes CO+S dans la collection 2005 et celle de 2006 sont respectivement, 0.0316 et 0.0045.
- De l'approche utilisée pour la réinjection de pertinence : puisque nous proposons différents processus pour l'extraction des termes et des structures pertinents, un nombre d'éléments pertinents peut être suffisant pour extraire une évidence mais pas pour une autre.

Dans ce paragraphe, nous allons étudier à travers des expérimentations la construction de l'échantillon selon les différents cas envisagés. Nous allons alors faire varier le nombre des éléments jugés pour chaque type de tâche de recherche (CO, CO+S VVCAS), dans chaque collection (2005 et 2006) et en utilisant les différentes approches : *ROC*, *ROS* et *RCN*. Le nombre d'éléments jugés varie dans l'ensemble {10, 20, 50}. Nous allons présenter les différents résultats par type de tâche.

4.3.1.1 Tâche CO

Les tableaux 4.2 et 4.3 listent les valeurs des améliorations relatives (*AR*) obtenues selon le nombre d'éléments jugés (*Nb-Elt-Jugés*) dans le cas des requêtes CO (comportant que des mots clés) de la collection 2005 et 2006. Le terme "Base" désigne les résultats de base obtenus sur la collection résiduelle.

Dans le cas de la collection 2006, les valeurs qui correspondent aux nombres d'éléments jugés 10 et 20 ne sont pas représentées car aucune amélioration n'est

TAB. 4.2 – Impact du nombre d’éléments jugés sur l’échantillon dans le cas de la tâche CO de la collection 2005

	App	MA_{nxC}G [10] gen	MAep gen	MA_{nxC}G [10] stricte	MAep stricte
Nb-Elt- Jugés=10	Base	0,1525	0,0401	0,0617	0,0137
	ROC	14%	31%	–	–
	ROS	101%	76%	49%	110%
	RCN	70%	65%	–	50%
Nb-Elt- Jugés=20	Base	0,1393	0,0379	0,0386	0,01
	ROC	48%	43%	–	10%
	ROS	120%	87%	138%	188%
	RCN	86%	75%	55%	106%
Nb-Elt- Jugés=50	Base	0,1736	0,0551	0,0269	0,0064
	ROC	0%	0%	–	64%
	ROS	77%	28%	242%	350%
	RCN	55%	23%	139%	230%

observée pour ces cas.

Le premier résultat intéressant que l’on observe, en particulier pour la collection 2005 est que nos trois approches apportent des améliorations significatives comparées aux résultats de base. Nous remarquons également que nous obtenons deux différents comportements selon la collection et l’approche considérées. En effet, on constate des améliorations significatives, en particulier pour les mesures généralisées dans le cas de la collection 2005 pour les différents nombre d’éléments jugés et pour les différentes approches. Concernant les mesures strictes on observe des améliorations claires, en particulier au niveau de la MAep, à partir de 20 éléments jugés. Les résultats sont beaucoup plus mitigés

TAB. 4.3 – Impact du nombre d’éléments jugés sur l’échantillon dans le cas de la tâche CO de la collection 2006

Tâches	App	MA_{nxC}G [10] gen	MAep gen	MA_{nxC}G [10] stricte	MAep stricte
Nb-Elt- Jugés=50	Base	0,142	0,0122	0,1065	0,0069
	ROC	4%	–	13%	4%
	ROS	–	–	15%–	–
	RCN	–	–	8%	–

TAB. 4.4 – Impact du nombre d’éléments jugés sur l’échantillon dans le cas de la tâche CO+S de la collection 2005

Tâches	App	MA _n xCG [10] gen	MAep gen	MA _n xCG [10] stricte	MAep stricte
Nb-Elt- Jugés=10	Base	0,2081	0,0461	0,1071	0,0444
	ROC	–	13%	–	–
	ROS	3%	15%	–	–
	RCN	3%	14%	–	–
Nb-Elt- Jugés=20	Base	0,1741	0,0407	0,0558	0,0197
	ROC	8%	33%	46%	30%
	ROS	3%	22%	5%	32%
	RCN	7%	23%	41%	32%
Nb-Elt- Jugés=50	Base	0,1027	0,028	0,0277	0,0085
	ROC	45%	46%	157%	23%
	ROS	29%	46%	–	–
	RCN	32%	46%	97%	16%

pour la collection 2006. On ne constate pas d’améliorations spécifiques, hormis, une légère amélioration au niveau de MA_nxCG[10] stricte pour 50 éléments jugés. Nous discuterons le nombre de requêtes qui améliorent les performances vis-à-vis de celles qui le détériorent dans la section 4.3.3.

Nous n’avons pas d’explication rationnelle quant à la différence des résultats entre ces deux collections. Nous pensons néanmoins que les deux collections n’ont pas les mêmes caractéristiques (nombre de balise, profondeur moyenne d’un noeud,...), et que ces caractéristiques influencent la performance du système de recherche ainsi que notre processus de réinjection de pertinence.

4.3.1.2 Tâche CO+S

Les tableaux 4.4 et 4.5 listent les valeurs des améliorations relatives obtenues selon le nombre d’éléments jugés (*Nb-Elt-Jugés*) dans le cas de la tâche de recherche CO+S des collections 2005 et 2006.

Le premier résultat que l’on peut tirer de ces deux tableaux est que nos différentes approches apportent des améliorations claires, en particulier à MA_nxCG[10] pour les deux mesures considérées et à partir de 20 éléments jugés. Le comportement au niveau de la MAep diffère dans les deux cas. En effet les

TAB. 4.5 – Impact du nombre d’éléments jugés sur l’échantillon dans le cas de la tâche CO+S de la collection 2006

Tâches	App	MA _n xCG [10] gen	MAep gen	MA _n xCG [10] stricte	MAep stricte
Nb-Elt- Jugés=10	Base	0,1299	0,0062	0,1115	0,0035
	ROC	–	–	17%	–
	ROS	9%	–	12%	–
	RCN	14%	–	16%	–
Nb-Elt- Jugés=20	Base	0,1224	0,0054	0,1114	0,0031
	ROC	15%	–	14%	–
	ROS	16%	–	15%	–
	RCN	19%	–	18%	–
Nb-Elt- Jugés=50	Base	0,0901	0,004	0,0717	0,0019
	ROC	33%	–	48%	10%
	ROS	30%	–	36%	0%
	RCN	47%	–	54%	0%

améliorations des MAep ne concernant que la collection 2005.

4.3.1.3 Tâche VVCAS

Le tableau 4.6 liste les valeurs des améliorations relatives obtenues selon le nombre d’éléments jugés (*Nb-Elt-Jugés*) dans le cas de la tâche VVCAS de la collection 2005.

On constate des améliorations significatives, en particulier à MA_nxCG[10] pour toutes les approches pour un nombre d’éléments jugés égal à 10 et 20. Un résultat surprenant, très positif, concerne la mesure stricte. En effet, on constate des améliorations très significatives pour toutes les approches et tous les gains considérés. Cette amélioration montre en outre un accroissement clair, en particulier au niveau de MAep strict, proportionnel au nombre d’éléments jugés.

En termes de tâches, nous remarquons que nos approches ont des impact très positifs dans la tâche VVCAS vis-à-vis les autres tâches. Ces résultats sont à considérer avec prudence car le nombre de requêtes considérées dans cette tâche est faible en le comparant aux autres tâches. En effet, parmi les 12 requêtes jugées, les requêtes ayant des éléments pertinents dans les 10, 20 et 50 premiers

TAB. 4.6 – Impact du nombre d’éléments jugés sur l’échantillon dans le cas de la tâche VVCAS de la collection 2005

Tâches	App	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Nb-Elt- Jugés=10	Base	0,22	0,0469	0,1177	0,0282
	ROC	7%	–	56%	147%
	ROS	19%	1%	83%	160%
	RCN	13%	2%	56%	146%
Nb-Elt- Jugés=20	Base	0,1425	0,0437	0,0904	0,0224
	ROC	51%	–	84%	187%
	ROS	34%	0%	63%	197%
	RCN	33%	4%	69%	207%
Nb-Elt- Jugés=50	Base	0,1772	0,0423	0,0927	0,0209
	ROC	–	–	63%	195%
	ROS	–	–	63%	214%
	RCN	–	–	64%	213%

éléments sont au nombre de 5, 6 et 8 (voir tableau 4.7). Ceci explique aussi le comportement brutal de cette tâche au niveau des mesures généralisées en considérant le nombre d’éléments jugés de 20 et 50, autrement dit, une seule requête peut influencer sur significativement le résultat global.

4.3.1.4 Discussion et bilan

Un premier résultat important que l’on peut tirer de ces premières expérimentations est qu’on observe des améliorations souvent significatives à partir des 10 premiers éléments retournés (MANxCG[10]) et/ou au niveau de la MAep pour les deux mesures strictes et généralisées et pour toutes les tâches considérées en jugeant les 20 premiers éléments hormis la tâche CO 2006.

Nous remarquons également que les approches ont un comportement très variable selon l’agrégation considérée (stricte/généralisée), les tâches et les collections, en particulier dans le cas de la tâche VVCAS, où les améliorations strictes sont beaucoup plus importantes que celles généralisées.

Afin de bien mettre en évidence l’impact du nombre d’éléments jugés, nous avons calculé la moyenne du nombre d’éléments pertinents dans chaque échantillon parmi les 10, 20 et 50 premiers éléments retournés par le systèmes.

TAB. 4.7 – Moyennes des éléments jugés pertinents dans les échantillons

	NRP (Top 10)	MEP (Top 10)	NRP (Top 20)	MEP (Top 20)	NRP (Top 50)	MEP (Top 50)
CO 2005	15/28	2.47	15/28	4.67	17/28	7.35
CO+S 2005	9/28	3.11	12/28	3.83	17/28	6.47
VVCAS 2005	5/12	2.8	6/12	4.83	8/12	8.88
CO 2006	51/114	2.78	63/114	5.52	81/114	11.58
CO+S 2006	65/114	2.31	77/114	3.34	88/114	6.10

Le tableau 4.7 résume, pour chaque type de tâches de recherche de chaque collection, le nombre de requêtes ayant des éléments pertinents (NRP) et le nombre moyen d'éléments jugés pertinents par requête (MEP). Pour cette dernière mesure, on ne considère pas les requêtes n'ayant aucun élément pertinent.

D'après le tableau 4.7, nous constatons que le nombre d'éléments jugés pertinents dans les 10 premiers éléments retournés par le système de recherche est généralement inférieur à 3. Ceci peut expliquer les cas souvent observés où nous n'obtenons pas d'améliorations ou bien des améliorations non significatives en considérant les 10 premiers éléments dans la construction de l'échantillon. De plus on constate que le nombre de requêtes ayant des éléments pertinents varie entre 6 et 17 pour la collection 2005. Ceci peut avoir des effets incontrôlables au niveau des performances. Ainsi, afin d'assurer un minimum d'éléments pertinents dans un échantillon, nous proposons de construire un échantillon non pas en fixant le nombre d'éléments jugés mais en fixant le **nombre d'éléments jugés pertinents**.

Nous allons étudier l'impact du nombre d'éléments pertinents sur les performances de notre système dans le paragraphe suivant.

Étant donné qu'en général, nous obtenons des améliorations significatives en considérant des échantillons construits à partir de 20 éléments pertinents (excepté le cas de la tâche CO de la collection 2006), nous allons focaliser notre étude pour des valeurs qui ne dépassent pas 5 éléments jugés pertinents. Nous ne considérons pas la tâche CO de la collection 2006 dans cette étude car aucune amélioration n'a été observée pour cette tâche.

TAB. 4.8 – Impact du nombre d’éléments jugés pertinents sur l’échantillon dans le cas de la tâche CO de la collection 2005

Tâches	App	MA _{nxC} G [10] gen	MA _{ep} gen	MA _{nxC} G [10] stricte	MA _{ep} stricte
Nb-Elt- Pert=1	Base	0,2367	0,0409	0,0757	0,0175
	ROC	–	28%	–	–
	ROS	28%	60%	22%	58%
	RCN	2%	48%	–	–
Nb-Elt- Pert=3	Base	0,2258	0,0355	0,1235	0,0307
	ROC	–	61%	–	–
	ROS	37%	95%	–	–
	RCN	23%	85%	–	–
Nb-Elt- Pert=5	e Base	0,213	0,0352	0,078	0,0202
	ROC	–	63%	–	–
	ROS	45%	97%	18%	45%
	RCN	30%	86%	–	20%

4.3.2 Choix du nombre d’éléments jugés pertinents dans un échantillon

Dans ce paragraphe nous varions le nombre des éléments jugés pertinents (*Nb-Elt-Pert*) dans l’ensemble {1, 3, 5}. Cette étude est effectuée pour les différentes tâches. Nous représentons les pourcentages des améliorations relatives pour les différentes mesures.

Les tableaux 4.8, 4.9, 4.11 et 4.10 présentent respectivement l’impact du nombre d’éléments jugés pertinents sur l’échantillon dans le cas des tâches de recherche CO+S et VVCAS de la collection 2005 et dans le cas de la tâche CO+S de la collection 2006.

Tout d’abord pour la collection 2005, d’une manière générale, dans le cas des tâches CO et CO+S, les meilleures performances ainsi que les meilleurs taux d’amélioration sont obtenus en considérant un échantillon composés de 3 éléments pertinents excepté dans quelques cas comme par exemple le cas de l’agrégation stricte de la tâche CO de la collection 2005 (tableau 4.8). Trois éléments pertinents semblent donc présenter un nombre suffisant pour appliquer la reformulation.

Concernant les requêtes CO+S de la collection 2006 (tableau 4.10), nous n’observons pas d’amélioration en utilisant les échantillons à 1 et 5 éléments per-

TAB. 4.9 – Impact du nombre d’éléments jugés pertinents sur l’échantillon dans le cas de la tâche CO+S de la collection 2005

Tâches	App	MA_{nxC}G [10] gen	MAep gen	MA_{nxC}G [10] stricte	MAep stricte
Nb-Elt- Pert=1	Base	0,1773	0,0324	0,1232	0,0357
	ROC	–	45%	–	–
	ROS	11%	39%	–	–
	RCN	76%	84%	47%	22%
Nb-Elt- Pert=3	Base	0,135	0,0247	0,0802	0,019
	ROC	7%	51%	–	–
	ROS	36%	65%	18%	26%
	RCN	137%	141%	106%	105%
Nb-Elt- Pert=5	Base	0,1473	0,0225	0,0841	0,0187
	ROC	3%	70%	–	–
	ROS	6%	63%	–	–
	RCN	77%	131%	71%	84%

TAB. 4.10 – Impact du nombre d’éléments jugés pertinents sur l’échantillon dans le cas de la tâche CO+S de la collection 2006

Tâches	App	MA_{nxC}G [10] gen	MAep gen	MA_{nxC}G [10] stricte	MAep stricte
Nb-Elt- Pert=3	Base	0,1458	0,0065	0,1263	0,0035
	ROC	8%	–	14%	–
	ROS	–	–	1%	–
	RCN	12%	–	19%	–

TAB. 4.11 – Impact du nombre d’éléments jugés pertinents sur l’échantillon dans le cas de la tâche VVCAS de la collection 2005

Tâches	App	MA _n xCG [10] gen	MA _{ep} gen	MA _n xCG [10] stricte	MA _{ep} stricte
Nb-Elt- Pert=1	Base	0,1804	0,1693	0,112	0,0283
	ROC	–	–	27%	105%
	ROS	29%	71%	55%	160%
	RCN	74%	16%	89%	187%
Nb-Elt- Pert=3	Base	0,175	0,0399	0,1126	0,0275
	ROC	–	–	30%	111%
	ROS	39%	5%	85%	165%
	RCN	40%	6%	76%	156%
Nb-Elt- Pert=5	Base	0,1471	0,0381	0,083	0,0258
	ROC	17%	–	62%	121%
	ROS	68%	5%	152%	177%
	RCN	114%	32%	146%	206%

tinents. Ces valeurs ne sont pas listées. En revanche, des améliorations significatives sont observées dans le cas des mesures MA_nxCG[10] en agrégation stricte et généralisée en appliquant l’approche orientée contenu et l’approche combinée.

Dans le cas de la tâche VVCAS, il y a une amélioration claire au niveau des mesures strictes, proportionnelle au nombre d’éléments jugés pertinents. Pour les mesures généralisées, les améliorations sont claires pour les approches ROS et RON, l’approche ROC à partir de 5 éléments pertinents.

