République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A.MIRA-BEJAIA
Faculté des Sciences Exactes Département d'Informatique

Mémoire

Présenté par

CHAA Messaoud

Pour l'obtention du diplôme de Magister

Filière : Informatique

Option : Réseaux et Systèmes Distribués

Thème

Apprentissage d'ordonnancements en recherche d'information structurée

Soutenu le 20/03/2013 Devant le jury composé de :

Président	TARI Abdelkamel	Maître de Conférences A	Université de Bejaia
Examinateur	IDOUGHI Djilali	Maître de Conférences A	Université de Bejaia
Examinateur	CHALAL Rachid	Maître de Conférences A	ESI-Oued Smar, Alger
Rapporteur	NOUALI Omar	Directeur de Recherche	CERIST, Alger
Invité	BAL Kamal	Attaché de Recherche	CERIST, Alger

Année Universitaire 2012-2013

A ma très chère mère Yamina, et mon très cher père Mohamed
qui se sont tant sacrifiés pour les besoins de mes études,
A ma femme, pour son soutien et sa patience,
A mes adorables petits-enfants, Mehdi et Nour El Imane,
A mes frères et à mes sœurs pour leur affectueux soutien moral.

Je dédie ce modeste travail

REMERCIEMENTS

Je remercie en premier Allah notre grand Dieu pour m'avoir donné le courage et la volonté pour terminer ce modeste travail.

J'adresse tout d'abord mes remerciements à mon directeur de mémoire, Monsieur Omar NOUALI, directeur de recherche au CERIST, pour m'avoir d'abord proposé le sujet, donné l'occasion de travailler sur une thématique pleine de perspectives et m'avoir aidé à conduire ce travail jusqu'au bout.

Je remercie, Monsieur Kamal BAL attaché de recherche au CERIST pour m'avoir orienté, guidé, corrigé mon travail et mis à ma disposition tous les moyens pour la finalité de ce travail, une riche documentation, soutenu et supporté dans des moments délicats, je lui suis profondément reconnaissant.

Je remercie: Mr. Kamel TARI, Mr Rachid CHALAL et Mr Djilali IDOUGHI, pour avoir accepté de juger ce modeste travail.

Je remercie Mr. Mohand BOUGHANEM d'avoir m'accepter d'effectuer un stage d'un mois au sein de l'équipe SIG de L'IRIT(Toulouse).

Un grand MERCI à Mes Parents, Ma Femme, Mes sœurs et Mes frères pour m'avoir soutenu à fin que ce travail arrive à sa fin.

Je remercie mes amis pour leur aide appréciable, leurs encouragements continus et leur soutien moral ininterrompu.

Je voudrais remercier vivement mes collègues du CERIST, et plus particulièrement celles et ceux de la division DRDSI, pour m'avoir poussé à faire le magister, encouragé, motivé et surtout soutenu et aidé avant et durant la préparation de mon mémoire.

Et pour être sûr de n'oublier personne, que tous ceux, qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce modeste travail, trouvent ici l'expression de ma profonde reconnaissance.

RÉSUMÉ

L’adoption accrue de XML comme format standard pour représenter les documents structurés nécessite le développement des systèmes, efficients et efficaces, capable de retrouver les éléments XML pertinents à une requête d'utilisateur. Ces éléments sont ensuite présentés ordonnés en fonction de leur pertinence par rapport à la requête. Généralement la stratégie adoptée consiste à combiner plusieurs sources de pertinences dans une seule fonction de score et le poids de chaque source est donné manuellement selon des méthodes empiriques.

Il est connu que, dans la recherche d’information classique, compte tenu de plusieurs sources de pertinences et l’utilisation des méthodes d’apprentissage d’ordonnancement en combinant ces sources de pertinences, améliore la performance des systèmes de recherche d’information.

Dans ce travail, certains caractéristiques de pertinences des éléments XML, ont été définies et utilisées pour l’apprentissage d’ordonnancement dans les documents structurés. Notre objectif est de combiner ces caractéristiques afin d’obtenir la bonne fonction d’ordonnancement et montrer l’impact de chaque caractéristique dans la pertinence de l’élément XML.

Des expérimentations sur une grande collection de la compagne d’évaluation de la recherche d’information XML (INEX) ont montré la performance de notre approche.

Mots clés: XML, Recherche d’information structurée, Apprentissage d’ordonnancement, Ranking SVM, BM25.

ABSTRACT

The increased adoption of XML as the standard for representing a document structure requires the development of tools to retrieve and rank effectively relevant elements of the XML documents to a user's query. Usually the strategy adopted involves the combination of multiple sources of relevance into a single ranking and the weights of sources are manually tuned.

It's known that, in classical information retrieval, considering multiple sources of relevance and the use of learning to rank methods to combine theses sources of relevance improves information retrieval.

In this work, some relevance features of XML elements, are defined and used in a learning to rank approach for XML information retrieval. Our aim is to combine theses features to derive good ranking function and show the impact of each feature in the relevance of XML element.

Experiments on a large collection from the XML Information Retrieval evaluation campaign (INEX) showed good performance of the approach.

Keywords: XML, structured information retrieval, learning-to-rank, Ranking SVM, BM25.

Table des matières

INTRODUCTION GÉNÉRALE 1
1. Contexte 1
2. Problématique 2
3. Contribution 2
4. Organisation du document 3

CHAPITRE 1 : PRINCIPES DE LA RECHERCHE D'INFORMATION 4
1.1. Introduction 5
1.2. Concepts de base de la recherche d'information 5
1.3. Processus de la recherche d'information 6
1.3.1. Indexation des documents et des requêtes 7
1.3.2. Appariement requête-document et ordonnancement des résultats 7
1.3.3. Reformulation de requêtes 7
1.4. Processus d'indexation 7
1.4.1. Etapes de l'indexation automatique 8
1.4.1.1. Analyse lexicale 8
1.4.1.2. Elimination des mots vides 8
1.4.1.3. Lemmatisation ou normalisation 8
1.4.1.4. Pondération des termes 9
1.4.1.5. Création de l'index 10
1.5. Modèles de recherche d'information 11
1.5.1. Modèle booléen ou ensembliste 11
1.5.2. Modèle vectoriel 12
1.5.3. Modèle probabiliste 13
1.5.4. Modèle de langue 16
1.5.5. Autres modèles de recherche d'information 16
1.6. Evaluation des systèmes de recherche d'information 17
1.6.1. Rappel et précision 17
Courbe de rappel-précision 18
1.6.2. Mesures alternatives 19
1.6.2.1. Mesure harmonique 20
1.6.2.2. Mesure d'évaluation « E » 20
1.6.3. Mesure NDCG 20
1.6.4. Collection de référence(TREC) 21
1.7. Conclusion 21