Nous n’allons pas nous attarder sur ce point car la solution proposée est peu applicable en pratique, en effet, en fixant le nombre d’éléments jugés pertinents, on peut se ramener à parcourir un grand nombre d’éléments retournés par le système ce qui pénalise l’évaluation résiduelle (dans certains cas il n’existe pas d’éléments strictement pertinents). Nous présentons dans le tableau 4.12 la moyenne des éléments parcourus (jugés) (MEJ) pour retrouver le nombre fixé d’éléments jugés pertinents.

Nous remarquons que le nombre d’éléments parcourus est élevé (nous avons considéré toutes les requêtes y compris celles pour lesquelles on ne retrouve pas d’éléments pertinents, c’est ce qui explique les valeurs élevées des nombres d’éléments). Nous remarquons que les moyennes varient selon les tâches ainsi que les collections. Par exemple si nous considérons le cas de la tâche CO, pour

TAB. 4.12 – Moyennes des éléments jugés dans les échantillons

	MEJ (P1)	MEJ (P3)	MEJ (P5)
CO 2005	211.92	289.78	304.28
CO+S 2005	199.82	331.85	383.53
VVCAS 2005	148.41	409.16	476
CO 2006	99.04	175.83	248.95
CO+S 2006	160.28	381.81	490.54

retrouver 3 éléments pertinents, on doit parcourir en moyenne 289 éléments dans la collection 2005 alors que 175 éléments est la moyenne d'éléments parcourus dans la collection 2006, elle est de 381 pour la même collection mais pour la tâche CO+S. Nous allons alors discuter les deux paramètres (nombre d'éléments jugés et le nombre d'éléments jugés pertinents) dans la section suivante.

4.3.3 Discussion

Nous constatons d'après les expérimentations effectuées précédemment que d'une part, si on fixe le nombre d'éléments jugés, on risque de ne pas avoir des éléments pertinents pour appliquer notre approche, et d'autre part si on fixe le nombre d'éléments jugés pertinents, on risque de parcourir l'ensemble des éléments retournés par le système pour certaines requêtes. Ceci nous conduit à chercher un compromis entre les deux facteurs. De ce fait, nous proposons de fixer le nombre des éléments jugés pertinents tout en fixant un nombre maximum d'éléments parcourus qu'on ne doit pas dépasser.

Cette proposition semble la plus raisonnable puisqu'elle pourra être appliquée dans des cas réels. En effet, si on charge un utilisateur de juger un ensemble d'éléments, il lui semblera inutile de continuer à parcourir une liste d'éléments s'il retrouve un nombre suffisant d'éléments pertinents dès les premiers éléments de l'ensemble. D'autre part, un utilisateur ne peut pas dépasser un certain nombre d'éléments à parcourir car il est généralement limité par le facteur temps. Par souci de généralisation, nous n'allons pas prendre les meilleures conditions pour chaque tâche, nous essayons de trouver des conditions qui peuvent plus au moins convenir à toutes les tâches. Dans notre cas, on prend les conditions suivantes : on considère 3 éléments pertinents et on juge au plus 20 éléments. Nous avons ensuite appliqué ce choix d'échantillon pour les différentes requêtes des collections 2005 et 2006. Nous présentons dans le tableau 4.13 les résultats obtenus.

TAB. 4.13 – Résultats selon le nouvel échantillon de test pour les différentes tâches de recherche

Tâches	App	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
CO 2005	Base	0,1742	0,0398	0,0963	0,0226
	ROC	19%	24%	–	–
	ROS	48%	44%	39%	60%
	RCN	19%	30%	–	–
CO+S 2005	Base	0,1622	0,0426	0,0863	0,0267
	ROC	–	19%	6%	–
	ROS	1%	12%	30%	28%
	RCN	0%	12%	23%	17%
CO+S 2006	Base	0,1353	0,0063	0,1203	0,0035
	ROC	24%	–	26%	0%
	ROS	17%	–	14%	–
	RCN	20%	–	19%	–
VVCASS 2005	Base	0,2499	0,0476	0,1657	0,0286
	ROC	1%	–	15%	132%
	ROS	3%	0%	26%	155%
	RCN	0%	0%	22%	153%

Nous observons d’une manière générale, des améliorations claires au niveau de ManxCG et/ou MAep pour toutes les tâches et quelle que soit l’approche. Nous constatons que les améliorations obtenues pour la tâche CO ne concernent que la collection 2005 où l’approche orientée structure (ROS) s’avère la plus intéressante. Ceci montre l’intérêt de la structure pertinente comme source d’évidence (on obtient 60% d’amélioration pour les MAep stricte). Dans le cas de la tâche CO+S, des améliorations sont obtenues dans les deux collections où l’approche orientée structure est la plus efficace dans la collection 2005. Dans la collection 2006, l’approche orientée contenu s’avère plus efficace mais reste sans impact positif en considérant les MAep. Dans le cas de la tâche VVCAS, quelle que soit l’approche utilisée, on n’observe des améliorations qu’au niveau des mesures strictes. Pour la collection 2006 bien qu’on n’ait pas observée d’amélioration globale pour la mesure MAep stricte dans le cas de ROS, nous avons regardé les améliorations obtenues pour chacune de requête. Nous avons alors constaté qu’il y a plus de requêtes qui améliorent les performances que de requêtes qui les détériorent dans le cas de la tâche CO+S (52/77) et autant de requêtes améliorant les performances que celles qui les détériorent dans le cas de la tâche CO (parmi 63 requêtes ayant des éléments pertinents dans les 20 premiers jugés, 28 requêtes permettant l’amélioration des performances, 27 qui détériorent et le reste des requêtes n’apporte rien (0%)). Ces résultats préliminaires sont à prendre avec prudence car les approches sont différentes et chacune a ses paramètres spécifiques qu’il faut régler. C’est ce que

TAB. 4.14 – Comparaison des résultats du nouvel échantillon et l'échantillon fixe

Tâches	App	MANxCG [10] gen		MAep gen		MANxCG [10] stricte		MAep stricte	
		Nou- Ech	Nb- Elt=20	Nou- Ech	Nb- Elt=20	Nou- Ech	Nb- Elt=20	Nou- Ech	Nb- Elt=20
CO 2005	ROC	0.2079	0.2062	0.0493	0.0545	0.0572	0.0365	0.0122	0.011
	ROS	0.2577	0.3078	0.0572	0.071	0.1336	0.0921	0.0361	0.0288
	RCN	0.2075	0.2603	0.0518	0.0664	0.0836	0.06	0.0201	0.0206
CO+S 2005	ROC	0.1515	0.1883	0.0506	0.0542	0.0969	0.0819	0.0279	0.0258
	ROS	0,1637	0.1807	0,0475	0.0499	0,1122	0.0969	0,0341	0.0279
	RCN	0,1629	0.1877	0,0475	0.0504	0,1062	0.079	0,0312	0.0261
CO+S 2006	ROC	0.1672	0.1409	0.0052	0.0044	0.1514	0.1276	0.0035	0.003
	ROS	0.1579	0.1428	0.0042	0.0042	0.1373	0.1292	0.0029	0.0029
	RCN	0.1621	0.1467	0.0042	0.0042	0.1428	0.132	0.0031	0.0029
VVCAS 2005	ROC	0.2515	0.2156	0.044	0.0425	0.1906	0.1669	0.0663	0.0644
	ROS	0.2569	0.1914	0.0474	0.0439	0.2088	0.1475	0.073	0.0688
	RCN	0.2491	0.1898	0.0478	0.0458	0.2023	0.1536	0.0725	0.0688

nous allons faire dans les sections suivantes. Nous allons étudier ces différentes approches en détail.

Afin de montrer l'intérêt de cet échantillonnage, nous allons plutôt comparer dans le tableau 4.14 les résultats, en termes de performances directes (la comparaison considérant les améliorations n'a pas de sens car la base est différente), obtenus pour les tâches CO de la collection 2005, CO+S de la collection 2005 et 2006 et VVCAS de la collection 2005 vis-à-vis les résultats déjà obtenus en fixant le nombre d'éléments jugés à 20 (échantillon fixe).

Nous remarquons que les deux échantillons ont des comportements différents selon la mesure considérée. Les résultats obtenus avec le nouvel échantillon sont meilleurs que ceux de l'échantillon fixe, niveau des mesures strictes quelle que soient la tâche et l'approche considérées. en ce qui concerne la mesure généralisée, les résultats varient selon les tâches. Pour CO et CO+S 2005 c'est l'échantillon fixe qui l'emporte, alors que pour VVCAS 2005 et CO+S 2006 c'est le nouvel échantillon.

Nous considérons pour l'évaluation de nos différentes approches dans le reste des

expérimentations le nouvel échantillon. L'avantage de cet échantillon, qualifié d'optimal, permet d'améliorer les performances des systèmes de recherche en se basant sur un minimum d'éléments jugés.

4.4 Évaluation de la RF Orientée Contenu

Nous rappelons que cette approche consiste à enrichir le contenu de la requête en réinjectant des termes pertinents. L'objectif des expérimentations effectuées dans cette section est d'évaluer l'impact des techniques d'extraction et de pondération des termes de la requête. Nous distinguons trois méthodes de sélection/pondération des termes :

1. La première consiste à extraire et sélectionner les termes pertinents selon leur poids probabiliste-contextuel noté *Prob-Cont*, (équation 3.5). Ce poids prend en compte la probabilité de pertinence des termes sachant les éléments pertinents ainsi que leur appartenance au contexte des termes de la requête. Le poids *Prob-Cont* servira aussi à la pondération des termes dans la requête finale.
Une alternative est d'extraire et sélectionner les termes en ne tenant compte que de leur probabilité de pertinence. Les évaluations de cette alternative ont montré que ce facteur est insuffisant pour la sélection des termes pertinents [86]. Dans nos expérimentations nous n'utiliserons donc que le poids *Prob-Cont*.
2. La deuxième consiste à extraire et sélectionner les termes pertinents selon le poids ajusté *Prob-Cont-Brt* (équation 3.7). Ce poids est une combinaison du poids *Prob-Cont* avec le facteur bruit. Ce poids servira lui aussi pour la pondération des termes de la requête finale. Dans cette méthode nous utilisons la réinjection de pertinence négative.
3. La troisième alternative consiste à utiliser l'un des poids *Prob-Cont* ou *Prob-Cont-Brt* pour l'extraction et la sélection des termes pertinents. La pondération des termes de la requête finale s'effectue en appliquant le poids Pd_{freq} tenant compte de leur importance dans la collection des éléments et celle des documents (équation 3.9).

Dans ce qui suit, nous allons étudier d'abord le nombre adéquat de termes pertinents à réinjecter (section 4.4.1) en utilisant la première méthode. Une fois ce paramètre fixé, nous comparons les différentes méthodes de sélection/pondération des termes de la requête dans la section 4.4.2.

TAB. 4.15 – Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO de la collection 2005

Choix du nombre de termes	MA _{nxC} G [10] gen	MA _{ep} gen	MA _{nxC} G [10] stricte	MA _{ep} stricte
Nb-TP=1	19%	24%	–	–
Nb-TP=2	26%	19%	6%	–
Nb-TP=3	12%	13%	–	–
Nb-TP=4	11%	5%	–	–
Nb-TP=5	10%	6%	–	–
Nb-TP=6	–	5%	–	–
Nb-TP=7	–	6%	–	–
Nb-TP=8	–	5%	–	–
Nb-TP=9	–	9%	–	–
Nb-TP=10	–	–	–	–

4.4.1 Nombre de termes réinjectés

Notre objectif est de retrouver le nombre de termes qu'on doit réinjecter pour chaque tâche de recherche de chaque collection de test. Pour ce faire, nous allons varier le nombre de termes pertinents à réinjecter (*Nb-TP*) de 1 jusqu'à 10 termes. Nous présentons les résultats obtenus en appliquant la stratégie d'évaluation résiduelle en se basant sur l'échantillon présenté dans la section précédente (composé de trois éléments jugés pertinents sans dépasser les 20 éléments jugés). Nous utiliserons la première méthode pour la sélection et la pondération des termes pertinents à savoir l'utilisation du poids probabiliste contextuel *Prob – Cont* (équation 3.5).

Nous présentons les résultats obtenus pour chaque tâche de recherche CO, CO+S et VVCAS de la collection 2005 et CO et CO+S de la collection 2006.

4.4.1.1 Tâche CO

Dans le cas de la tâche de recherche CO de la collection 2005, nous remarquons dans le tableau 4.15 que les améliorations significatives sont obtenues pour les mesures généralisées en réinjectant un nombre de termes pertinents ne dépassant pas 3. Notre approche a ainsi plus d'impact en réinjectant un nombre limité de termes pertinents. Nous remarquons en outre que nous n'obtenons pas d'amélioration des mesures strictes hormis le cas de l'ajout de deux termes pertinents. Ceci ne traduit cependant pas forcément le fait que notre approche n'est pas fiable. Les valeurs présentées sont des moyennes pour toutes les requêtes.

TAB. 4.16 – Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO+S de la collection 2005

Choix du nombre de termes	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Nb-TP=1	–	19%	6%	4%
Nb-TP=2	7%	14%	17%	0%
Nb-TP=3	3%	16%	8%	18%
Nb-TP=4	3%	9%	7%	9%
Nb-TP=5	9%	11%	19%	6%
Nb-TP=6	5%	15%	6%	12%
Nb-TP=7	2%	10%	22%	2%
Nb-TP=8	13%	12%	44%	14%
Nb-TP=9	6%	11%	42%	13%
Nb-TP=10	13%	11%	46%	18%

Nos expérimentations ont montré que la réinjection de pertinence ne permet pas d'améliorer les performances de recherche dans le cas de la collection 2006. Par ailleurs les différentes mesures présentent des moyennes de toutes les requêtes. Comme nous l'avons déjà mentionné, des améliorations sont observées pour 28 requêtes vis-à-vis 27 qui détériorent les performances du système dans le cas de l'ajout d'un seul terme.

4.4.1.2 Tâche CO+S

D'après le tableau 4.16, nous remarquons que la réinjection de termes pertinents dans le cas de la tâche de recherche CO+S de la collection 2005 permet d'améliorer significativement les résultats à partir de l'ajout d'un terme et quelle que soit la mesure considérée. On constate également que l'amélioration est proportionnelle au nombre de termes ajoutés dans le cas de la mesure stricte.

Dans le cas de la collection 2006, nous remarquons dans le tableau 4.17 que les améliorations ne concernent que les MANxCG[10]. On constate des améliorations à partir de l'ajout d'un terme. Dans ce cas la réinjection de termes pertinents permet d'augmenter le nombre des éléments pertinents dans l'ensemble des premiers éléments retournés. Même si nous n'observons pas d'améliorations au niveau de MAep stricte, nous aboutissons à des résultats satisfaisants puisque notre approche permet d'augmenter significativement ($AR > 10\%$) les performances au niveau des 10 premiers éléments retournés par le

TAB. 4.17 – Impact du nombre de termes pertinents à réinjecter dans le cas de la tâche CO+S de la collection 2006

Choix du nombre de termes	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Nb-TP=1	23%	–	26%	0%
Nb-TP=2	24%	–	25%	–
Nb-TP=3	19%	–	20%	–
Nb-TP=4	18%	–	18%	–
Nb-TP=5	19%	–	20%	–
Nb-TP=6	22%	–	23%	–
Nb-TP=7	20%	–	20%	1% –
Nb-TP=8	25%	–	27%	–
Nb-TP=9	22%	–	24%	–
Nb-TP=10	19%	–	21%	–

système.

4.4.1.3 Tâche VVCAS de la collection 2005

Dans le tableau 4.18, nous remarquons d’une manière générale que les seules améliorations obtenues sont pour la mesure MAep stricte où l’amélioration est très significative (supérieure de 110%) quelque soit le nombre de termes pertinents réinjectés, aucune amélioration n’est observée dans les MANxCG. En comparant les améliorations obtenues pour les différents nombres de termes ajoutés, nous remarquons un comportement comparable quelque soit ce nombre. En outre la seule amélioration significative (15%) de la mesure MANxCG[10] stricte est obtenue dans le cas de l’ajout d’un seul terme (ce qui correspond au résultat déjà observé dans le tableau 4.13). Nous remarquons également que la taille moyenne des requêtes de la tâche VVCAS est aux alentours de 4 termes donc de petites tailles. Il est vraisemblable que l’ajout d’un nombre élevé de termes peut dégrader les performances du système. Nous discuterons de la notion de nature de requêtes dans le paragraphe suivant.

Nous rappelons que les améliorations des mesures généralisées peuvent être observées dans les cas des échantillons vus précédemment.