CHAPITRE 2 : RECHERCHE D'INFORMATION STRUCTUREE 22
2.1. Introduction 23
2.2. Concepts de base des documents XML 23
2.2.1. Notion de structure (balise et élément) 23
2.2.2. DTD (Document Type Defintion) 24
2.2.3. DOM (Document Object Model) 24
2.2.4. XPath 25
2.3. Spécificité de la recherche d'information structurée 25
2.3.1. Granularité de l'information recherchée 25
2.3.2. Approches orientées données et Approches orientées documents 26

2.4. Indexation des documents structurés 27
2.4.1. Indexation de l'information textuelle 27
2.4.1.1. Sous-arbres imbriqués 28
2.4.1.2. Unités disjointes 29
2.4.2. Pondération des termes d'indexation 29
2.4.3. Indexation de l'information structurelle 29
2.4.3.1. Indexation basée sur des champs 30
2.4.3.2. Indexation basée sur des chemins 30
2.4.3.3. Indexation basée sur des arbres 31
2.5. Modèles de recherche d'information structurée 31
2.5.1. Approche par propagation des termes des documents 32
2.5.1.1. Modèle vectoriel étendu 32
2.5.1.2. Modèle de langue 34
2.5.1.3. Modèle XIVIR 35
2.5.2. Approche par propagation des scores des éléments 36
2.5.2.1. Modèle GPX 36
2.5.2.2. Modèle XFIRM 37
2.6. Evaluation des systèmes de recherche d'information structurée 39
2.6.1. Compagne d'évaluation 39
2.6.1.1. Collection de référence 39
2.6.1.2 Requêtes 39
2.6.1.3 Jugements de pertinence 40
2.6.2. Mesures d'évaluation 42
2.6.2.1. Gain cumulé étendu (XCG) 43
2.6.2.2. HiXEVAL 44
2.7. Conclusion 45

CHAPITRE 3 : APPRENTISSAGE D'ORDONNANCEMENT EN RECHERCHE D'INFORMATION 46
3.1. Introduction 47
3.2. Apprentissage automatique 47
3.2.1. Apprentissage automatique supervisé 48
3.2.2. Ensembles de données et généralisation du modèle 50
3.3. Apprentissage des fonctions d'ordonnancement 50
3.3.1. Approche pointwise : 52
3.3.1.1. Ordonnancement par régression 53
3.3.1.2. Ordonnancement par classification binaire 54
3.3.1.3. Régression ordinale 55
3.3.1.4. Limite de l'approche pointwise 56
3.3.2. Approche pairwise 56
3.3.2.1. RankBoost 57
3.3.2.2. Ranking SVM 58
3.3.2.3. Avantage des approches pairwise 60
3.3.3. Approche listwise 60
3.3.3.1. ListMLE 61
3.3.3.2. Avantage des approches listwise 62
3.4. Apprentissage d'ordonnancement et RIS 62
3.4.1. Modèle pairwise pour la RIS 62
3.4.2. Modèle listwise pour la RIS (ListBM). 63

3.5. Evaluation des algorithmes d'apprentissage d'ordonnancement 65
Collections de référence 65
3.6. Conclusion 66

CHAPITRE 4 : APPROCHE PROPOSEE ET EVALUATION DES RESULTATS ..67
4.1. Introduction 68
4.2. Modélisation de l'approche proposée 68
4.3. Algorithme choisi 70
4.4. Caractéristiques utilisées 72
4.4.1. Score du document (BM25Doc) 72
4.4.2. Score de l'élément (BM25Elem) 73
Calcul de « ief » 73
4.4.3. Proximité des termes de la requête (ProxElem) 74
4.4.4. Taille de l'élément (SizeElem) 75
Proposition d'un nouveau score 75
4.5. Expérimentations et évaluation 77
4.5.1. Données expérimentales 77
4.5.1.1. Indexation 77
4.5.1.2. Traitement des requêtes 79
4.5.2. Echantillonnage des requêtes d'apprentissage 79
4.5.3. Echantillonnage des éléments d'apprentissage 80
4.5.4. Etiquetage des couples élément-requêtes 80
4.5.4.1. Problème d'étiquetage 81
4.5.4.2. Solution proposée 81
4.5.5. Apprentissage et tests 82
4.5.6. Evaluation des résultats 83
4.5.6.1 . Combinaison de toutes les caractéristiques 83
4.5.6.2 . Etude de l'impact de chaque caractéristique 85
4.6. Conclusion 87
CONCLUSION GÉNÉRALE 88
Perspectives 89
BIBLIOGRAPHIE 91