4.4.1.4 Discussion

Comme nous l’avons mentionné sur les différentes tâches de recherche de chaque collection, le nombre de termes adéquats à réinjecter peut être lié aux requêtes. Une requête est caractérisée par sa taille et l’ambiguïté éventuelle

TAB. 4.18 – Impact du nombre de termes pertinents à réinjecter dans le cas des requêtes VVCAS de la collection 2005

Choix du nombre de termes	MA _{nxC} G [10] gen	MAep gen	MA _{nxC} G [10] stricte	MAep stricte
Nb-TP=1	1%	–	15%	132%
Nb-TP=2	–	–	–	135%
Nb-TP=3	–	–	–	136%
Nb-TP=4	–	–	–	136%
Nb-TP=5	–	–	–	138%
Nb-TP=6	–	–	–	134%
Nb-TP=7	–	–	–	135%
Nb-TP=8	–	–	–	117%
Nb-TP=9	–	–	–	118%
Nb-TP=10	–	–	–	135%

de ses termes. Ceci est déjà confirmé en RI [160]. On se focalise dans nos travaux sur le critère *taille*. Il paraît évident que la taille de la requête influe sur le nombre adéquat de termes à réinjecter. Par exemple, l'ajout de 2 termes pertinents pour une requête composée de 7 mots clés diffère de l'ajout du même nombre de termes pour une requête composée de 2 mots clés seulement, car dans ce cas les termes ajoutés peuvent changer le sens original de la requête. Nous avons alors essayé de voir s'il existe une règle pour le choix du nombre de termes. Nous avons alors observé les 28 requêtes ayant des améliorations, de la tâche CO de la collection 2006. Comme ceci est présenté dans la figure 4.1, nous avons considéré le nombre de termes ajoutés permettant la meilleure amélioration pour chacune des requêtes.

La seule constatation qu'on a pu faire est qu'il n'existe aucun cas où le nombre de termes réinjectés est supérieur à la taille initiale de la requête. En d'autres termes : soit T la taille initiale de la requête, $Nb - TP$ le nombre de termes à réinjecter : $Nb - TP \leq T$.

Au delà de cette constatation aucune conclusion n'a pu être tirée. Nous avons par ailleurs effectué des expérimentations en faisant varier le nombre de termes rajoutés à la requête en fonction de sa taille (nombre de termes). L'idée est que le nombre de termes rajoutés ne dépasse pas la taille de la requête initiale et la requête reformulée ne dépasse pas 7 termes (pour les requêtes de taille < 7). Aucun résultat significatif n'a été observé.

Il n'y a pas un nombre de termes idéal qui sort du lot. Il semble par ailleurs que l'ajout de peu de termes, entre 1 et 3, apporte des améliorations significa-

Requêtes	Taille								
	2	3	4	5	6	7	8	9	10
295				1					
296					1				
298								1	
308		1							
310					1				
311			1						
316					2				
320							1		
325				1					
334		1							
338	1		1						
341		9							
343			2						
345						1			
352			1						
355							1		
357		2							
360					1				
365				1					
369			1						
378									10
382		2							
386	1								
391			4						
395			4						
400			4						
405				5					
Total	2	15	18	8	5	1	2	1	10

FIG. 4.1 – Nombre de termes à réinjecter en fonction de la taille des requêtes.

tives, souvent meilleurs que l'ajout de "beaucoup" de termes (au delà des trois). Nous choisissons pour le reste des expérimentations les conditions suivantes : ajout d'un seul terme dans le cas des tâches CO, CO+S de la collection 2006 et VVCAS de la collection 2005. Dans le cas de la tâche CO+S de la collection 2005, on réinjecte 3 termes pertinents.

L'ajout de peu de termes peut toutefois être justifié par la taille des éléments jugés. En effet, comme ces éléments sont strictement pertinents, ils sont donc très spécifiques ; donc souvent de petite taille.

Par ailleurs, on pourrait penser à juste titre que l'ajout de peu de termes, en particulier de 1 terme, pourrait ne pas avoir d'influence sur le processus de réinjection. En fait ceci n'est pas tout à fait vrai car au delà de l'ajout des termes, il y a également la repondération des termes de la requête initiale. Ceci peut également avoir un impact sur les résultats.

TAB. 4.19 – Impact des stratégies de sélection et pondération des termes dans le cas des requêtes CO de la collection 2005

	MA _n xCG [10] gen	MAep gen	MA _n xCG [10] stricte	MAep stricte
<i>Prob-Cont</i>	9%	6%	–	–
<i>Prob-Cont-Brt</i>	–	25%	–	–
<i>Prob-Cont/Pd_{freq}</i>	8%	19%	–	–
<i>Prob-Cont-Brt/Pd_{freq}</i>	8%	19%	–	–

4.4.2 Impact des stratégies de sélection et de pondération des termes de la requête

Dans cette section nous étudions l'impact des différentes stratégies de sélection et pondération des termes de la requête. Nous rappelons que nous distinguons les stratégies suivantes :

- La première est celle que nous avons avant appliquée dans la section précédente en attribuant les mêmes poids *Prob-Cont* (équation 3.5) pour la sélection des termes ainsi que pour la pondération des termes de la requête finale.
- La deuxième consiste à sélectionner et pondérer les termes pertinents selon le poids *Prob-Cont-Brt* (équation 3.7).
- La troisième est composée de deux stratégies. Elle consiste à utiliser l'un des poids *Prob-Cont* ou *Prob-Cont-Brt* pour l'extraction et la sélection des termes pertinents. La pondération des termes s'effectue en appliquant le poids *Pd_{freq}* (équation 3.9). Les deux types de stratégies seront désignées par : *Prob-Cont/Pd_{freq}* et *Prob-Cont-Brt/Pd_{freq}*.

Nous allons procéder par type de tâche de recherche pour analyser nos expérimentations.

4.4.2.1 Tâche CO

Comme nous l'avons mentionné précédemment, nous ajoutons dans ce cas un seul terme pertinent.

Nous remarquons dans le tableau 4.19 que les différentes stratégies permettent une amélioration significative de MAep généralisée dans la collection 2005, et notamment, lorsque le facteur bruit est considéré (*Prob-Cont-Brt* : 25%). Cependant, l'intérêt du facteur bruit est relatif puisqu'il dégrade les per-

TAB. 4.20 – Impact des stratégies de sélection et pondération des termes dans le cas de la tâche CO+S de la collection 2005

	MAxCG [10] gen	MAep gen	MAxCG [10] stricte	MAep stricte
<i>Prob-Cont</i>	5%	16%	8%	18%
<i>Prob-Cont-Brt</i>	–	9%	–	–
<i>Prob-Cont/Pd_{freq}</i>	–	6%	–	–
<i>Prob-Cont-Brt/Pd_{freq}</i>	–	6%	–	–

TAB. 4.21 – Impact des stratégies de sélection et pondération des termes dans le cas de la tâche CO+S de la collection 2006

	MAxCG [10] gen	MAep gen	MAxCG [10] stricte	MAep stricte
<i>Prob-Cont</i>	24%	–	28%	–
<i>Prob-Cont-Brt</i>	19%	–	18%	–
<i>Prob-Cont/Pd_{freq}</i>	11%	–	10%	–
<i>Prob-Cont-Brt/Pd_{freq}</i>	11%	–	10%	–

formance au niveau des 10 premiers éléments.

Nous remarquons en outre que quelque soit le poids utilisé pour la sélection des termes, on retrouve les mêmes résultats en considérant une pondération selon Pd_{freq} . Comparée à *Prob-Cont-Brt*, *Prob-Cont-Brt/Pd_{freq}* apporte une amélioration au niveau des 10 premiers éléments, son analogue *Prob-Cont/Pd_{freq}*, comparé à *Prob-Cont* augmente la MAep généralisée (19% par rapport à 6%). Les résultats montrent alors l'intérêt relatif de cette pondération mais laissent entendre, comme on peut s'y attendre, que les poids des termes de la requête influent directement sur les performances du système.

Dans le cas de la collection 2006, on ne trouve pas d'améliorations globales.

4.4.2.2 Tâche CO+S

Comme nous l'avons mentionné précédemment, nous ajoutons 3 termes pertinents à la requête initiale dans le cas de la collection 2005 et 1 seul terme dans le cas de la collection 2006. Les tableaux 4.20 et 4.21 listent les résultats obtenus en appliquant les différentes variantes de sélection/pondération des termes. Nous remarquons pour COS2005 une amélioration de la MAep généralisée pour toutes les méthodes avec une légère préférence pour *Prob-Cont*, qui en outre améliore les mesures strictes et généralisées au niveau des 10 premiers éléments

TAB. 4.22 – Impact des stratégies de sélection et pondération des termes dans le cas de la tâche VVCAS de la collection 2005

	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
<i>Prob-Cont</i>	1%	–	15%	132%
<i>Prob-Cont-Brt</i>	–	–	5%	132%
<i>Prob-Cont/Pd_{freq}</i>	–	–	5%	126%
<i>Prob-Cont-Brt/Pd_{freq}</i>	–	–	5%	124%

et la MAep. Ceci est plus ou moins confirmé pour la collection 2006. En effet, on observe des améliorations significatives obtenues par les différentes stratégies. Elles ne concernent que les mesures MANxCG[10] (stricte et généralisée). Ceci nous permet de conclure que pour ce type de requêtes,

- Le facteur bruit (*Prob-Cont-Brt*) n’a pas d’impact réel sur la sélection des termes.
- La pondération en considérant l’importance des termes dans la collection des éléments et celle des documents ne permet pas de mieux exprimer les degrés d’importance des termes car elle ne tient pas compte de la présence des termes dans les documents pertinents. En d’autres termes le poids probabiliste contextuel traduit mieux l’importance des termes.

4.4.2.3 Tâche VVCAS

Dans le cas des requêtes VVCAS de la collection 2005, nous avons testé les différentes variantes en réinjectant un seul terme. Le tableau 4.22 liste les différents résultats. Nous remarquons que la stratégie utilisant les poids calculés en fonction du bruit (*Prob-Cont-Brt*), n’apporte rien.

La pondération des termes de la requête en considérant leur importance dans la collection des éléments et celle des documents n’apporte pas d’améliorations dans les deux cas de sélection des termes. En général, nous remarquons que quelle que soit la stratégie appliquée, les résultats affirment l’intérêt de notre approche (AR(MAep strict > 120%).

4.4.3 Bilan

En conclusion, nous avons montré l’intérêt de notre approche orientée contenu basée sur la distribution des termes dans les éléments pertinents et sur la no-

tion de contexte pour l'extraction et la sélection des termes à réinjecter. Nous avons montré en occurrence, que les indicateurs considérés dans la pondération ($tf - ief - idf$) ne traduisent pas bien les degrés d'importance des termes. Nous avons montré également que la réinjection de pertinence négative n'a pas un impact remarquable sur la sélection des termes.

En outre, nous avons étudié le choix du nombre de termes à réinjecter qui influe directement sur les performances de notre approche. La seule conclusion que nous avons pu tirer est que le nombre de termes réinjectés ne doit pas dépasser la taille initiale de la requête, il est situé entre 1 et 3 termes. Ceci est loin des 20 termes souvent utilisés en RI. Ceci peut provenir comme nous l'avons mentionné de la taille des éléments considérés. En effet, nous pensons que ces éléments manquent de diversité au niveau de leurs termes pour pouvoir dégager plusieurs bons termes à rajouter à la requête.

Nous constatons de manière générale, que la stratégie *Prob - Cont* permet d'améliorer dans la majorité des tâches les MANxCG et la MAep. Ceci n'exclut pas le fait qu'elle puisse aussi être la cause de dégradation de performances pour certaines requêtes. Ceci a été déjà observé en RI de manière générale et on a considéré que ce problème revient à la nature des requêtes dites difficiles. En effet, Buckley dans [27] a essayé de classer les requêtes difficiles selon 10 catégories différentes. Chaque catégorie de requête doit être adaptée à une méthode différente de traitements.

4.5 Évaluation de la reformulation Orientée-Structure

4.5.1 Nombre adéquat de structures à réinjecter

Par analogie à la reformulation orientée contenu, nous allons tester dans cette approche le nombre adéquat de structures à réinjecter pour chaque tâche de recherche de chaque collection. Pour ce faire, nous allons varier le nombre de structures pertinentes ($Nb-Str$) à réinjecter de 1 à 3. Le tableau 4.23 présente l'impact du nombre de structures à réinjecter pour les tâches CO, CO+S et VV-CAS de la collection 2005 et CO+S de la collection 2006. Le premier résultat intéressant qu'on observe est l'impact positif qu'apporte la réinjection de structure. Elle est très significative dans le cas des requêtes non structurées. On constate une stabilité des performances en réinjectant une seule structure pertinente dans le cas des tâches CO+S et VVCAS de la collection 2005.

TAB. 4.23 – Impact du nombre de structures pertinentes à réinjecter dans le cas des tâches CO, CO+S et VVCAS de la collection 2005 et la tâche CO+S de la collection 2006

		MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
<i>CO 2005</i>	Nb-Str=1	48%	44%	39%	60%
	Nb-Str=2	49%	44%	40%	60%
	Nb-Str=3	50%	44%	44%	62%
<i>CO+S 2005</i>	Nb-Str=1	1%	12%	30%	28%
<i>VVCAS 2005</i>	Nb-Str=1	3%	0%	26%	155%
<i>CO+S 2006</i>	Nb-Str=1	17%	–	14%	–
	Nb-Str=2	3%	–	7%	–
	Nb-Str=3	4%	–	8%	11%

Plus précisément, on observe des améliorations significatives de toutes les mesures considérées dans le cas des requêtes de type CO (AR > 40%). Ceci prouve d'une part l'intérêt de la réinjection des structures pertinentes et d'autre part l'efficacité de notre algorithme d'extraction des structures pertinentes. La légère croissance de l'AR en fonction du nombre de structures réinjectées est conforme avec nos statistiques présentées dans le chapitre précédent (section 3.4.1).

En outre la comparaison des requêtes CO reformulées par réinjection de structures pertinentes avec les requêtes CO+S (où l'utilisateur spécifie le type d'éléments répondant à son besoin) a montré :

- dans le cas de l'ajout d'une structure, 40% des requêtes reformulées contiennent la condition de structure exprimée dans les requêtes CO+S.
- dans le cas de l'ajout de deux structures pertinentes, 60% des requêtes reformulées contiennent la ou les conditions de structures exprimées dans les requêtes CO+S.
- dans le cas de l'ajout de trois structures 100% des requêtes reformulées contiennent la ou les conditions de structures exprimées dans les requêtes CO+S.

Dans le cas de la collection 2006 et de l'ajout d'une, de deux ou de trois structures pertinentes, les requêtes reformulées contenant la ou les conditions de structures exprimées dans les requêtes CO+S sont à l'entour de 50%.

Dans les deux collections nous obtenons des résultats en terme de performance plus importants que ceux obtenus dans la tâche CO+S. Ceci peut être expliqué par le fait que l'utilisateur ne connaît pas forcément le type d'éléments répondant à ses besoins. Cette idée est déjà prouvée dans [193].

Dans le cas des tâches CO+S et VVCAS de la collection 2005, on obtient des améliorations significatives en considérant les mesures en agrégation stricte

(notamment la MAep stricte).

Dans le cas de la tâche CO+S de la collection 2006, des améliorations significatives de MANxCG[10] sont observées en réinjectant une seule structure. Alors que l'ajout de 3 structures pertinentes permet d'améliorer la MAep stricte. Ceci peut nous renseigner sur l'aspect diversifié des structures existantes dans la collection 2006. La prise en compte de plus de structures ne permet pas forcément l'augmentation de l'amélioration relative puisque les structures de la collection 2006 sont peu reliées sémantiquement par des relations de hiérarchie (comme par exemple : *collectionlink*, *section*, *unknownlink*, *item*..).

On observe clairement que la réinjection d'une structure permet d'améliorer de manière significative les résultats. Les autres résultats obtenus par la réinjection de 2 et 3 structures pertinentes, ne sont pas statistiquement meilleurs que le premier résultat (ajouter 1 seule structure).

Comme nous l'avons détaillé dans le chapitre 3, la réinjection de structure peut être effectuée en ne spécifiant que le type de structure désiré par l'utilisateur ou en spécifiant tout le chemin. Nous présentons la comparaison de ces deux aspects dans les paragraphes suivants.

4.5.2 Réinjection de la balise ou du chemin

Notre objectif est de comparer les différentes méthodes d'extraction et de réinjection des structures pertinentes. Nous distinguons les 4 méthodes suivantes :

- Une première est celle que nous avons utilisée précédemment, elle consiste à réinjecter la dernière balise (c'est dire l'élément cible *Ec*) de la structure extraite en appliquant l'algorithme SCA. Cette stratégie est désignée par *SCA*.
- Une seconde consiste aussi à réinjecter l'élément cible extrait selon la méthode de classification (section 3.4.4 du chapitre 3). Cette stratégie est désignée par *Ec – class*.
- La troisième méthode consiste à réinjecter toute la structure (en spécifiant toutes les balises intermédiaires) extraite par la méthode de classification (section 3.4.4 du chapitre 3). Cette stratégie est désignée par *Ch – Spes*.
- La quatrième méthode consiste à réinjecter un *chemin générique* présenté dans la section 3.4.4 du chapitre 3, c'est à dire sans spécifier toutes les balises intermédiaires de la structure pertinente. Cette stratégie est désignée par *Ch – Gen*.