Table des illustrations

Figure 1. 1 : Processus de recherche d'information [Mar 10] 6
Figure 1. 2 : Précision et rappel 18
Figure 1. 3 : la courbe rappel-précision 19
Figure 2. 1 : Exemple de document XML article.xml. 23
Figure 2. 2 : Exemple de DTD. 24
Figure 2. 3 : Exemple d'arbre DOM correspondant au document de la figure 2.1. 25
Figure 2. 4 : indexation des documents XML. 28
Figure 2. 5 : Exemple de recherche par structure avec le système XIVIR [Aou 09]. 36
Figure 2. 6 : Exemple de requête de type CO+S issue de la compagne d'évaluation INEX'2005. 40
Figure 2. 7 : Extrait d'un fichier de jugement de pertinence explique l'exhaustivité et la spécificité des éléments XML. 42
Figure 3. 1 : Apprentissage automatique supervisé. 49
Figure 3. 2 : Le cadre générale de l'apprentissage d'ordonnancement [Liu 11] 51
Figure 3. 3 : Fonction d'erreur des moindres carrés. 54
Figure 3. 4 : Fonction d'erreur charnière (Hinge Loss). 55
Figure 3. 5 : Une illustration de la règle de modification dans Prank [CRA 02]. 56
Figure 3. 6 : Machine à vecteurs de support SVM. 58
Figure 3. 7 : Transformation du problème de Ranking en classification par paires. 58
Figure 3. 8 : ListBM : algorithme pour apprendre k*1* de BM25 [Gao 09]. 64
Figure 4. 1 : Modélisation de l'approche proposée pour l'apprentissage d'ordonnancement en RIS. 70
Figure 4. 2 : Algorithme Ranking SVM 72
Figure 4. 3 : Distribution de la taille des éléments évalués strictement pertinent dans la Collections INEX 2005[Sig 06]. 75
Figure 4.4 : Requête de type CO+S. 79
Figure 4. 5 : Fichier d'apprentissage *train.txt*. 82
Figure 4. 6 : Fichier résultat de l'apprentissage *model.txt*. 82
Figure 4. 7 : Courbe de ep/gr de notre approche et des résultats officiels de la campagne d'évaluation INEX 2005, tâche CO, quantification fgen. 84
Figure 4. 8 : Comparaison avec les participants officiels d'INEX'2005 selon la mesure nxCG, tâche CO, quantification fgen. 84

Liste des tableaux

Tableau 1.1 : Fichier inverse. 11
Tableau 1.2 : Table de contingence des termes [Bou 08]..... 15
Tableau 1.3 : Calcul de précision et de rappel. 19
Tableau 2.1 : Système de recherche d'information vs SGBD. 26
Tableau 2.2 : Exemple d'indexation basé sur des champs. 30
Tableau 2.3 : Exemple d'indexation basé sur des chemins. 30
Tableau 2.4 : Exemple d'indexation basé sur des arbres. 31
Tableau 3.1 : Résumé des approches de l'apprentissage de fonction d'ordonnancement [Liu11]. 52
Tableau 3.2 : statistiques des collections de l'apprentissage d'ordonnancement..... 65
Tableau 4.1 : exemple de trois éléments et leurs vecteurs de caractéristiques. 71
Tableau 4.2 : les nouvelles instances avec leurs vecteurs de caractéristiques après la transformation par paire. 71
Tableau 4.3 : Types d'éléments sélectionnés de la collection. 74
Tableau 4.4 : Les tables principales de la base de données. 78
Tableau 4.5 : Partitionnement des requêtes pour trois-fold cross validation. 80
Tableau 4.6 : Etiquetage des couples élément-requêtes. 82
Tableau 4.7 : Evaluation des résultats de notre Approche, BM25 et les deux meilleures expérimentations soumises à la compagnie INEX 2005 sur la mesure MAnxCG. 85
Tableau 4.8 : Résultats montre l'impact de chaque caractéristique. 86

INTRODUCTION GÉNÉRALE

1. Contexte

Ces dernières années, avec la croissance rapide du World Wide Web (WWW) et les difficultés trouvées pour chercher l'information désirée, des systèmes de recherche d'information (SRI), efficaces et efficients, sont devenus plus importants que jamais et le moteur de recherche est devenu un outil essentiel pour trouver cette information. Il est responsable de la mise en correspondance entre les requêtes traitées et les documents indexés. L'ordonnancement (Ranking) des résultats trouvés est un élément essentiel dans chaque moteur de recherche. En raison de son rôle central, une grande attention a été accordée à la recherche et le développement des modèles d'ordonnancement (modèles de Ranking).

L'adoption accrue de XML comme format standard pour représenter les documents structurés nécessite aussi le développement des systèmes efficaces pour la recherche d'information structurée (RIS), dont le but est de trouver des éléments XML pertinents (paragraphe, section, sous section,…etc.) et non pas des documents entiers. Ces éléments sont ensuite présentés ordonnés en fonction de leur pertinence par rapport à la requête.

Pour cette raison, les systèmes de recherche d'information structurée font appel à une fonction d'ordonnancement, dont le but est de trouver les éléments d'une collection pertinents à une requête, renvoie à l'utilisateur une liste d'éléments ordonnés par leur pertinence, souvent calculée sous la forme d'un score réel. L'ordre de cette liste est primordial puisqu'en général les utilisateurs s'intéressent surtout aux premiers éléments retournés pour leur requête. Le problème donc, est de savoir définir une fonction de score capable de positionner les éléments les plus pertinents au début de la liste de résultats. Pour cela, plusieurs fonctions de calcul de score de pertinence ont été mises au point.

La plupart des approches de recherche d'information structurée ont été des adaptations des modèles de recherche d'information classique au contexte XML (modèle vectoriel, probabiliste et modèle de langue). D'autres modèles spécifiques à la recherche d'information structurée ont été aussi développés pour restituer une liste ordonnée d'éléments XML pertinents. Pour ce faire des fonctions de scores sont définies. Ces fonctions attribuent un score de pertinence à chaque élément XML. Pour arriver à définir ces fonctions de scores, un certain nombre de source de pertinence sont choisies, combinées puis pondérées dont le poids de chacune d'elles est fixé manuellement d'une manière empirique après plusieurs expérimentations.

Bien évidemment, avec le nombre croissant des sources de pertinences à prendre en compte, l'approche manuelle est devenue difficile à mettre en œuvre. Pour pallier à cette difficulté, nous proposons dans ce travail une approche à base d'apprentissage qui permettre de combiner et pondérer automatiquement un certain nombre de source de pertinence.

2. Problématique

Dans la recherche d'information structurée, Il n'est pas encore possible de décider quel modèle fonctionne le mieux. Une difficulté réside encore dans la définition de la fonction de score de pertinence. Cette difficulté est due aux nombreux facteurs (sources de pertinences) à prendre en compte pour estimer la pertinence des éléments XML. Ces facteurs sont variés et hétérogènes (par exemple, la taille de l'élément, le type de l'élément, la pertinence de ses ancêtres, la proximité entre les termes de la requête dans l'élément, …etc.).