Le tableau 4.24 liste les résultats obtenus. Nous remarquons que la réinjection de l'élément cible (une balise) apporte une meilleure performance, quelle que

TAB. 4.24 – Réinjection de structure (Element cible, Chemin spécifique et Chemin générique)

		MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
CO 2005	<i>SCA</i>	48%	44%	39%	60%
	<i>Ec – Class</i>	57%	40%	44%	60%
	<i>Ch – Spes</i>	20%	38%	–	–
	<i>Ch – Gen</i>	20%	36%	–	–
CO+S 2005	<i>SCA</i>	1%	12%	30%	28%
	<i>EC – Class</i>	1%	11%	30%	28%
	<i>Ch – Spes</i>	–	–	–	–
	<i>Ch – Gen</i>	–	–	–	–
VVCAS 2005	<i>SCA</i>	3%	0%	26%	155%
	<i>EC – Class</i>	–	–	1%	152%
	<i>Ch – Spes</i>	–	–	–	68%
	<i>Ch – Gen</i>	6%	–	10%	151%
CO+S 2006	<i>SCA</i>	17%	–	14%	–
	<i>EC – Class</i>	11%	–	–	–
	<i>Ch – Spes</i>	5%	–	0%	–
	<i>Ch – Gen</i>	5%	–	–	–

soit la tâche, avec un bénéfice clair (soit en MAep ou en MANxCG[10]) pour les tâches CO et CO+S de la collection 2005. De ce fait, l'ajout de la balise (élément cible) est plus à même d'améliorer les performances quelle que soit la tâche.

En ce qui concerne la réinjection du chemin spécifique et du chemin générique, nous notons une amélioration au niveau de COS2005, mais ceci ne se retrouve pas dans les autres tâches.

4.5.3 Bilan

Le premier bilan que l'on peut faire à partir de ces expérimentations est qu'il est important de considérer la structure comme une source d'évidence. De plus nous avons constaté que l'ajout de la balise cible, de préférence une seule balise, est plus à même d'apporter des améliorations comparativement à l'ajout de plusieurs balises ou du chemin.

4.6 Évaluation de la reformulation Orientée-Contenu & Structure

Nous avons défini 3 formes de combinaison des deux approches présentées précédemment : une forme naïve, une forme tenant compte de la "sémantique" des balises sous deux versions (en utilisant l'équation 3.14 (Sémantique1) et l'équation 3.15 (Sémantique2)) et une forme flexible. Nous rappelons que :

- la combinaison naïve (section 3.5.1 du chapitre 3) consiste tout simplement à réinjecter les termes pertinents ainsi que les structures pertinentes issus des deux approches appliquées indépendamment. La combinaison se fait au niveau de la réécriture.
- la combinaison avec dépendance sémantique (section 3.5.2 du chapitre 3) consiste à considérer la sémantique des éléments jugés pertinents pour la sélection des termes. Nous distinguons deux versions de cette méthode : la première considère les poids des structures pertinentes dans la sélection des termes pertinents (elle est désignée par Sémantique 2) et la seconde ne considère que le poids d'extraction des termes calculé selon l'approche orientée contenu (Sémantique 1).
- la combinaison flexible (section 3.5.3 du chapitre 3) consiste à distribuer les termes pertinents selon leur degré d'appartenance aux différentes contraintes structurelles spécifiées dans la requête.

Les résultats obtenus sont détaillés dans les paragraphes suivants pour chaque tâche de recherche.

Pour étudier l'impact de cette approche, nous utilisons le même nombre de termes fixés pour chaque tâche de recherche que ceux utilisés dans la comparaison des stratégies de l'approche orientée contenu. Les termes sélectionnés seront combinés avec trois structures génériques extraites en appliquant l'algorithme SCA dans le cas des tâches CO 2005 et CO+S 2006. Nous choisissons trois structures afin de pouvoir appliquer la combinaison flexible ; dans le cas d'une seule structure, l'approche flexible n'a pas d'impact. Dans le cas des tâches CO+S et VVCAS de la collection 2005, nous réinjectons une seule structure puisque les résultats de la section précédente, ont montré une stabilisation des améliorations en réinjectant une seule structure. Nous présentons les différentes méthodes de combinaison pour chaque tâche de recherche. Les résultats issus de ces méthodes seront comparés aux meilleurs résultats obtenus en appliquant l'approche orientée contenu et l'approche orientée structure.

TAB. 4.25 – Reformulation de requêtes par combinaison dans le cas de la tâche CO de la collection 2005

	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Naive	23%	33%	–	0%
Semantique1	15%	33%	–	–
Semantique2	16%	22%	–	–
Flexible	27%	31%	–	–
Orientée- Structure	50%	44%	44%	62%
Orientée- Contenu	9%	6%	–	–

4.6.1 Tâche CO

D’après le tableau 4.25, nous remarquons que la combinaison des deux sources d’évidence (contenu et structure) permet des améliorations significatives quelle que soit la méthode de combinaison pour les mesures généralisée. Si nous comparons les différentes méthodes, nous constatons que les meilleures améliorations sont obtenues en appliquant la méthode naïve et la méthode flexible. Ceci peut être expliqué par le fait que ces deux méthodes ne pénalisent pas certains termes qui n’appartiennent pas aux structures sélectionnées comme pertinentes.

Ces résultats restent moins bons en les comparant avec ceux obtenus en appliquant l’approche orientée structure mais ils sont meilleurs comparativement à l’approche orientée contenu seulement.

4.6.2 Tâche CO+S

On observe d’après les tableaux 4.26 et 4.27 que toutes les combinaisons

permettent d’améliorer de manière significative les performances au niveau de la MAep ou/et au niveau de la MANxCG[10]. Pour les deux collections aucune méthode ne surpasse les autres de manière claires. On constate une amélioration plus importante au niveau de la mesure stricte. Ceci est logique car l’ajout de la structure restreint le champ de la recherche.

Comparés aux résultats déjà obtenus par les deux approches orientée contenu et orientée structure, nous remarquons que la combinaison permet, dans la majorité des cas, une meilleure amélioration pour les deux collections. En particulier au niveau de la mesure MAep stricte dans le cas de la collection 2006.

TAB. 4.26 – Reformulation de requêtes par combinaison dans le cas de la tâche CO+S de la collection 2005

	MAnxCG [10] gen	MAep gen	MAnxCG [10] stricte	MAep stricte
Naive	–	11%	–	13%
Semantique1	7%	14%	20%	28%
Semantique2	–	11%	–	13%
Flexible	0%	11%	10%	16%
Orientée- Structure	1%	12%	30%	28%
Orientée- Contenu	3%	16%	8%	18%

TAB. 4.27 – reformulation de requêtes par combinaison dans le cas de la tâche CO+S de la collection 2006

	MAnxCG [10] gen	MAep gen	MAnxCG [10] stricte	MAep stricte
Naive	29%	–	32%	26%
Semantique1	18%	–	24%	26%
Semantique2	29%	–	32%	26%
Flexible	32%	–	35%	31%
Orientée- Structure	4%	–	8%	11%
Orientée- Contenu	24%	–	28%	–

TAB. 4.28 – Reformulation de requêtes par combinaison dans le cas de la tâche VVCAS de la collection 2005

	MAncCG [10] gen	MAep gen	MAncCG [10] stricte	MAep stricte
Naive	–	4%	14%	153%
Seman- tique1	1%	4%	30%	154%
Semantique2	–	4%	14%	153%
Flexible	–	4%	14%	153%
Orientée- Structure	3%	0%	26%	155%
Orientée- Contenu	1%	–	15%	132%

4.6.2.1 Tâche VVCAS

Dans le cas de la tâche VVCAS de la collection 2005, nous remarquons d’après le tableau 4.28 que toutes les méthodes appliquées permettent des améliorations semblables et significatives en considérant l’agrégation stricte, notamment les MAep où $AR > 150\%$, pour la méthode *Sémantique 1* comme dans le cas de la tâche CO+S 2005, ce qui peut être expliqué par le nombre de structures pertinentes limité (=1) dans les deux cas.

Comparée aux approches précédentes orientée structure et orientée contenu, l’approche combinée permet des améliorations similaires à celle en réinjectant une structure pertinente avec une légère amélioration au niveau des MAep généralisée.

En général, la combinaison est plus bénéfique que les deux approches précédentes.

4.6.3 Conclusion

En conclusion nous constatons différentes conséquences de la cohabitation des deux sources d’évidence selon le type de tâche de recherche.

- la combinaison des deux sources d’évidence permet de renforcer les améliorations observées pour les deux approches (orientée structure et orientée contenu) en particulier, pour les tâches CO+S (2005 et 2006) et VVCAS (2005).
- dans le cas de la tâche CO l’ajout de la structure seule reste plus important que la combinaison.

Nous remarquons en outre que dans le cas de la réinjection d’un nombre limité de structures pertinentes (les performances se stabilisent en réinjectant une seule structure), la combinaison avec dépendance contextuelle (*Sémantique 1*)

s'avère légèrement plus intéressante.

Dans le reste des expérimentations nous considérons la reformulation combinée selon la méthode flexible.

4.7 Autres études qualitatives

Dans cette section nous allons étudier d'une part l'impact des jugements de pertinence et d'autre part celui des résultats de base.

4.7.1 Impact des jugements de pertinence

Durant les différentes expérimentations, nous nous sommes basés sur un échantillon composé par des éléments jugés strictement pertinents qui traduisent les résultats les plus rigoureux du point de vue de l'utilisateur (éléments très spécifiques et très exhaustifs).

Des éléments pertinents selon la fonction d'agrégation généralisée peuvent aussi répondre à des besoins utilisateurs moins stricts. Nous allons alors tester l'influence d'un échantillon composé d'éléments pertinents d'une manière généralisée sur les approches de reformulation.

Pour ce faire, nous allons considérer une exhaustivité plus large ≥ 1 et une spécificité ≥ 0.1 . Nous observons les impacts pour les différentes tâches de chaque collection en appliquant la réinjection de pertinence combinée.

Nous présentons dans le tableau 4.29 les nouvelles améliorations (selon le jugement généralisés) ainsi les celles obtenues avec un jugement stricte désignées par *jug – strict*. Nous constatons que le jugement généralisé permet d'augmenter les taux d'amélioration dans le cas des tâches de recherche de la collection 2005, notamment au niveau de la MANxCG[10] généralisée dans le cas de la tâche VVCAS pour laquelle nous n'avons pas obtenu d'amélioration en se basant sur des éléments strictement pertinents. Nous constatons alors que le choix des éléments pertinents a un impact net sur les performances de la réinjection de pertinence.

De plus, même si nous observons des améliorations plus importantes, les résultats obtenus ne sont pas forcément meilleurs, nous remarquons que les résultats de base sont en général moins bons que ceux en utilisant le jugement strict.

Dans le cas de la tâche CO+S de la collection 2006, les résultats sont moins bons puisque tous les éléments pertinents de la collection 2006 sont d'exhaustivité égale à 2, donc la généralisation porte seulement sur la spécificité des éléments, ce qui peut expliquer le comportement différent de celui dans la col-

TAB. 4.29 – Réinjection de pertinence basée sur un jugement de pertinence généralisé

	MA_nxCG [10] gen	MAep gen	MA_nxCG [10] stricte	MAep stricte
Base (CO 2005)	0,1890	0,0457	0,1268	0,0254
CO 2005	24%	–	48%	172%
CO 2005 (jug-strict)	27%	31%	–	–
Base (CO+S 2005)	0,2477	0,0500	0,1042	0,0218
CO+S 2005	29%	22%	26%	–
CO+S 2005 (jug-strict)	0%	11%	10%	16%
Base (VVCAS 2005)	0,1890	0,0457	0,1268	0,0254
VVCAS 2005	24%	–	48%	172%
VVCAS 2005 (jug-strict)	–	4%	14%	153%
Base (CO+S 2006)	0,1148	0,0054	0,1017	0,0030
CO+S 2006	9%	–	15%	0%
CO+S 2006 (jug-strict)	32%	–	35%	31%

lection 2005.

Nous pouvons conclure que les améliorations dépendent aussi du type de jugement de pertinence considéré.

4.8 Autres applications de la Réinjection de pertinence

4.8.1 Application de plusieurs itérations de réinjection

Comme nous l'avons présenté dans le deuxième chapitre, la réinjection de pertinence peut se faire en une ou plusieurs itérations. Dans ce paragraphe nous allons tester la réinjection de pertinence combinée flexible en appliquant 2 et 3 itérations. Puisqu'on utilise la réinjection résiduelle, on ne peut pas aller plus loin dans le nombre d'itérations sinon, on pourra se retrouver avec des valeurs trop faibles ou même nulles des MAep. Les tableaux 4.30 et 4.31 présentent respectivement les améliorations relatives obtenues dans la 2^{ème} et 3^{ème} itération.

Nous constatons clairement que les itérations multiples sont souvent intéressantes dans le cas où nous n'obtenons pas d'amélioration au cours de la première itération. Ceci est net dans le cas de la tâche CO de la collection 2006 et sur

TAB. 4.30 – Réinjection de pertinence en 2 itérations

	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Base (CO 2005)	0,2001	0,044	0,1089	0,0372
CO 2005	–	1%	–	9%
Base (CO+S 2005)	0,1497	0,0479	0,0612	0,0191
CO+S 2005	6%	–	13%	–
Base (VVCAS 2005)	0,2116	0,0487	0,1365	0,0264
VVCAS 2005	–	–	–	–
Base (CO 2006)	0,1421	0,0035	0,1346	0,003
CO 2006	23%	77%	20%	67%
Base (CO+S 2006)	0,1434	0,0049	0,1304	0,0037
CO+S 2006	–	14%	–	5%

TAB. 4.31 – Réinjection de pertinence en 3 itérations

	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
Base (CO 2005)	0,1481	0,0418	0,0609	0,0183
CO05	24%	8%	–	98%
Base (CO+S 2005)	0,168	0,422	0,0795	0,0174
COS05	–	–	–	26%
Base (VVCAS 2005)	0,2497	0,0462	0,2048	0,0272
VVCAS05	–	1%	–	–
Base (CO 2006)	0,1526	0,0041	0,1451	0,0038
CO06	4%	49%	0%	37%
Base (CO+S 2006)	0,1202	0,0048	0,1089	0,0035
CO+S06	46%	25%	49%	29%

TAB. 4.32 – Réinjection de pertinence "aveugle"

	MANxCG [10] gen	MAep gen	MANxCG [10] stricte	MAep stricte
CO05-Av	19%	–	68%	–
COS05-Av	–	3%	3%	–
VVCAS05-Av	21%	3%	71%	38%
COS06-Av	–	–	1%	–

les MAep stricte de la tâche CO de la collection 2005 où l'amélioration atteint 98% à la troisième itération. On obtient des résultats similaires pour la tâche CO+S de la collection 2006.

4.8.2 Utilisation de la réinjection de pertinence "aveugle"

Nous avons montré l'intérêt de nos différentes approches en mode interactif : l'utilisateur intervient pour un jugement de pertinence. Or, parmi les techniques qui sont souvent utilisées pour améliorer directement les performances d'un système il y a la réinjection "aveugle" aussi connue sous le nom de "*blind relevance feedback*". Dans nos expérimentations nous avons considéré les 3 premiers éléments comme pertinents et nous avons appliqué la reformulation combinée flexible avec les mêmes nombres de termes et de structures que précédemment. Le résultat après reformulation est comparé directement avec le résultat de base sans aucun prétraitement (blocage ou résiduel).

Le tableau 4.32 présente les résultats de la réinjection aveugle désignée par *Av*.

Dans le cas de la tâche CO de la collection 2005, nous remarquons d'après le tableau 4.32 que la reformulation aveugle permet une amélioration significative des MANxCG[10] notamment en agrégation stricte (AR=68%).

Dans le cas de la tâche CO+S des collections 2005 et 2006 les améliorations sont rarement observées et restent non significatives quand elles existent.

Des améliorations significatives des mesures strictes sont observées dans le cas de la tâche VVCAS soit 71% en MANxCG[10].

La réinjection aveugle peut être bénéfique dans certains cas.