En effet, la recherche d'information dans les documents XML peut être considérée comme un problème de combinaisons où le but est de décider :

1- Quelles sont les sources de pertinence à utiliser pour estimer la pertinence des éléments XML par rapport aux requêtes des utilisateurs.

2- Comment les combiner pour améliorer la performance de la recherche d'information structurée.

3- comment les pondérer pour avoir des bons résultats du système de recherche d'information structurée.

L'un des objectifs est alors de tenter de répondre à ces trois questions en utilisant les méthodes à base d'apprentissage automatique.

3. Contribution

Afin de répondre aux questions de la problématique, nous avons proposé une approche à base d'apprentissage automatique pour l'ordonnancement des éléments XML. Quatre sources de pertinences (caractéristiques) des éléments XML, qui semblent être efficaces pour la RIS, ont été choisies et définies. Un algorithme d'apprentissage d'ordonnancement, Ranking SVM, a été utilisé pour construire un nouveau modèle pour la recherche d'information structurée.

L'apprentissage et l'évaluation ont nécessité plusieurs traitements, à savoir la préparation des données d'apprentissage (Indexation, échantillonnage des éléments et des requêtes, étiquetage des couples élément-requêtes et extraction des caractéristiques).

Des expérimentations sur la collection de référence INEX 2005 nous ont permit de valider notre approche et mesurer l'impact de chaque caractéristique.

Les résultats obtenus sont encourageants et l'approche proposée montre des améliorations significatives pour la recherche d'information structurée.

4. Organisation du document

Ce mémoire est organisé en quatre chapitres principaux.

Le chapitre 1 présente les différents principes de la RI. Nous donnons dans un premier temps les concepts de base de la RI. Nous abordons ensuite le principe de l'indexation qui est une étape préliminaire importante en RI. Nous indiquons quelques modèles utilisés par les systèmes de RI pour calculer la similarité entre les documents et les requêtes. Afin de mesurer l'efficacité de la recherche, une phase d'évaluation est décrite. Nous détaillons les mesures d'évaluation et la campagne TREC.

Le chapitre 2 est consacré à la présentation d'un état de l'art sur les travaux existants dans le domaine de la recherche d'information structurée. Nous exposons les différents aspects de la RIS (modèles de recherche d'information, techniques d'indexation et mesures d'évaluation de la RIS dans la compagne INEX).

Dans le chapitre 3, nous présentons les concepts fondamentaux de l'apprentissage automatique supervisé. Nous mettons l'accent sur les différentes approches de l'apprentissage de fonction d'ordonnancement en RI et en RIS. Nous terminons ce chapitre en présentant la collection de référence LETOR utilisé pour l'évaluation des algorithmes d'apprentissage d'ordonnancement.

Dans le chapitre 4, nous expliquons l'adaptation de l'algorithme Ranking SVM sur la recherche d'information structurée, en détaillant les caractéristiques choisies, les éléments XML échantillonnés et la méthode de l'étiquetage des couples élément-requête adoptée. Nous exposons également les expérimentations que nous avons réalisées afin de valider notre approche.

Enfin, ce mémoire se termine avec une conclusion générale et des perspectives.

CHAPITRE 1 : PRINCIPES DE LA RECHERCHE D'INFORMATION

1.1. Introduction

Quelques années après l'invention de l'ordinateur, les années soixante, la RI est apparue pour répondre au besoin de gérer l'explosion de la quantité d'informations. C'est la science qui s'intéresse à l'acquisition, l'organisation, la recherche et la sélection de l'information. Afin de répondre à un besoin d'information le monde de la recherche s'est intéressé aux SRI dans lesquels sont mis en œuvre des techniques et des mécanismes assurant la gestion automatique des informations documentaires. Ce chapitre a pour objectif de présenter les principaux concepts de la RI, les principaux modèles de recherche ainsi que les approches d'évaluation des SRI.

1.2. Concepts de base de la recherche d'information

La tâche principale d'un système de recherche d'information (SRI) est de trouver à partir d'une base de documents ceux qui sont susceptibles de répondre aux besoins en information de l'utilisateur. Son but est de retourner à l'utilisateur une liste de documents ordonnés contient le maximum de documents pertinents pouvant satisfaire son besoin et le minimum de documents non pertinents.

Dans cette définition, il y a trois notions clés: document, requête, pertinence.

- Document: Un document peut être un texte, un morceau de texte, une page web, une image, une bande vidéo, etc. elle constitue l'information élémentaire recherchée par le SRI.

Requête: l'utilisateur exprime son besoin sous forme d'une requête, une suite de mots clés souvent séparés par des opérateurs logiques (et, ou et non), ou par des variables linguistiques telles que proche de, contient. . .

On distingue trois types de requêtes :

- Les requêtes basiques, la requête est un ensemble de mots-clés,
- Les requêtes logiques (booléennes), la base des requêtes est un ensemble d'opérateurs logiques (et, ou et non),
- Les requêtes structurées, ce type de requêtes porte des informations non seulement sur le contenu mais aussi des informations sur la structure des documents telles que en-têtes, titres . . . etc.

Pertinence: La pertinence est une notion fondamentale dans le domaine de la RI. Une définition simple de cette notion complexe est donnée dans **[Sau 05]** : "La pertinence est la

correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête". Il est l'objet de tout système de recherche d'information. Un document est dit pertinent lorsqu'il satisfait les besoin de l'utilisateur, et non pertinent dans le cas contraire. La notion de pertinence est alors fortement subjective, c'est-a-dire dépendante de l'utilisateur.

1.3. Processus de la recherche d'information

Afin d'être en mesure d'offrir aux utilisateurs les informations correspondants à leurs attentes, Le processus de recherche d'information met en œuvre un certain nombre de processus qui permettent de mettre en relation l'ensemble des informations disponibles dans le fond documentaire d'une part et les besoins en information des utilisateurs d'une autre part.