4.9 Bilan

4.9.1 Résumé

Dans ce chapitre nous avons étudié différents aspects de la réinjection de pertinence. Outre l'évaluation de nos différentes approches : orientée contenu, orientée structure et combinée, nous avons testé d'autres modes d'application de la réinjection de pertinence à savoir la réinjection en plusieurs itérations et la réinjection aveugle.

Nous récapitulons ici les stratégies adéquates pour chaque tâche de chaque collection.

- Dans le cas de la tâche CO de la collection 2005, les meilleures améliorations sont obtenues en appliquant la réinjection de 1 à 3 structures pertinentes. En ce qui concerne la tâche CO de la collection 2006, les améliorations sont obtenues pour toutes les mesures considérées à partir de la deuxième itération. Notons qu'on obtient 67% d'amélioration pour la MAep stricte dans le cas de la collection 2006 (en appliquant deux itérations).
- Dans le cas de la tâche CO+S les meilleures améliorations sont obtenues en appliquant l'approche combinée. Quelle que soit la méthode utilisée, nous obtenons des améliorations significatives notamment en agrégation stricte. Pendant notre participation officielle dans la campagne d'INEX 2006, nous avons obtenu la meilleure amélioration de MAep généralisée (42%) vis-à-vis d'autres participations, en appliquant la combinaison naïve sur les résultats de base fournis par le système XFIRM officiellement.
- De même pour la tâche VVCAS, les meilleures améliorations sont obtenues grâce à la combinaison des sources d'évidence. Elle est plus bénéfique en tenant compte de la sémantique des éléments lors de l'extraction des termes pertinents.

Afin de mettre en relief nos résultats sur la base des indications d'INEX, nous avons listé dans le tableau 4.33 les résultats obtenus en appliquant pour chaque tâche la stratégie la plus adéquate en une seule itération. Nous considérons 3 termes pertinents et 3 structures pertinentes à réinjecter à la requête initiale dans le cas de l'approche combinée. En ce qui concerne l'approche orientée structure appliquée à la tâche CO de la collection 2005, nous considérons 3 structures pertinentes à réinjecter. Nous utilisons le protocole d'évaluation de la campagne INEX. Nous gardons les mêmes paramètres de base (résultats de

TAB. 4.33 – Evaluation selon le protocole d’INEX

	MANxCG [50] gen	MAep gen	MANxCG [50] stricte	MAep stricte
CO05	53%	49%	12%	–
COS05	46%	47%	8%	5%
VVCAS05	8%	–	35%	35%
COS06	6%	–	11%	6%

base, jugement de pertinence et échantillon).

Pour pouvoir observer l’impact de la reformulation, nous présentons la MANxCG[50] puisque avec la stratégie ”*freezing*”, on garde les 20 premiers éléments des résultats de base.

Nous pourrions alors conclure clairement, à partir du tableau 4.33 que nos approches permettent d’améliorer significativement quelle que soit la tâche, les performances du système de recherche au niveau des 50 premiers éléments retournés. La MAep généralisée est nettement améliorée dans le cas des tâches CO et CO+S de la collection 2005 alors que l’amélioration de la MAep stricte est très significative dans le cas de la tâche VVCAS 2005, ceci confirme ce que nous avons conclu pendant nos évaluations avec la stratégie résiduelle.

4.9.2 Étude comparative

L’objectif de cette section est de confronter nos résultats précédemment obtenus avec ceux obtenus par les participants d’INEX 2005 et 2006. Nous présentons dans les tableaux 4.34, 4.35 et 4.36, les classifications des participants selon la mesure MAep généralisée (utilisée dans les deux campagne 2005 et 2006). Ces tableaux correspondent respectivement à la tâche CO, la tâche CO+S de la collection 2005 et la tâche CO de la collection 2006. La tâche VVCAS n’est pas présentée puisque seule notre participation est signalée. Dans la collection 2006, la tâche CO+S est confondue avec CO.

Nous remarquons dans le cas de la tâche CO 2005, que la réinjection de structures pertinentes permet d’obtenir de meilleurs résultats que les résultats officiels. Les résultats obtenus aux deuxième et troisième rang sont basés sur la technique de réordonnement des éléments [175]. La participation de *Cirquid Project* [136] consiste à réinjecter le nom du journal auquel appartiennent des éléments pertinents et le nom de leur balise. La deuxième participation de *Max-Planck-Institut fuer Informatik* [175] consiste à réinjecter l’ancêtre et un descendant des éléments pertinents. Quant à la participation de *IBM Haifa Research Lab* [132], elle basée sur l’adaptation de l’algorithme de Rocchio.

Nous constatons ainsi l’apport effectif de la structure comme étant une nou-

Rang	Participant	MAep généralisée
1	IRIT	49%
2	Max-Planck-Institut fuer Informatik	13%
3	Max-Planck-Institut fuer Informatik	8%
4	Cirquid project	7%
5	Cirquid project	7%
6	Cirquid project	6%
7	Max-Planck-Institut fuer Informatik	5%
8	Max-Planck-Institut fuer Informatik	3%
9	IBM Haifa Research Lab	2%
10	IBM Haifa Research Lab	2%

TAB. 4.34 – Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2005 dans le cas de la tâche CO

Rang	Participant	MAep généralisée
1	IRIT (officiel)	182%
2	IRIT (officiel)	182%
3	IRIT (officiel)	182%
4	IRIT	47%
5	Cirquid project	4%
6	...	-

TAB. 4.35 – Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2005 dans le cas de la tâche CO+S

velle source d'évidence par rapport aux autres type de contraintes réinjectées (ancêtre, descendant, nom du journal, type de balise, ou du contenu extrait selon Rocchio).

Dans le cas des requêtes CO+S, notre participation officielle dans laquelle nous avons appliqué la réinjection combinée permet une meilleure amélioration (182%). Comparée à celle que nous avons obtenue précédemment (47%), la différence est justifiée par le fait qu'on se base sur des résultats de base différents. Les deux résultats sont nettement meilleurs que ceux obtenus par les autres participations, où l'amélioration devient négative à partir du sixième rang.

Notre participation officielle dans INEX 2006 correspond à l'amélioration des requêtes de type CO+S, la première ligne donne les résultats obtenus par réinjection combinée. L'amélioration est significative et confirme les résultats précédents. Les deux dernières lignes correspondent à la réinjection du contenu, elle permet d'obtenir des améliorations significatives (24% et 18%) mais moins importante que celle obtenue par la participation de *Max-Planck-Institut fuer Informatik* (25%) basée sur la réinjection de l'ancêtre et du descendant de l'élément pertinent. Rappelons que nos expérimentations présentées dans ce rapport ne permettent pas d'obtenir des améliorations sur cette tâche et ne

Rang	Participant	MAep généralisée
1	IRIT(officiel)	42%
2	Max-Planck-Institut fuer Informatik	25%
3	IRIT(officiel)	24%
4	IRIT(officiel)	18%
5	...	-

TAB. 4.36 – Classement de notre système parmi les résultats officiels de la campagne d'évaluation INEX 2006 dans le cas de la tâche CO+S

sont donc pas présentées dans ce tableau.

D'une manière générale, la réinjection de structure est la meilleure technique pour améliorer les requêtes non structurées (CO), alors que pour améliorer les requêtes structurées, la réinjection combinée est plus bénéfique.

4.9.3 Conclusion

Les principales conclusions que l'on peut tirer de toutes les expérimentations sont les suivantes :

- La réinjection de pertinence orientée contenu par ajout des termes pertinents comme nous l'avons proposée, nous a permis d'observer des améliorations significatives dans toutes les tâches, même quand ces améliorations ne sont pas significatives sur l'ensemble des requêtes, comme dans le cas des collections 2006. De plus nous préconisons l'ajout d'un nombre limité de termes dans le cas où les éléments sont très spécifiques.
- Une autre conclusion concerne l'ajout de la structure. Nous avons montré qu'il existe bien des structures pertinentes et que la prise en compte de cette source d'évidence a montré son intérêt quelle que soit la tâche de recherche et la collection considérée.
- Une troisième conclusion est celle qui concerne la combinaison des sources d'évidence. Quelles que soient les sources d'évidence, concernant les indicateurs de calcul du degré d'importance de terme (approche orientée contenu) ou celles liées au type de l'information (textuelle et structurale exprimée à travers l'approche combinée), nos expériences ont confirmé la robustesse de la cohabitation de différentes sources d'évidences.
- Enfin, nous soulignons que ces résultats sont le fruit de dizaine d'expérimentations, les quelques conclusions que nous avons tirées confirme l'intérêt de la reformulation par réinjection de structure et/ou contenu dans les documents

semi-structurés. Ces résultats dépendent des collections des requêtes. En particulier leur nombre joue un rôle important, si on regarde CO, CO+S, VVCAS il y a respectivement 15, 12, 6 requêtes ayant des éléments pertinents dans le top 20 dans la collection 2005. En effet, il suffit qu'une requête améliore ou dégrade les résultats de manière importante, pour que l'amélioration globale change de manière drastique. Ceci reste difficile à contrôler.

Conclusion Générale

Synthèse

Notre travail se situe dans le cadre de la reformulation de requêtes en Recherche d'Information dans des documents semi-structurés de type XML. Nous nous sommes particulièrement intéressés à la réinjection de pertinence (communément appelée *Relevance Feedback*).

Plusieurs questions se posent dans ce contexte. Les principales sont : quels indicateurs pourront être utiles pour l'extraction des termes pertinents à partir des éléments de différentes granularités jugés pertinents ? Comment tenir compte du fait que les éléments peuvent être imbriqués les uns dans les autres ? Existe-t-il des structures pertinentes et comment sont-elles définies ? Est-il intéressant d'enrichir une requête avec des contraintes structurelles ? Comment peut-on exprimer les relations contextuelles pouvant exister entre un terme et la balise dans laquelle il apparaît ? Comment intégrer ces deux évidences dans la requête initiale ? Des questions plus techniques font aussi le sujet de cette thèse, elles concernent la réécriture des requêtes : Doit-on re-pondérer les termes originaux ? Comment rajouter des structures à des requêtes déjà structurées ? A quels groupes de mots-clés doit-on ajouter des conditions structurelles ?

Notre objectif est d'apporter des réponses à certaines de ces questions. Nous avons alors proposé un mécanisme de reformulation de requêtes par réinjection de pertinence utilisant les deux sources d'évidence : contenu et structure, à la fois de manière dépendante et indépendante. Plus précisément, nous avons proposé trois approches de reformulation : une première orientée contenu, une seconde orientée structure et une troisième combinant le contenu et la structure.

Au niveau de l'approche orientée contenu, la réinjection de pertinence se fait par ajout de termes pertinents extraits des éléments jugés pertinents.

Nous avons proposé une approche permettant de sélectionner les termes pertinents selon leur distribution dans les éléments pertinents et non pertinents ainsi que leur proximité vis-à-vis des termes de la requête initiale. Nous avons en outre proposé de prendre en compte la pertinence négative traduite par le facteur bruit.

Nous avons montré à travers les expérimentations l'intérêt de la prise en compte de plus d'un indicateur pour la sélection des termes pertinents. Nous avons montré en outre que le nombre de termes à ajouter dépend principalement de la requête, mais il semble que l'ajout de peu de termes est préconisé. De plus la pondération des termes doit être faite ; comme en RI classique, en tenant compte de la présence/absence des termes dans les éléments pertinents.

Concernant l'approche orientée structure, Nous avons montré de manière empirique l'existence de la notion de structure pertinente. En effet, nous avons proposé un algorithme qui permet d'extraire une structure pertinente à partir des éléments pertinents. Les résultats de l'évaluation de notre approche montrent de manière claire que l'ajout de structures pertinentes entre 1 et 3 a un effet positif en termes de performances. Nous obtenons des améliorations significatives dans la majorité des tâches considérés.

Les travaux réalisés dans le cadre de la réinjection de pertinence consistent à enrichir les requêtes initiales par le contexte des éléments pertinents qui traduit les caractéristiques d'un élément recherché sans spécifier exactement sa structure. La spécificité de notre approche vis-à-vis l'état de l'art, réside tout d'abord dans son aspect générique, elle est indépendante de toute DTD. Ensuite, la prise en compte explicite de la structure pertinente dans la requête reformulée.

Nous avons en outre proposé une approche qui combine les deux sources d'évidence de différentes façons : naïve, avec dépendance contextuelle et flexible. Nous avons conclu à travers les expérimentations que la cohabitation des deux sources d'évidences permet de mieux exprimer les besoins de l'utilisateur, et ainsi d'améliorer les performances du système. Cette combinaison est spécifique à nos travaux. En effet, les approches déjà développées en littératures ne considère pas les deux sources d'évidences. La combinaison nous a permis de prouver des relations de dépendances qui peuvent exister entre termes et structures, c'est le cas de la combinaison flexible, ainsi que la combinaison avec dépendance contextuelle.

Il est également à noter que nos approches sont applicables sans avoir de restrictions ni sur des collections de documents précises, ni sur des types de requêtes ni sur un type de DTD particulier.

Perspectives

Nous proposons dans ce qui suit quelques perspectives à ces travaux.

1. Une première perspective est de mieux comprendre la pondération de termes pertinents. Les résultats sont variables selon les collections, ceci ne corrobore pas souvent avec ce qui est fait en RI classique. Nous pensons que la taille des éléments jugés pertinents doit avoir un impact important dans toutes les phases de la réinjection. Les éléments strictement pertinents sont souvent de taille réduite, ceci peut avoir un impact évident sur les phases d'extraction et pondération des termes.
2. La diversité des sources documentaires rend la prise en compte de l'aspect hétérogène inévitable. Une première application qu'on n'a pas pu tester est celle qui concerne la réinjection de pertinence en considérant des corpus hétérogènes ayant différentes DTD. Nous avons déjà proposé une solution en ce qui concerne l'approche orientée structure, il reste à la tester sur une collection de documents hétérogènes.
En ce qui concerne l'approche orientée contenu le problème d'extraction des termes pertinents est le même que celui dans le cas des documents homogène. En revanche, la prise en compte de la dépendance contextuelle entre termes et éléments doit être repenser en s'appuyant par exemple sur des méthodes de classification.
3. Dans cette thèse, nous avons proposé la reformulation de requêtes par réinjection de pertinence en considérant des sources d'évidence extraites des éléments jugés par l'utilisateur. Nous remarquons qu'en général les jugements sont subjectifs et dépendent essentiellement du profil de l'utilisateur. Une de nos perspective est d'enrichir nos sources d'évidence en utilisant par exemple le profil. Ce dernier peut être construit à partir de l'historique des recherches de l'utilisateur ou par des informations fournies explicitement par l'utilisateur. Plus précisément, ces informations peuvent nous servir de deux manières différentes :
 - soit au niveau de la sélection des termes, de l'extraction des structures pertinentes ou aussi au niveau des relations entre la sémantique des éléments et leurs contenus selon la perception de l'utilisateur.
 - soit au niveau de jugement de pertinence. En effet on pourra modéliser l'utilisateur en se basant sur son profil, cette modélisation nous permettra de ne plus faire intervenir l'utilisateur d'une manière interactive, mais plutôt, il interviendra pour le jugement à travers son profil.
4. Une perspective envisagée concerne la prise en compte de l'information

multimédia dans les documents semi-structurés. Une question en lien avec nos travaux concerne la prise en compte de ce type d'information lors de la réinjection de pertinence. Ceci peut être pris en compte à 2 niveaux. Le premier concerne la réécriture de la requête. Dans ce cas, la réinjection de pertinence ne portera pas seulement sur la description des besoins de l'utilisateur par des mots clés et des contraintes structurelles mais aussi des contraintes décrivant les caractéristiques de bas niveau du document multimédia recherché. Le second, plus complexe, consiste à à extraire à partir des éléments de type image de l'information textuelle ou structurée à réinjecter dans la requête.

Annexe A

Les Documents XML

A.1 Structure du document XML

Un document XML est structuré en 3 parties :

1. un **prologue**, situé dans l'entête des documents XML, permet d'indiquer la version de la norme XML utilisée pour créer le document (cette indication est obligatoire) ainsi que le jeu de caractères (en anglais *encoding*) utilisé dans le document (attribut facultatif).

Dans l'exemple de la figure A.1, on spécifie qu'il s'agit du jeu ISO-8859-1, jeu LATIN, pour permettre de prendre en compte les accents français.

Le prologue se poursuit avec des informations facultatives sur des instructions de traitement à destination d'applications particulières. Leur syntaxe est la suivante :

`< ?instruction de traitement ? >`.