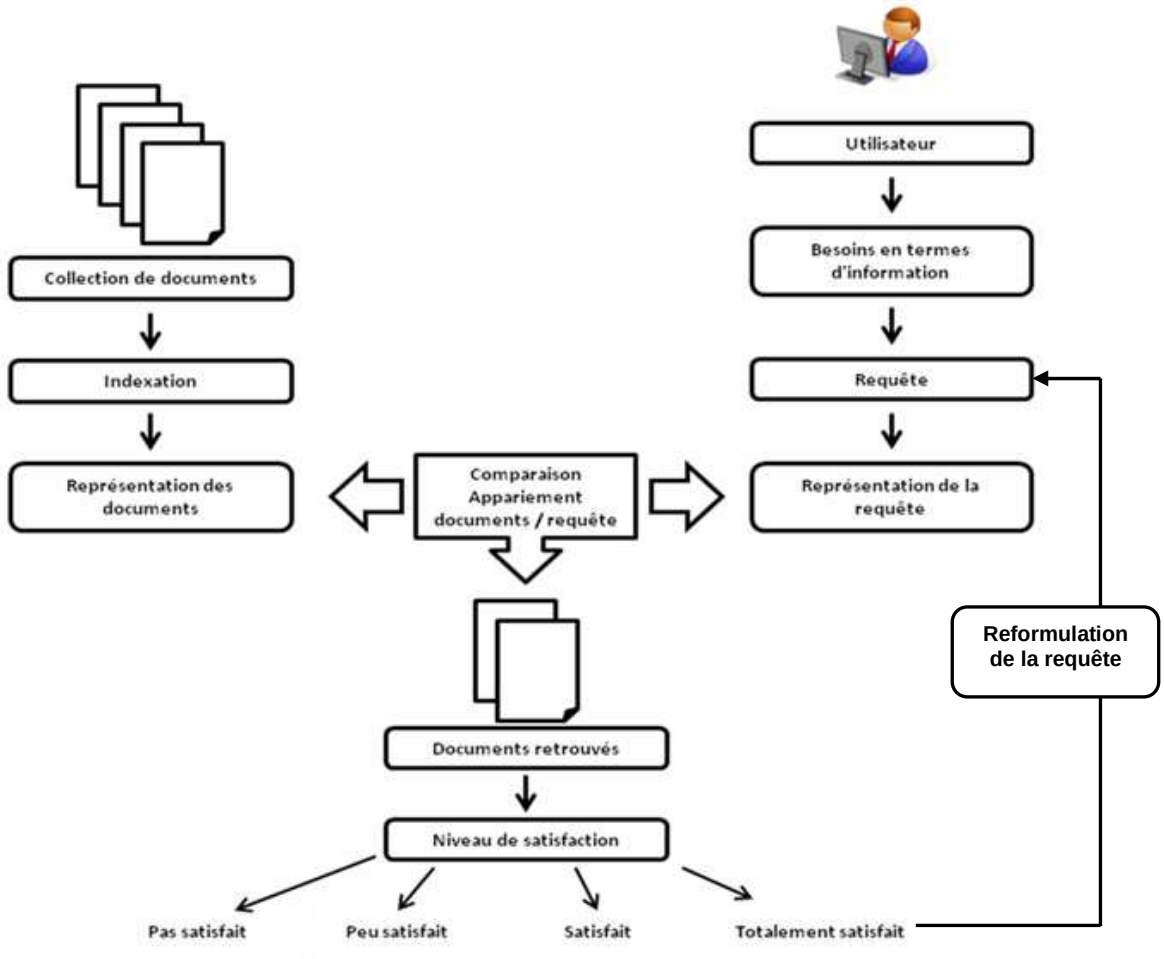

Figure 1. 1 : Processus de recherche d'information [Mar 10]

Ce processus est composé de trois fonctions principales:

1.3.1. Indexation des documents et des requêtes

L'indexation est une technique de passer d'un document textuel a une représentation exploitable par un modèle de RI, Elle a pour but d'extraire à partir d'un document ou d'une requête une représentation paramétrée qui couvre au mieux son contenu sémantique. Le résultat de l'indexation est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids pour différencier leur degré d'importance.

1.3.2. Appariement requête-document et ordonnancement des résultats

Les résultats d'un SRI doit s'appuyer sur un modèle de pertinence des documents. Celui-ci va ainsi permettre de réaliser, pour chaque requête, un calcul de score de pertinence pour chacun des documents, ce score est généralement calculé à partir d'une fonction, qui utilise plusieurs sources de pertinence, notée *RSV(Q,D)* (Retrieval Slatus Value), où *Q* est une requête et *D* un document. Les documents sont ensuite présentés aux utilisateurs par ordre décroissant et ceux qui auront le meilleur score de pertinence seront alors au début de la liste. On parlera de calcul de ranking ou fonctions d'ordonnancement. Si les documents recherchés ne sont pas présents au début de cette liste, l'utilisateur considérera le SRI comme mauvais vis-à-vis de sa requête.

1.3.3. Reformulation de requêtes

L'idée de La reformulation de requête est d'impliquer l'utilisateur dans le processus de contrôle de pertinence de la recherche, afin d'améliorer le résultat final. Il consiste à modifier les requêtes en fonction des résultats présentés et le jugement donné par l'utilisateur.

1.4. Processus d'indexation

La stratégie de recherche séquentielle ne peut pas être réalisable pour accéder à une grande quantité d'informations dans des délais acceptables. Pour résoudre ce problème d'accès afin d'optimiser le cout de la recherche, une étape primordiale doit s'effectuer sur les documents avant l'étape de recherche effective de l'information. Cette étape est appelée indexation.

L'indexation peut être :

Manuelle : chaque document est analysé par un spécialiste du domaine ou par un documentaliste afin d'extraire les termes significatifs des documents ;

Semi-automatique : Le processus automatique extraire les termes des documents. Cependant, un spécialiste ou un documentaliste pour le choix final des termes significatifs ;

Automatique : Le processus d'indexation est entièrement informatisé; Selon Salton et McGill **[Sal 83]**, le but de l'indexation automatique est de « transformer des documents en substituts capables de représenter leurs contenus ».

Une étude comparative entre l'indexation automatique et l'indexation manuelle a été faite par Anderson et Perez-Carballo **[And 01].** Cette étude montre que le choix de l'une ou de l'autre est en fonction du domaine, de la collection et de l'application considérés.

Comme l'indexation automatique est la technique la plus répandue dans le domaine de recherche d'information, à cause de la taille des collections de documents utilisées(indexation manuelle très coûteuse), nous allons décrire les étapes essentielles de cette technique qui regroupe plusieurs traitements automatique sur un document, de l'extraction des mots jusqu'à la création de l'index.

1.4.1. Etapes de l'indexation automatique

1.4.1.1. Analyse lexicale

C'est le processus qui permet d'extraire l'ensemble des termes (unité lexicales) constituant un document, aussi appelées *tokens*. Cette analyse permet de reconnaitre les espaces de séparation, les chiffres et les ponctuations.

1.4.1.2. Elimination des mots vides

L'une des étapes importante dans le processus d'indexation qui a une grande influence sur la précision de la recherche, est l'élimination des mots vides (pronoms personnels, prépositions...). Ce sont des mots peu significatifs et porteurs de peu de sens. Pour éliminer ces mots vides, on a recours à deux techniques :

- L'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire ou stoplist).
- L'´elimination des mots dépassant un certain nombre d'occurrences dans la collection ou les mots rares de la collection.

1.4.1.3. Lemmatisation ou normalisation

Elle consiste à réduire les mots à leur racine grammaticale, car un mot donné peut avoir différentes formes dans un texte, mais leur sens reste très similaire. Plusieurs types stratégiques de lemmatisation ont été proposé dans la littérature **[Fra 92]** : la table de

consultation (dictionnaire), l'élimination des affixes (algorithme de Porter **[Por 80]),** la troncature, les variétés de successeurs ou encore la méthode des n-grammes **[Ada 74]**.

1.4.1.4. Pondération des termes

La pondération des termes est l'une des fonctions fondamentales en RI. Elle a pour but de trouver les termes représentant le mieux le contenu d'un document en mesurant l'importance de ces termes dans le document. Cette importance est calculée par des méthodes qui se basent généralement sur des considérations statistiques basées sur la distribution des termes dans les documents. Ces méthodes tirent leur origine de la loi de Zipf et de la conjecture de Luhn.

La loi zipf : En 1949, Zipf a constaté une certaine régularité sur la fréquence d'apparition des mots **[Zip 49]**. Ainsi, après le classement des mots par ordre de leurs fréquences décroissantes, il a constaté que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste.

*Rang * fréquence = constante*

A partir de ces constations, des techniques de pondération sont apparues, et la plupart de ces techniques se basent sur deux facteurs, à savoir : la pondération locale et la pondération globale.

tf (Term Frequency) : cette mesure indique l'importance du terme par rapport à un document donné (pondération locale). Cette importance est proportionnelle à la fréquence du terme. Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons (log(tf), présence/absence,...).

idf (Inverse of Document Frequency) : ce facteur mesure l'importance d'un terme par rapport à la collection (pondération globale). Il traduit l'impact d'un terme selon son nombre d'apparitions dans la base documentaire. La formule qui calcul l'importance d'un terme dans la collection est : log(N/df), où df représente le nombre de documents contenant le terme et N le nombre total de documents de la collection.

La combinaison des deux mesures (tf * idf) est une bonne approximation de l'importance du terme dans un document, notamment dans les corpus de documents de tailles homogènes. Contrairement dans les collections de taille très variable les termes appartenant aux documents longs apparaissent très fréquemment et l'emportent en poids sur les termes

appartenant à des documents moins longs. Pour résoudre ce problème, Robertson **[Rob 94]** et Singhal et al. **[Sin 96]** ont intégré la taille des documents à la formule de pondération.

La formule la plus utilisée dans le domaine de la RI est celle de Robertson:

$$w_{ij} = \frac{tf_{ij}\,(k_1+1)\;x\;log\frac{N-n_i+0.5}{n_i+0.5}}{k_1\;x\left((1-b)+\;b\frac{ld_j}{avg_{ld_j}}\right)+\;tf_{ij}} \qquad (1.1)$$

Où:

tf_{ij} : La fréquence du terme t_i dans le document d_j ;

ld_j : La taille du document d_j;

avg_{ld_j} : La taille moyenne des documents dans la collection;

- k_1 et b sont des paramètres. Dans la campagne d'évaluation TREC4. Les meilleurs résultats ont été obtenus lorsque *k1 = 1.2*, *b = 0.75* **[Bou 08]**.

1.4.1.5. Création de l'index

Le résultat final du processus d'indexation et de créer l'index, c'est une structures de stockage pour mémoriser les informations sélectionnées lors de ce processus, qui permet de répondre plus rapidement a une requête et de sélectionner pour n'importe quelle terme les documents où il appartient. Plusieurs techniques de stockage ont été développées parmi lesquelles : les fichiers inverses (”inverted files”), les tableaux de suffixes (”suffix arrays”) et les fichiers de signatures (”signature files”).

Les fichiers inverses sont actuellement les plus utilisés pour la plupart des applications. Ils sont composés de deux éléments principaux :

– Le vocabulaire: l'ensemble de tous les mots différents du texte ;

– Les occurrences (posting) : la liste de toutes les positions de chaque mot dans le texte pour lesquelles il apparait.

Le tableau 1.1 ci-dessous montre un exemple d'un fichier inverse pour un document :

Document :	"Les fichiers inverses sont actuellement les plus utilisé"
Positions	1 5 14 23 28 41 45 50
Fichier inverse	
Vocabulaire	**Posting**
Fichier	5
Inverse	14
Actuel	28
Utiliser	50

Tableau 1.1 : Fichier inverse.

1.5. Modèles de recherche d'information

Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence **[Sau 05]**. On distingue trois principaux modèles :

Les modèles ensemblistes: reposent sur la théorie des ensembles, les termes de la requête sont séparés par des opérateurs logiques.

Les modèles algébriques : Se basent sur la théorie algébrique. La pertinence est définie par des mesures de similarité dans un espace vectoriel.

Les modèles probabilistes: se basent sur la théorie des probabilités. La pertinence est vue comme une probabilité de pertinence document/requête.