2. XML fournit un moyen de vérifier la syntaxe d'un document grâce aux **DTD** (*Document Type Definition*).

Il s'agit d'un fichier décrivant la structure des documents. Un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles. Un document suivant les règles de XML est appelé *document bien formé*. Un document XML possédant une DTD et étant conforme à celle-ci est appelé *document valide*. Une DTD peut être définie de 2 façons :

- sous forme interne, c'est-à-dire en incluant la grammaire au sein même du document,
- sous forme externe, soit en appelant un fichier contenant la grammaire à partir d'un fichier local ou bien en y accédant par son URL.

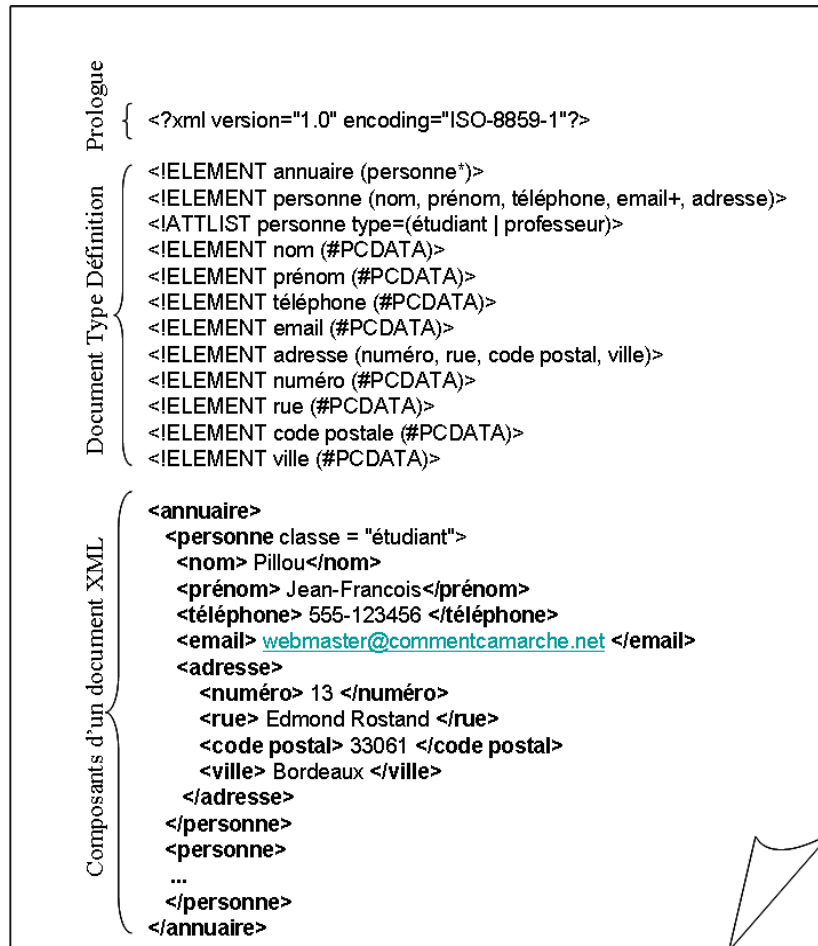


FIG. A.1 – Exemple d'un document XML

Un autre outil pour décrire la grammaire de document XML est le langage de schéma XML [59]. Celui-ci apporte une grande souplesse dans la définition des documents XML en permettant la prise en charge des types de données garantissant le contenu à affecter à un élément XML et apportant une validation plus efficace, non seulement sur la structure du document, mais aussi sur le type de son contenu.

Dans la figure A.1, la deuxième partie représente une DTD décrivant les composants d'un document XML.

La première ligne de la DTD sert à déclarer un élément de type *annuaire* composé d'éléments de type *personne*. D'après cette DTD, un élément de type *personne* doit avoir les éléments imbriqués de types : *nom*, *prénom*, ...et *adresse*. L'expression ATTLIST de la troisième ligne de la DTD permet de déclarer un attribut (exemple l'attribut *type* qui peut avoir soit une valeur ="étudiant" ou "professeur").

3. l'ensemble des éléments composant le document XML comme représenté dans la troisième partie de la figure A.1.

Un élément est limité par une balise ouvrante dans laquelle on retrouve éventuellement la valeur de l'attribut (exemple l'attribut *type* de l'élément *personne*) et une balise fermante. Il peut contenir directement l'information textuelle (exemple l'élément *nom*) et/ou d'autres éléments qui sont imbriqués (exemple l'élément *adresse*).

L'ensemble des éléments peut être représenté sous forme d'arbre décrivant d'une manière plus simple les relations entre les différents éléments représentés par des nœuds.

XML, outre le fait d'être particulièrement adapté à l'échange de données et de documents, présente principalement les avantages suivants :

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre le contenu d'un document XML
- Son caractère auto-descriptif et extensible
- Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques
- Universalité et portabilité : il peut être facilement distribué par n'importe quel protocole et même de transporter du texte, comme HTTP. En outre, les différents jeux de caractères sont pris en compte.
- Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML)
- Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

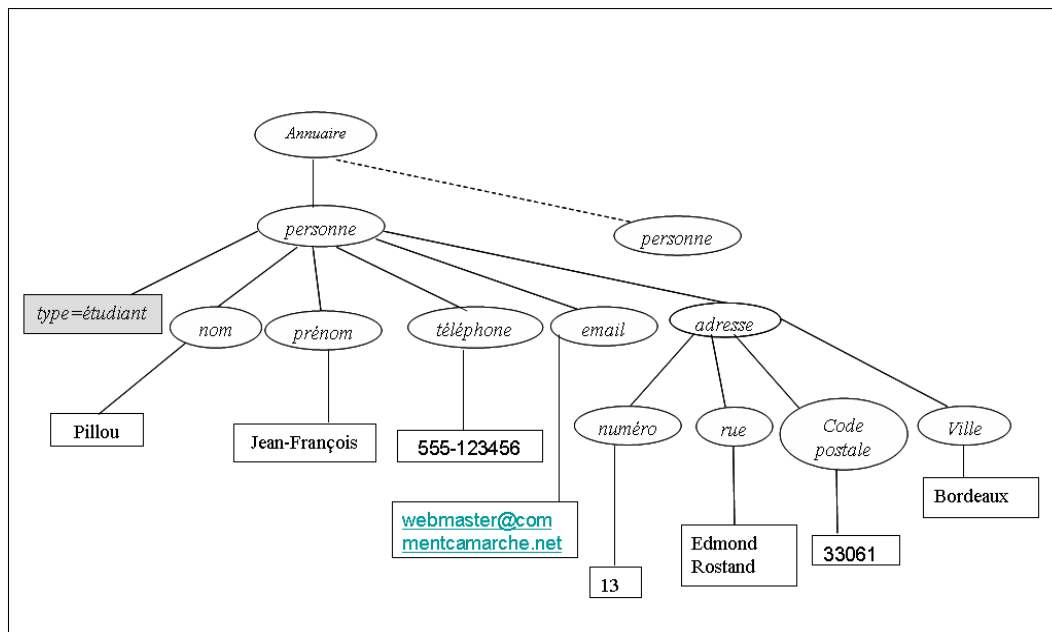


FIG. A.2 – L'arbre DOM d'un document XML

A.2 Les DOMs

Le *Modèle d'Objet de Document* (DOM) [201], développé par le W3C, présente une interface de programmation d'applications (API : *Applications Programming Interface*) qui définit la structure logique, les modes de gestion et d'accès des documents XML, HTML et CSS (*Cascading StyleSheet*). Il permet donc un accès dynamique aux documents et la mise à jour de leur contenu, de leur structure et de leur style par l'intermédiaire de programmes ou de scripts. On trouvera sur la figure A.2 un exemple d'arbre DOM associé au document XML de la figure A.1 :

Dans l'arbre, les **nœuds feuilles** sont les nœuds comportant l'information textuelle et qui ne possèdent pas de descendants (par exemple "Pillou" est un nœud feuille). La **racine** représente le document entier et ne possède pas d'ancêtres (sur la figure, l'élément *annuaire* est la racine). Les nœuds **intermédiaires** (ou internes) sont des nœuds qui ont à la fois des ancêtres et des descendants (sur la figure, les éléments *personne*, *adresse* et *email* sont des nœuds intermédiaires).

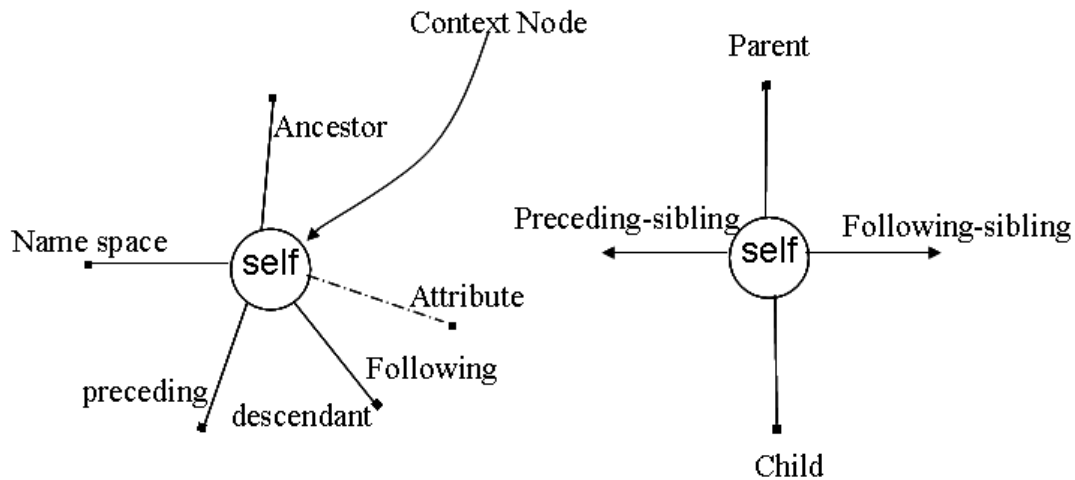


FIG. A.3 – Axes de navigation XPath

A.3 XPath

XPath [41] est un langage d'expression s'appliquant à XML ; il s'agit d'un langage permettant de sélectionner des sous-arbres d'un document XML. Il possède une syntaxe simple et non ambiguë et implémente des types usuels (chaînes, nombres, booléens, variables, fonctions). Il permet aussi de manipuler des nœuds et des ensembles de nœuds. XPath est utilisé par Xpointer [78] et XSLT [53].

Une expression XPath est un chemin de localisation constitué par une suite d'éléments ou d'attributs séparés par une barre de fraction (« / »). XPath fournit des fonctions intégrées, permet d'utiliser des variables, de définir des filtres et de spécifier des axes comme décrit dans la figure A.3.

Ces axes sont :

- **child** : : enfants du noeud contextuel
- **descendant** : : descendant du noeud contextuel
- **parent** : : parent du noeud contextuel
- **ancestor** : : ancêtre du noeud contextuel
- **following-sibling** : : tous les nœuds suivant le noeud contextuel et ayant le même noeud parent
- **preceding-sibling** : : tous les nœuds précédant le noeud contextuel et ayant le même noeud parent
- **following** : : tous les nœuds dans le même document que le noeud contextuel et étant après lui dans l'ordre du document (lecture séquentielle)
- **preceding** : : tous les nœuds dans le même document que le noeud contextuel et étant avant lui dans l'ordre du document
- **attribute** : : attributs du noeud contextuel
- **namespace** : : nœuds espaces de nom du noeud contextuel

- **self** : : le noeud contextuel lui-même
- **descendant-or-self** : : le noeud contextuel ou ses descendants
- **ancestor-or-self** : : le noeud contextuel ou ses ancêtres

On trouvera ci-dessous des exemples d'expression XPATH appliquées au document de la figure [A.1](#) :

- `/` : sélectionne l'élément qui englobe tout le document. Dans l'exemple il s'agit de tout *l'annuaire*.
- `//personne` : sélectionne tous les éléments de type *personne*.
- `/personne/email` : sélectionne tous les éléments de type *email* appartenant à l'élément *personne*.
- `//personne[@type="étudiant"]` : sélectionne tous les éléments de type *personne* dont l'attribut *type*="étudiant"

Bibliographie

- [1]
- [2] *e-XML*. Disponible sur <http://www.e-xmlmedia.fr>.
- [3] *XQL (Langage d'interrogation de XML)*. Disponible sur <http://www.ibilio.org/xql>, 1999.
- [4] *INitiative for the Evaluation of XML Retrieval*. disponible sur <http://inex.is.informatik.uni-duisburg.de:2004/tracks/rel/>, 2004.
- [5] *INitiative for the Evaluation of XML Retrieval*. disponible sur <http://inex.is.informatik.uni-duisburg.de:2005/tracks/rel/>, 2005.
- [6] S. Abiteboul, D. Quass, J. McHugh J. Widom, and J. Wiener. *The Lorel Query Language for Semistructured Data*. Disponible sur <http://cite-seer.ist.psu.edu/abiteboul97lorel.html>, 1997.
- [7] M. Abolhassani and N. Fuhr. Applying the divergence from randomness approach for content-only search in XML documents. In *Proceedings of ECIR 2004, Sunderland*, pages 409–419, 2004.
- [8] J. Allan, J. Callan, M. Sanderson, J. Xu, and S. Wegmann. INQUERY at TREC-7. In *Proceedings of TREC-7*, pages 201–216, 1998.
- [9] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. Texquery : A full-text search extension to Xquery. In *Proceedings of WWW 2004*, 2004.
- [10] V.N. Anh and A. Moffat. Compression and an IR approach to XML retrieval. In *Proceedings of INEX 2002 Workshop, Dagstuhl, Germany*, 2002.
- [11] R. Attar and A.S. Fraenkel. Local feedback in full-text retrieval systems. *Journal of the ACM*, 24(3) :pages 397–417, 1977.
- [12] Ricardo Baeza-Yates and Berthier Riberto-Neto. *Modern Information Retrieval*. New-York : ACP Press, Addison-Wesley, 1999.
- [13] N. J. Belkin, A. Cabezas, C. Cool, K. Kim, K. B. Ng, S. Park, R. Pressman, S. Rieh, P. Savage, and H. Xie. Rutgers interactive track at trec-5. In *Proceedings of the Sixth Text Retrieval Conference (TREC-5)*., pages 257–266, 1997.
- [14] N. J. Belkin, J. Perez Carballo, C. Cool, S. Lin, S. Y. Park, S. Y. Rieh, P. Savage, C. Sikora, H. Xie, and J. Allan. Rutgers' trec-6 interactive

- track experience. In *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*., pages 597–610, 1998.
- [15] N. J. Belkin, C. Cool, J. Koenemann, K. Bor Ng, and S. Park. Using relevance feedback and ranking in interactive searching. In *Proceedings of the Fourth Text Retrieval Conference (TREC-4)*., pages 181–210, 1996.
- [16] N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. In *Information Processing and Management*., pages 431–448, 1995.
- [17] N.J. Belkin, J. Perez Carballo, D. Kelly, S. Lin, S.Y. Park, S.Y. Rieh, P. Savage-Knepshield, C. Sikora, and C. Cool. Rutgers' trec-7 interactive track experience. In *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*., pages 275–284, 1999.
- [18] N.J. Belkin and W.B. Croft. Information retrieval and information filtering : two sides of the same coin? *Communications of the ACM*, 35(12), December 1992.
- [19] S.K. Bhatia. Selection of search terms based on user profile. In *ACM/SIGAPP Symposium on Applied computing (vol I) : technological challenges of the 1990's. Proceedings of the 1992*, pages 224–233, 1992.
- [20] P. Bohannon, J. Freire, P. Roy, and J. Simeon. From XML schema to relations : A cost-based approach to XML storage. In *Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA, USA*. Morgan Kaufmann, 2002.
- [21] M. Boughanem, C. Chrisment, J. Mothe, C. Soulé-Dupuy, and L. Tamine. Connexionist and genetic approaches to achieve ir. *Soft Computing in Information Retrieval Techniques and application Editorial*., pages 173,198, 2000.
- [22] M. Boughanem, C. Chrisment, and C. Soulé-Dupuy. Query Modification based on relevance back-propagation in adhoc environnement. *Information Processing & Management*, 35 :121–139, avril 1999.
- [23] M. Boughanem, C. Chrisment, and C. Soule-Dupuy. Query modification based on relevance backpropagation in adhoc environment. *Information Processing and Management*, 35 :pages 121–139, 1999.
- [24] M. Boughanem and C. Soulé-Dupuy. Query modification based on relevance back propagation. In *International Conference on Computer-Assisted Information Retrieval, RIAO, Montréal*, pages 469–487. CID, juin 1997.
- [25] A. Brini. *Un modèle de recherche d'information basé sur les réseaux possibilistes* . Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2006.
- [26] A. Brini, M. Boughanem, and D. Dubois. A Model for Information Retrieval Based on Possibilistic Networks . In *String Processing and Information Retrieval (SPIRE 2005)* , Buenos Aires, ARGENTINE, , pages 271–282. LNCS, Springer Verlag, November 2005.

- [27] C. Buckley. Why current ir engines fail. In *SIGIR '04 : Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 584–585, New York, NY, USA, 2004. ACM.
- [28] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART : TREC 3. In *Text REtrieval Conference*, pages 0–, 1994.
- [29] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart : Trec-3. In *Proceedings of the Third Text Retrieval Conference (TREC-3).*, pages 69–80, 1995.
- [30] J. Callan. Passage-level evidence in document retrieval. In *Proceedings of SIGR 1994, Dublin, Ireland*, pages 302–309, 1994.
- [31] I. Campbell. Supporting information needs by ostensive definition in an adaptive information space. In *MIRO '95. electronic Workshops in Computing, Springer Verlag.*, 1995.
- [32] I. Campbell. Interactive evaluation of the ostensive model, using a new test-collection of images with multiple relevance assessments. *Journal of Information Retrieval.*, 2(1) :89–114, 1999.
- [33] I. Campbell and C. J. Van Rijsbergen. Ostensive model of information needs. In *Proceedings of the Second International Conference on Conceptions of Library and Information Science : Integration in Perspective (CoLIS 2).*, pages 251–268, 1996.
- [34] L. De Campos, Juan F. Huete, and Juan M. Fernandez-Luna. Document instantiation for relevance feedback in the bayesian network retrieval model.
- [35] D. Carmel, N. Efraty, G. Landau, Y. Maarek, and Y. Mass. An extension of the vector space model for quering xml fragments. In *ACM SIGIR'2002 Workshop on XML and IR*. Finland, August 2002.
- [36] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL : A graphical language for querying and restructuring WWW data. In *Proceedings Of the 8th Int. WWW Conference, WWW8, Toronto, Canada*, May 1999.
- [37] D. Chamberlin, J. Robie, and D. Florescu. Quilt : An XML query language for heterogeneous data sources. In *Proceedings of the 3rd International Workshop on World Wide Web and databases, Dallas, USA*, pages 1–25, 2000.
- [38] Y.K. Chang, C. Cirillo, and J. Razon. Evaluation of feedback retrieval using modified freezing, residual collection and test and control groups. *The SMART retrieval system- experiments in automatic document processing*, pages 355,370, 1971.
- [39] J.-P. Chevallet and J.Y. Nie. Intégration des analyses du français dans la recherche d'informations. In *Recherche d'Informations Assistée par Ordinateur (RIAO'97), Montreal*, pages 761–772, jun 1997.

- [40] Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical report, Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, 1996.
- [41] J. Clark and S. Derose. XML Path Language (XPath) , version 1.0. Technical report, World Wide Web Consortium (W3C), W3C Recommendation, Novembre 1999.
- [42] V. Claveau and P. Sébillot. Extension de requêtes par lien sémantique nom-verbe acquis sur corpus. april 2004.
- [43] D. Colazzo, C. Sartiani, A. Albano, P. Manghi, G. Ghelli, L. Lini, and M. Paoli. A typed text retrieval query language for XML documents. *JASIST*, 53(6) :pages 647–488, 2002.
- [44] B. Croft. Experiments with representations in a document retrieval system. *Information Technology : Research and Development.*, 35(4) :1,21, 1983.
- [45] B. Croft and D. Harper. Using probabilistic models of information without relevance information. *Journal of Documentation.*, 35(4) :285,295, 1979.
- [46] W.B. Croft, R. Cook, and D. Wilder. Providing government information on the internet : Experiences with THOMAS. U. of Mass. Technical report 95-45, 1995.
- [47] C. Crouch, A. Mahajan, and A. Bellamkonda. Flexible XML retrieval based on the vector space model. In *INEX 2004 Workshop Proceedings*, pages 292,302. Germany, December 2004.
- [48] C. J. Crouch, S. Apte, and H. Bapat. An approach to structured retrieval based on the extended vector model. In *Proceedings of INEX 2003 Workshop*, pages 89,93. Germany, December 2003.
- [49] C. J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In *Proceedings of the ACM-SIGIR Conference on Research and Development in Information Retrieval , Copenhagen, Denmark*, pages 77–88, 1992.
- [50] L. Denoyer and P. Gallinari. Bayesian network model for semi-structured document classification. *Information Processing and Management*, 40 :807,827, 2004.
- [51] L. Denoyer and P. Gallinari. The wikipedia xml corpus. *SIGIR Forum*, 40(1) :64–69, 2006.
- [52] L. Denoyer, G. Wisniewski, and P. Gallinari. Document structure matching for heterogenous corpora. In *Proceedings of the 27th Annual International ACM SIGIR Conference*. Sheffield, United Kingdom, July 2004.
- [53] S. Derose, E. Maler, and D. Orchard. XML Linking Language (XLink), version 1.0. Technical report, World Wide Web Consortium (W3C),W3C Recommendation, juin 2001.

- [54] A. Deutsch, M. F. Fernandez, and D. Suciu. Storing semistructured data with STORED. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *Proceedings ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*, pages 431–442, June 1999.
- [55] E. Efthimiadis and P. Biron. Ucla-okapi at trec-2 : query expansion experiments. In *Proceedings of the Second Text Retrieval Conference (TREC-2)*., pages 279–290, 1994.
- [56] E.N. Efthimiadis. Interactive query expansion : a user based evaluation in relevance feedback environment. *Journal of the American Society for Information Science*, 51(11) :989,1003, 2000.
- [57] D. Ellis. A behavioural approach to information system design. *Journal of Documentation.*, 45(3) :171–212, 1989.
- [58] E-XMLMedia XMLizer. <http://www.e-xmlmedia.fr/site-francais/produits-xmlizer.htm>.
- [59] D.C. Fallside. XML Schema. Technical report, World Wide Web Consortium (W3C), W3C Recommendation, 2001.
- [60] M. Fernandez. *XQuery 1.0 and XPath 2.0 Data Model W3C Working Draft*. Disponible sur <http://www.w3.org/TR/xpath-datamodel/>, October 2004.
- [61] D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. *IEEE Data Engineering Bulletin*, 22(3) :pages 27–34, 1999.
- [62] C. Fox. *Lexical analysis and stoplists*, pages 102–130. Frakes W B, Baeza-Yates R (eds) Prentice Hall, New jersey, 1992.
- [63] W. B. Frakes. *Stemming Algorithms*, pages 131–160. Frakes W B, Baeza-Yates R (eds) Prentice Hall, New jersey, 1992.
- [64] N. Fuhr, N. Govert, G. Kazai, and M. Lalmas. Proceedings of the first workshop of the initiative for the evaluation of XML retrieval (INEX 2002), 2002.
- [65] N. Fuhr and K. Grossjohann. XIRQL : a query language for information retrieval in XML documents. In *In Proceedings of SIGIR 2001, Toronto, Canada*, 2003.
- [66] N. Fuhr, M. Lalmas, and S. Malik. INEX 2003 workshop proceedings, 2003.
- [67] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlavik. INEX 2004 workshop proceedings. Springer, 2004.
- [68] N. Fuhr, M. Lalmas, and A. Trotman. INEX 2006 workshop proceedings, 2006.
- [69] N. Fuhr, Mounia Lalmas, S. Malik, and G. Kazai. INEX 2005 workshop proceedings, 2005.

- [70] N. Gövert, M. Abolhassani, N. Fuhr, and K. Grossjohann. Content-oriented XML retrieval with hyrex. In *Proceedings INEX 2002, Dagstuhl, Germany*, 2002.
- [71] N. Gövert, M. Abolhassani, N. Fuhr, and K. Grossjohann. Content-oriented XML retrieval with hyrex. In *Proceedings INEX 2002, Dagstuhl, Germany*, 2002.
- [72] S. Geva. Gpx-gardens point xml information retrieval at inex 2004. In *INEX 2004 Workshop Proceedings*, pages 211,223. Dagstuhl, Germany, December 2004.
- [73] S. Geva, M. Hassler, and X. Tannier. XOR - XML Oriented Retrieval Language. In *Proceedings of ACM SIGIR 2006 Workshop on XML Element Retrieval Methodology, Seattle, WA, USA*. ACM Press, New York City, NY, USA, August 2006.
- [74] C. Goldfarb. *The SGML Handbook*. Oxford University Press, Oxford, 1990.
- [75] T. Grabs and H.-J. Scheck. Flexible information retrieval from xml with PowerDB XML. In *Proceedings of INEX 2002, Dagstuhl, Germany*, pages 26–32, December 2002.
- [76] T. Grabs and H. Schek. Eth zurich at inex, flexible information retrieval from xml with powerdb-xml. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML REtrieval(INEX)*, pages 141,148. Dagstuhl, Germany, December 2002.
- [77] Torsten Grabs. *Storage and Retrieval of XML Documents within a Cluster of Database Systems*. PhD thesis, Ecole Polytechnique Fédérale de Zürich, 2003.
- [78] P. Grosso, E. Maler, J. Marsh, and N. Walsh. XML Pointer Language (XPointer). Technical report, World Wide Web Consortium (W3C), W3C Recommendation, march 2003.
- [79] T. Grust. Accelerating xpath location steps. In *SIGMOD '02 : Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120, New York, NY, USA, 2002. ACM Press.
- [80] A. Gutierrez, R. Motz, and D. Viera. Building databases with information extracted from web documents. In *Proceedings XX international conference of the Chilean computer sciences society*, pages 41–49, 2000.
- [81] D. Haines and W.B. Croft. Relevance feedback and inference network. In *16th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 2,11, 1993.
- [82] D. Harman. Towards interactive query expansion. In *11th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 321,331, 1988.
- [83] D. Harman. Relevance feedback revisited. In *15th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 1,10, 1992.

- [84] K. Hatano, H. Kinutani, and M. Watanabe. An appropriate unit of retrieval results for xml document retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML REtrieval (INEX)*. Dagsthul, Germany, Decemder 2002.
- [85] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 569–584, 1998.
- [86] L. Hlaoua, , M. Torjmen, K. Pinel-Sauvagnat, and M. Boughanem. XFIRM at INEX 2006. Ad-hoc, Relevance Feedback and MultiMedia tracks. In *International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Allemagne, 18/12/2006-20/12/2006*, 2006.
- [87] L. Hlaoua. *Recherche d'Information dans des Documents XML : Utilisation d'une Technique de Propagation de la Pertinence*. rapport dea, Université Paul Sabatier de Toulouse, 2004.
- [88] L. Hlaoua. Reformulation de Requêtes par Structure en RI dans les Documents XML. In *Conférence francophone en Recherche d'Information et Applications, Lyon, 15/03/06-17/03/06*, pages 395–400, <http://www.irit.fr/ARIA>, mars 2006. Association Francophone de Recherche d'Information et Applications (ARIA).
- [89] L. Hlaoua and K. Pinel-Sauvagnat and M. Boughanem. Relevance Feedback for XML Retrieval : using structure and content to expand queries. In Colette Rolland, Oscar Pastor, and Jean-Louis Cavarero, editors, *International Conference on Research Challenges in Information Science (RCIS), Ouarzazate- Maroc, 23/04/2007-26/04/2007*, pages 195–202, <http://www.emsi.ma/>, avril 2007. EMSI - Ecole MArocaïne des Sciences de l'Ingénieur.
- [90] L. Hlaoua and M. Boughanem. Towards Contextual and Structural Relevance Feedback in XML Retrieval. In Michel Beigbeder and Wai Gen Yee, editors, *workshop on Open Source Web Information Retrieval, compïègne, 19/09/05*, pages 35–38. ISBN :2-913923-19-4, septembre 2005.
- [91] L. Hlaoua, M. Boughanem, and K. Pinel-Sauvagnat. Combination of Evidences in Relevance Feedback for XML Retrieval. In *Conference on Information and Knowledge Management (CIKM), Lisbonne, Portugal, novembre 2007*.
- [92] L. Hlaoua, M. Boughanem, and K. Pinel-Sauvagnat. Using a Content-and-Structure Oriented Method for Relevance Feedback in XML Retrieval. In *Large-Scale Semantic Access to Content (Text, Image, Video and Sound) (RIAO), Pittsburgh (PA) États-Unis, 30/05/2007-01/06/2007*, page (electronic medium), <http://www.le-cid.org>, juin 2007.
- [93] L. Hlaoua, M. Boughanem, and K. Sauvagnat. Combinaison des caractéristiques des termes pour l'extension des requêtes en recherche d'in-

- formation dans les documents xml. In *CORIA 2007*, pages 77,92. Saint Etienne, Mars 2007.
- [94] L. Hlaoua and K. Sauvagnat. Structure-oriented relevance feedback in xml retrieval. In *InSciT2006.*, Merida, Espagne, October 2006.
- [95] L. Hlaoua, K. Sauvagnat, and M. Boughanem. Réinjection de structures pour la reformulation de requêtes en ri structurée. In *INFORSID 2006*, pages 435,450. Hammet, Tunisie, Juin 2006.
- [96] L. Hlaoua, K. Sauvagnat, and M. Boughanem. A structure-oriented relevance feedback method for xml retrieval. In *Proceedings of the 15th ACM Annual Conference on Information and Knowledge Management CIKM'06*. Arlington, Virginia, USA., November 2006.
- [97] G. Hubert. A voting method for XML retrieval . In Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors, *Advances in XML Information Retrieval : Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, LNCS 3493 / 2005*, Dagstuhl, Germany, , pages 183–196. Springer-Verlag GmbH, mai 2005. Dates de conférence : mai 2005 2005.
- [98] G. Huck, I. Macherius, and P. Fankhauser. PDOM : Lightweight persistency support for the document object model. In *Succeeding with Object Databases*, John Wiley, 2000.
- [99] E Ide. New experiments in relevance feedback. In *The SMART retrieval system - experiments in automatic document processing.*, pages 337–354, 1971.
- [100] P. Ingwersen. Polyrepresentation of information needs and semantic entities : elements of a cognitive theory for information retrieval interaction. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*, pages 101–110, 1994.
- [101] P. Ingwersen. Cognitive perspectives of information retrieval interaction : elements of a cognitive ir theory. *Journal of Documentation.*, 52(1) :3–50, 1996.
- [102] H.C. Jang, Y.I. Kim, and D.W. Shin. An effective mechanism for index update in structured documents. In *Proceedings ACML CIKM, Kansas City*, pages 383–390, 1999.
- [103] K.S. Jones. Further reflections on trec. *Inf. Process. Manage.*, 36(1) :37–85, 2000.
- [104] Y.Choy K. Lee and S. Cho. An efficient algorithm to compute differences between structured documents. *IEEE Transaction On Knowledge and Data Engineering*, 16(8), August 2004.
- [105] V. Kakade and P. Raghavan. Encoding XML in vector spaces. In *Proceedings of ECIR 2005, Saint Jacques de Compostelle, Spain*, 2005.

- [106] J. Kamps, M. de Rijke, and B. Sigurbjornsson. Length normalization in XML retrieval. In *Proceedings of SIGIR 2004, Sheffield, England*, pages 80–87, 2004.
- [107] C.-C. Kanne and G. Moerkotte. Efficient storage of XML data. In *In Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA*, page 198, 2000.
- [108] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *Proceedings of SIGIR 1997, Philadelphia, USA*, pages 178–185, 1997.
- [109] G. Kazai and M. Lalmas. Inex 2005 evaluation metrics. In *INEX 2005 Workshop Pre-Proceedings*, pages 401,406. Germany, November 2005.
- [110] G. Kazai, M. Lalmas, and T. Roelleke. A model for the representation and focused retrieval of structured documents based on fuzzy aggregation. In *SPIRE'2001*, pages 123,135. Lagune de San Rafaël, Chile, 2001.
- [111] G. Kazai, M. Lalmas, and T. Roelleke. Focused document retrieval,. In *9th International Symposium on string processing and information retrieval, Lisbon, Portugal*, September 2002.
- [112] C.C. Kuhlthau. Principle for uncertainty for information seeking. *Journal of Documentation.*, 49(4) :339–355, 1993.
- [113] J. Fernandez-Luna L. Compos and J. Huete. Using context information in structured document retrieval : An approach based on influence diagrams. *Information Processing and Management*, 40 :829,847, 2004.
- [114] M. Lalmas. Dempster shafer s theory of evidence applied to structured documents : modelling uncertainty. In *Proceedings of annual international ACM SIGIR'97 Conference*, pages 110–118. Philadelphia PA, USA, 1997.
- [115] R.R. Larson. Cheshire ii at inex : Using a hybrid logistic regression and boolean model for xml retrieval. In *Proceedings of the First Workshop of the INiative for the Evaluation of XML REtrieval(INEX)*, pages 18,25. Dagsthul, Germany, December 2002.
- [116] R.R. Larson. Cheshire ii at inex'04 : Fusion and feedback for the adhoc and heterogenous tracks. In *INEX 2004 Workshop Proceedings*, pages 322,336. Dagsthul, Germany, December 2004.
- [117] O. Lassila and R.R. Swick. Resource Description Framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium (W3C),W3C Recommendation, Februar 1999.
- [118] J. H. Lee. Combining the evidence of different relevance feedback methods for information retrieval. *Information Processing and Management.*, 34(6) :681–691, 1998.
- [119] Y.K. Lee, S.J. Yoo, and K. Yoon. Index structures for structured documents. In *In Proc. ACM Workshop on XML and IR, Bethesda*, pages 91–99, 1996.
- [120] M. Lehtonen. Extirp 2004 : Towards heterogeneity. In *INEX 2004 Workshop Proceedings*, pages 372,381. Dagsthul, Germany, December 2004.