Dans cette section, nous décrivons pour chacun de ces courants le modèle le plus représentatif (à savoir : le modèle booléen, le modèle vectoriel et le modèle probabiliste).

1.5.1. Modèle booléen ou ensembliste

Le modèle booléen **[Sal 71]** est l'un des premiers modèles utilisés en recherche d'information. Les documents sont représentés par des ensembles de termes et les requêtes sont traitées comme des expressions logiques. Le modèle booléen considère que les termes de l'index sont présents ou absents d'un document. En conséquence, les poids des termes dans

l'index sont binaires (présent/absent). Si *D* est un document et *t* un terme, on définit l'expression *RSV (D,t)* comme étant vraie si le terme *t* se trouve dans le document *D* et comme fausse, s'il en est autrement**[Mar 10]**.

$$RSV(D,t) = \begin{cases} 1 \; si \; t \; appartient \; au \; document \; D \\ 0 \; sinon \end{cases} \quad (1.2)$$

La formulation de la requête se base sur les trois opérateurs ET, OU et SAUF qui influent sur les valeurs booléennes :

1) La conjonction **ET** : R (D, t1 ET t2) : est vrai si et seulement si le document D contient les deux termes t1 et t2.
2) La disjonction **OU** : R (D, t1 OU t2) : est vrai si au moins un des termes soit présent dans le document D.
3) La négation **SAUF** : R (D, t1 SAUF t2) : est vrai si le document D contient t1 et ne contient pas t2.

Avantages et inconvénients

Le principal avantage du modèle est sa transparence. Le système sélectionne les documents qui répondent exactement à la requête formulée par l'utilisateur, un document est soit pertinent soit non pertinent. L'inconvénient de ce modèle est qu'il rend la tâche de formulation de la requête par l'utilisateur plus complexe, à cause de sa manière d'appariement. En outre, il est incapable de fournir une liste ordonnée de documents car la perception de la pertinence selon le modèle booléen est très différente de celle de l'utilisateur.

1.5.2. Modèle vectoriel

Le modèle vectoriel est un modèle statistique qui consiste à représenter les documents et les requêtes sous forme de vecteurs de termes pondérés.

Gérard Salton **[Sal 70]** et son équipe dans le projet SMART (Salton's Magical Automatic Retriever of Text) ont proposé de représenter les documents et les requêtes sous forme de vecteurs dans l'espace vectoriel engendré par les termes extraits de tous les documents de la collection. La pertinence d'un document par rapport à la requête est évaluée par le degré de similarité entre le vecteur du document *D* et celui de la requête *Q*

Contrairement au modèle booléen où les requêtes sont traitées comme des expressions logiques, le modèle vectoriel permet à l'utilisateur de formuler sa requête sous forme d'une liste de mots clés ou en langage naturel.

La représentation Formelle du document et de requête est la suivante :

- Le document $\overrightarrow{d_j} = (w_{1j}, w_{2j}, \dots, w_{tj})$ où w_{ij} représente le poids des termes dans le document D.
- La requête $\vec{q} = (w_{1q}, w_{2q}, \dots, w_{tq})$

t étant le nombre total de termes de l'index,

Une des plus simples mesures de similarité est celle du produit scalaire :

$$RSV(\overrightarrow{d_j}, \vec{q}) = \sum_{i=1}^{t} w_{ij} * w_{iq} \qquad (1.3)$$

Plusieurs fonctions de similarité ont été proposées dans la littérature. Parmi lesquelles on peut citer les mesures de Cosinus, Jaccard, Dice et Overlap.

Indice de Cosinus $$RSV(\overrightarrow{d_j}, \vec{q}) = \frac{\sum_{i=1}^{t} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{t} {w_{iq}}^2} X \sqrt{\sum_{i=1}^{t} {w_{ij}}^2}}$$

Indice de Jaccard $$RSV(\overrightarrow{d_j}, \vec{q}) = \frac{\sum_{i=1}^{t} w_{ij} * w_{iq}}{\sum_{i=1}^{t} w_{iq} + \sum_{i=1}^{t} w_{ij} - \sum_{i=1}^{t} w_{ij} * w_{iq}}$$

Indice de Dice $$RSV(\overrightarrow{d_j}, \vec{q}) = \frac{2\sum_{i=1}^{t} w_{ij} * w_{iq}}{\sum_{i=1}^{t} w_{iq} + \sum_{i=1}^{t} w_{ij}}$$

Overlap $$RSV(\overrightarrow{d_j}, \vec{q}) = \frac{\sum_{i=1}^{t} w_{ij} * w_{iq}}{Min(\sum_{i=1}^{t} w_{iq}, \sum_{i=1}^{t} w_{ij})}$$

Malgré sa simplicité, le modèle vectoriel est aujourd'hui l'un des modèles les plus populaires dans le domaine de la recherche d'information grâce à ca capacité d'ordonner les résultats retrouvés par rapport aux autres modèles.

1.5.3. Modèle probabiliste

Le premier modèle probabiliste a été proposé par Maron et Kuhns **[Mar 60]** au début des années 1960. Le principe de base est que la mesure de similarité et l'ordonnancement des résultats de recherche d'un SRI sont fondés sur la probabilité de pertinence d'un document vis-à-vis d'une requête. Robertson **[Rob 77]** résume ce critère d'ordre par PRP (Probability Ranking Principle).

Etant donnée une requête *Q* et un document *D*, le modèle PRP peut être traduit comme suit: quelle est la probabilité que le document *D* soit pertinent pour la requête *Q*?

R = pertinence, *NR* non pertinence

$P(R|D)$ est la probabilité que *D* soit pertinent et $P(NR|D)$ la probabilité que D soit non pertinent. Les documents seront classés par la fonction de Similarité suivante :

$$RSV(Q,D) = \frac{P(R|D)}{P(NR|D)} \quad (1.4)$$

On applique le théorème de bayes

$$RSV(Q,D) = \frac{P(D|R)P(R)}{P(D|NR)P(NR)} \quad (1.5)$$

$\frac{P(R)}{P(NR)}$ est le même pour tout les documents de la collection, l'indice de similarité revient donc:

$$RSV(Q,D) = \frac{P(D|R)}{P(D|NR)} \quad (1.6)$$

P(D|R) est la probabilité d'obtenir le document *D* en tirant un document de l'ensemble pertinent et *P(D |NR)* est la probabilité d'obtenir le document *D* en tirant un document de l'ensemble non pertinent.