- [121] A. Levy, M. Fernandez, D. Suciu, D. Florescu, and A. Deutsch. XML-QL : A query language for XML. Technical report, World Wide Web Consortium technical report, Number NOTE- xml-ql-19980819, 1998.
- [122] Q. Li and B. Moon. Indexing and querying XML data for regular path expressions. In *Proceedings of the 27th VLDB Conference, Roma, Italy*, 2001.
- [123] Y. Li, C. Yu, and H.V. Jagadish. Schema-free xquery. In *VLDB*, 2004.
- [124] W. Lian and D. Cheung. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Transaction And Data Engineering*, 16(1), 2004.
- [125] J. A. List, V. Mihajlovic, A. P. de Vries, and G. Ramirez. The TIJAH XML-IR system at INEX 2003. In *Proceedings of INEX 2003 Workshop*, pages 102,109. Dagsthul, Germany, December 2003.
- [126] R.W.P. Luk, H.V. Leong, T.S. Dillon, A.T.S. Shan, W.B Croft, and J. Allan. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology*, 53(3) :pages 415–435, 2002.
- [127] C. Lundquist, D. Grossman, and O. Frieder. Improving relevance feedback in the vector space model. In *Proceedings of the 6th ACM Annual Conference on Information and Knowledge Management (CIKM'97)*, 1997.
- [128] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery*, 7 :pages 216–244, 1960.
- [129] M. Marx, J. Kamps, M. Rijke, and B. Sigurbjornsson. The importance of morphological normalization for xml retrieval. In *Proceedings of the First Workshop of the INiative for the Evaluation of XML REtrieval(INEX)*, pages 41,48. Dagsthul, Germany, December 2002.
- [130] Y. Mass and M. Mandelbrod. Retrieving the most relevant xml components. In *Proceedings of INEX 2003 Workshop*, pages 53,58. Dagsthul, Germany, December 2003.
- [131] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for XML retrieval. In *INEX 2004 Workshop Proceedings*, pages 73,84. Dagsthul, Germany, December 2004.
- [132] Y. Mass and M. Mandelbrod. Relevance feedback for XML retrieval. In *INEX 2004 Workshop Proceedings*, pages 303,310. Germany, December 2004.
- [133] Y. Mass, M. Mandelbrod, E. Amitay, Y. Maarek, and A. Soffer. JuruXML-an XML retrieval system at INEX'02. In *Proceedings of the First Workshop of the INiative for the Evaluation of XML REtrieval(INEX)*, pages 73,80. Dagsthul, Germany, Decemder 2002.

- [134] M.I. M.Azevedo, L.P. Amorim, and N. Ziviani. A universal model for xml information retrieval. In *INEX 2004 Workshop Proceedings*, pages 311,321. Dagsthul, Germany, December 2004.
- [135] V. Mihajlovic, G. Ramirez, A.P de Vries, D. Hiemstra, and H.E. Blok. TIJAH at INEX 2004 modeling phrases and relevance feedback. In *INEX 2004 Workshop Proceedings*, pages 276,291. Germany, December 2004.
- [136] V. Mihajlovic, G. Ramirez, T. Westerveld, H.E. Block, A.P de Vries, and D. Hiemstra. TIJAH scratches INEX 2005 vague element selection, overlap, image search, relevance feedback, and users. In *INEX 2005 Workshop Pre-Proceedings*, pages 54,71. Dagsthul, Germany, November 2005.
- [137] G.A. Miller. Wordnet : A lexical database for english. In *HLT*, 1994.
- [138] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*, pages 206–214. Melbourne, 1998.
- [139] P. Ogilvie and J. Callan. Combining document representations of known-item search. In *Proceedings of annual international ACM SIGIR Conference*. Toronto, Canada, 2003.
- [140] P. Ogilvie and J. Callan. Combining document representations of known-item search. In *Proceedings of the 26st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 143–150. Tronoto, Canada, July,28-August,1 2003.
- [141] J. Parikh and S. Kapur. Unity : relevance feedback using user query logs. In *SIGIR '06 : Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 689–690, New York, NY, USA, 2006. ACM Press.
- [142] H. J. Peat and P. Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science.*, 42(5) :pages 378–383, 1991.
- [143] K. Pinel-Sauvagnat, L. Hlaoua, and M. Boughanem. XML retrieval : what about using contextual relevance ? In *Annual ACM Symposium on Applied Computing (SAC), Dijon, 23/04/2006-27/04/2006*, pages 1114–1120, [http ://www.acm.org/](http://www.acm.org/), avril 2006. ACM Press.
- [144] B. Piwowarski. Working group report : the assessment tool. In *Proceedings of INEX 2003, Dagstuhl, Germany*, pages 181–183, December 2003.
- [145] B. Piwowarski. Eprum metrics and inex 2005. In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors, *INEX*, volume 3977 of *Lecture Notes in Computer Science*, pages 30–42. Springer, 2005.
- [146] B. Piwowarski, G. Faure, and P. Gallinari. Bayesian networks and inex. In *Proceedings of the First Workshop of the INiative for the Evaluation of XML REtrieval(INEX)*, pages 149,154. Dagsthul, Germany, December 2002.

- [147] J.M. Ponte and W.B. Croft. A language modelling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, 1998.
- [148] M. F. Porter. An algorithm for suffix stripping. Program 14, 1980.
- [149] Y. Qiu and H.P. Frei. Concept based query expansion. In *Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PAA, USA*, pages 160–169, 1993.
- [150] B. A. Ribeiro-Neto and R. Muntz. A belief network model for IR. In *Proceedings Of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Suisse*, pages 253–260, 1996.
- [151] C. J. Van Rijsbergen. *Information retrieval*. Butterworths. 2nd edition., 1979.
- [152] C.J. Van Rijsbergen, D. Harper, and M. Porter. The selection of good search terms. *Information Processing and Management.*, 17(2) :pages 77–91, 1981.
- [153] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC 3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.
- [154] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Proceedings of the Third Text Retrieval Conference (TREC-3).*, pages 109–126, 1995.
- [155] S.E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4) :pages 294–304, 1977.
- [156] S.E. Robertson and J.K.Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3) :129, 146, 1976.
- [157] S.E. Robertson, S.E. Walker, and M.M. Beaulieu. Large test collection experiments on an operational interactive system : Okapi at trec. *Information Processing & Management Journal*, 31 :260,345, 1995.
- [158] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART retrieval system-experiments in automatic document processing*, pages 313,323. Prentice Hall Inc, 1971.
- [159] T. Roelleke, M. Lalmas, G. Kazai, J. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. In *Proceedings of ECIR 2002*, 2002.
- [160] I. Ruthven and M. Lalmas. Selective relevance feedback using term characteristics. *CoLIS 3, Proceedings of the Third International Conference on Conceptions of Library and Information Science.*, 1999.
- [161] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2) :95–145, 2003.

- [162] I. Ruthven, M. Lalmas, and C.J. Van Rijsbergen. Ranking expansion terms using partial and ostensive evidence. In *Proceedings of the 4th International Conference on Conceptions of Library and Information Science. CoLIS 4.*, pages 109–126. Seattle, 2002.
- [163] I. Ruthven, M. Lalmas, and K. Rijsbergen. Combining and selecting characteristics of information use. *JASIST*, 53(5) :378,396, 2002.
- [164] G. Salton. A comparison between manual and automatic indexing methods. *Journal of the American Documentation*, 20(1) :61,71, 1971.
- [165] G. Salton. *Automatic text processing : The transformation, analysis and retrieval of information by computer*. Addison-Wesley publishing, MA, 1989.
- [166] G. Salton and C Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society of Information Science*, 41(4) :288,297, 1990.
- [167] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 31(2) :1002–1036, November 1983.
- [168] G. Salton and M. McGill. Introduction to modern information retrieval. *McGraw-Hill Book Company*, page 1, 1983.
- [169] K. Sauvagnat. Xfirm, un modèle flexible de recherche d’information pour le stockage et l’indexation de documents xml. In *Actes de CORIA ’04*, pages 121,142. Toulouse, France, Mars 2004.
- [170] K. Sauvagnat. *Modèle flexible pour la recherche d’information dans des corpus de documents semi-structurés*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, juin 2005.
- [171] K. Sauvagnat and M. Boughanem. *Recherche d’Information dans les documents XML*. rapport interne, Université Paul Sabatier de Toulouse, 2004.
- [172] K. Sauvagnat, M. Boughanem, and C. Chrisment. Searching XML documents using relevance propagation. In Alberto Apostolico and Massimo Melucci, editors, *Symposium on String Processing and Information Retrieval (SPIRE), Padoue, Italie*, LNCS, pages 242–254, <http://www.springerlink.com>, octobre 2004. Springer.
- [173] K. Sauvagnat, L. Hlaoua, and M. Boughanem. Xfirm at inex 2005 : ad-hoc and relevance feedback track. In *INEX 2005 Workshop Pre-Proceedings*, pages 72,83. Germany, November 2005.
- [174] K. Sauvagnat, G. Hubert, M. Boughanem, and J. Mothe. IRIT at INEX 2003. In *International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany*, pages 142–148, décembre 2003.
- [175] R. Schenkel and M. Theobald. Relevance feedback for structural query expansion. In *INEX 2005 Workshop Pre-Proceedings*, pages 260,272. Germany, November 2005.

- [176] T. Schileder and H. Meuss. Querying and ranking XML documents. *Journal of the American Society for Information Science and Technology*, 53(6) :pages 489–503, 2002.
- [177] A. Schmidt, M. Kersten, and M. Windhouwer. Querying xml documents made easy : nearest concept queries. In *Data Engineering, 2001. Proceedings. 17th International Conference*, pages 321–329, 2001.
- [178] G. Shafer. *A mathematical theory of evidence*. Princeton, NJ : Princeton University Press, 1976.
- [179] D.W. Shin, H.C. Jang, and H.L Jin. BUS : an effective indexing and retrieval scheme in structured documents. In *Proceedings of digital libraries, Pittsburgh*, pages 235–243, 1998.
- [180] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An element-based approach to XML retrieval. In *Proceedings of INEX 2003 workshop, Dagstuhl, Germany*, dec. 2003.
- [181] B. Simonnot. *Modélisation multi-agents d'un système de recherche d'information multimédia à forte composante vidéo, (Multi-Agent Modelling of a multimedia information retrieval system for still images and videos collections)*. Phd thesis, Henri Poincaré University, 1996.
- [182] A. Singhal, J. Choi, D. Hindle, and F. C. N.Pereira. ATT at TREC-6 : SDR track. In *Text REtrieval Conference*, pages 227–232, 1997.
- [183] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *20th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 25,32, 1997.
- [184] A. Smeaton and C.J. Van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal.*, 26(3) :239,246, 1983.
- [185] K. Sparck-Jones and R. Needham. Automatic theme classification and retrieval. *Information Processing and Management*, 4 :91,100, 1972.
- [186] J. Spiegel and E. Bennett. A modified statistical association procedure for automatic document content analysis and retrieval. In *Statistical Association Methods For Mechanized Documentation. National Bureau of Standards Miscellaneous Publications 269.*, pages 47–60. M. E. Stevens, V. E. Guiliano and L. B. Heilprin. eds, 1964.
- [187] A. Spink and T. D. Wilson. Toward a theoretical framework for information retrieval (ir) evaluation in an information seeking context. In *Mira '99 : Evaluating Information Retrieval.*, 1999.
- [188] R. G. Sumner, K. Yang, R. Akers, and W. M. Shaw. Interactive retrieval using iris : Trec-6 experiments. In *Proceedings of the Sixth Text Retrieval Conference (TREC-6).*, pages 711–734, 1998.
- [189] C. Sun, C. Chan, and A.K. Goenka. Multiway slca-based keyword search in xml data. In *WWW '07 : Proceedings of the 16th international conference on World Wide Web*, pages 1043–1052, New York, NY, USA, 2007. ACM Press.

- [190] L. Tamine. *Optimisation de requêtes dans un système de recherche d'information*. Phd, Université Paul Sabatier de Toulouse, December 2000.
- [191] A. Theobald and G. Weikum. The index-based XXL search engine for querying XML data with relevance ranking. In *EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic*, pages 477–495, 2002.
- [192] A. Trotman. Choosing document structure weights. *Information Processing and Management*, 41 :243,265, 2005.
- [193] A. Trotman and M. Lalmas. Why structural hints in queries do not help xml-retrieval. In *SIGIR*, pages 711–712, 2006.
- [194] A. Trotman and R. A. O’Keefe. Identifying and ranking relevant document element. In *Proceedings of INEX 2003 Workshop*, pages 149,154. Dagstuhl, Germany, December 2003.
- [195] A. Trotman and R. A. O’Keefe. The simplest query language that could possibly work. In *Proceedings of INEX 2003 Workshop*, pages 167,174. Dagstuhl, Germany, December 2003.
- [196] A. Trotman and B. Sigurbjornsson. Narrowed extended xpath i(nexi). In *INEX 2004 Workshop Proceedings*, pages 16,40. Germany, December 2004.
- [197] H. Turtle. *Inference Networks for Document Retrieval*. Phd thesis, University of Massachusetts, 1991.
- [198] H. Turtle and W.B. Croft. Inference networks for document retrieval. In A. Bookstein, Y. Chiarmella, G. Salton, and V. Raghavan, editors, *Proceedings of ACM SIGIR*, pages 1,24, 1990.
- [199] C. Vogt. *Adaptive combination of evidence for information retrieval*. PhD thesis, University of California, San Diego, 1999.
- [200] Ellen M. Voorhees. The trec robust retrieval track. *SIGIR Forum*, 39(1) :11–20, 2005.
- [201] W3C. DOM Level 1 (Document Object Model). Technical report, World Wide Web Consortium (W3C), W3C standard, october 1998.
- [202] W3C. *Extensible markup language (XML)*. Disponible sur <http://www.w3.org/TR/1998/REC-xml-19980210>, 1998.
- [203] S. Walker, S.E. Robertson, M. Boughanem, G.J.F. Jones, and K. Sparck Jones. Okapi at trec-6 : Automatic ad hoc, vlc, routing, filtering and qsdr.
- [204] F. Weigel, H. Meuss, F. Bry, and K.U. Schulz. Content-aware dataguides : Interleaving IR and DB indexing techniques for efficient retrieval of textual XML data. In *Proceedings of ECIR 2004, Sunderland, UK*, pages 378–393, 2004.
- [205] Z. Wen. New algorithms for the lca problem and the binary tree reconstruction problem. *Information Processing. Lett*, 51(1) :11, 16, 1994.

-
- [206] R. Wilkinson. Effective retrieval of structured documents. In *Proceedings of SIGIR 1994, Dublin, Ireland*, pages 311–317, 1994.
- [207] J.E. Wolff, H. Florke, and A.B. Cremers. Searching and browsing collections of structural information. In *Proceedings of IEEE advances in digital libraries, Washington, 2000*, pages 141–150, 2000.
- [208] H. Wu and G. Salton. The estimation of term relevance weights using relevance feedback. *Journal of Documentation*, 37(4) :194,214, 1981.
- [209] Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest lcas in xml databases. In *SIGMOD '05 : Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 527–538, New York, NY, USA, 2005. ACM Press.
- [210] R.R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18 :pages 183–190, 1988.
- [211] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. XRel : A path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology*, 1(1) :pages 110–141, 2001.
- [212] H. Zargayouna. Contexte et sémantique pour une indexation de documents semi-structurés. In *Actes de CORIA 04, Toulouse, France*, pages 161–178, 2004.
- [213] G. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.
- [214] J. Zobel, A. Moffat, R. Wilkinson, and R. Sacks-Davis. Efficient retrieval of partial documents. *Information Processing and Management*, 31(3) :361–377, 1995.