On décompose l'évènement « tirer le document *D* » comme l'intersection des événements

« t_i=1 tirer un document contenant le descripteur t_i»

« t_i=0 : tirer un document ne contenant pas le descripteur t_i»

P(D|R) = P(t_1=x_{j1}, t_2=x_{j2}, t3=x_{j3}, ... t_n=x_{jK}|R)

P(D|NR) = P(t_1=x_{j1}, t_2=x_{j2}, t3=x_{j3}, ... t_n=x_{jK}|NR)

$$P(D|R) = \prod_{i=1}^{n} P(t_i = x_i|R) = \prod_{i=1}^{n} P(t_i = 1|R)^{xi} * P(t_i = 0|R)^{1-xi} \quad (1.7)$$

On fait le même développement pour *P(D|NR)*. Notons *p(t_i=1|R)*, par P_i, et *p(t_i=1|NR)* par q_i, et après transformation RSV peut s'écrire comme suit:

$$RSV(Q,D) = \prod_{i=1}^{n} \frac{{p_i}^{xi}(1-p_i)^{(1-xi)}}{{q_i}^{xi}(1-q_i)^{(1-xi)}} \quad (1.8)$$

Après un développement en utilisant la fonction log, RSV peut s'écrire alors:

$$RSV(Q,D) = \sum_{i=1} log \frac{p_i(1-q_i)}{q_i(1-p_i)} \quad (1.9)$$

Une manière pour estimer p_i et q_i est de considérer que nous avons des échantillons de documents pertinents et non pertinents pour la requête, puis en fonction de la présence ou l'absence d'un terme dans un document pertinent on peut estimer ces paramètres.

	Pertinent	Non pertinent	Total
Contenant le terme t_i	r_i	$n_i - r_i$	n_i
Ne contenant pas le terme t_i	$R - r_i$	$Ni - n_i - R + r_i$	$N - n_i$
	R	N-R	N

Tableau 1.2 : Table de contingence des termes [Bou 08].

N: le nombre de documents.

R: le nombre de documents pertinents pertinent pour la requête Q.

ni : le nombre de document contenant le terme.

ri : le nombre de document pertinents contenant le terme.

On a alors : $p_i = \frac{ri}{R}$ et $q_i = \frac{ni - ri}{N - R}$

Après remplacement de p_i et q_i et ajoutant la valeur 0.5 aux valeurs centrales de la table, la fonction s'écrit alors:

$$RSV(Q,D) = \sum_{i=1} log \frac{(ri+0.5)(N-ni-R+ri+0.5)}{(R-ri+0.5)(ni-ri+0.5)} \quad (1.10)$$

L'inconvénient de ce modèle est que sans la disponibilité des collections d'entrainement on ne peut pas estimer ses paramètres. Pour résoudre ce problème, S.Roberston a proposé le modèle 2-poisson [Rob 94]. Le résultat était la formule BM25, largement utilisée dans les applications de la recherche d'information **[Bou 08]**.

$$RSV(Q,D) = \frac{qtf*(k_2+1)}{k_2*qtf} * \frac{tf_{ij}(k1+1)*log\frac{N-ni+0.5}{ni+0.5}}{k1*\left((1-b)+b\frac{dl_j}{avg_{dl}}\right)+tf_{ij}} \quad (1.11)$$

ni : le nombre de documents contenant ti

N : le nombre de documents pertinents dans la collection

dl_j : la longueur du document dj

avg_dl : la longueur moyenne des documents de la collection

tfij : la fréquence d'apparition du terme ti dans le document *dj*.

qtf : la fréquence d'apparition du terme ti dans la requête *q*.

k1, *k2* et *b* sont des constantes. Apres plusieurs expérimentations sur les collections TREC, les valeurs *k1=1.2, k2=8 et b=0.75* donnent des meilleurs résultats **[Bou 08]**.

1.5.4. Modèle de langue

Le principe des modèles de langue est de déterminer la probabilité de générer la requête Q à partir du document D, alors que dans les modèles probabilistes, on évalue la probabilité de pertinence d'un document vis-à-vis d'une requête.

Etant donné une requête Q =(t_1, t_2, . . . , t_n), Md est le modèle de langue du document D, la pertinence est mesuré par :

$$RSV(Q,D) = P(Q|M_d) = \prod_{i=1}^{n} P(T_i|D) \quad (1.12)$$

$P(T_i|D) = \frac{tf(T_i|D)}{\sum_T tf(T,D)}$, Où $tf(T_i|D)$ est la fréquence du terme *Ti* dans le document D.

On constate que si un terme de la requête est absent du document nous aurons systématiquement $(\mathrm{Q},\mathrm{D}) = 0$, Afin de résoudre ce problème, des techniques de lissage (*smoothing parameter)* peuvent être utilisées pour assigner des probabilités non nulles aux termes qui n'apparaissent pas dans le document. Une des approches proposées dans ce cadre est le lissage par le modèle de collection **[Hie 98]**. La pertinence d'un document vis-à-vis d'une requête est alors mesurée par une combinaison linéaire du modèle de document et du modèle de la collection **[Bou 08].**

Elle est donnée par:

$$\mathrm{RSV}(\mathrm{Q},\mathrm{D}) = P(T_1, T_2, \ldots T_n|D) = \prod_{i=1}^{n}((1-\lambda)P(T_i|C) + \lambda\, P(T_i|D)) \quad (1.13)$$

avec $\mathrm{P}(\mathrm{T_i}|\mathrm{C}) = \frac{\mathrm{df}(\mathrm{T_i})}{\sum_{\mathrm{T}} \mathrm{df}(\mathrm{T})}$ est la probabilité du terme Ti dans la collection de documents, d'où $df(T)$ est le nombre de documents dans lesquels T apparaît. λ est une constante.

1.5.5. Autres modèles de recherche d'information

Plusieurs modèles dérivés de ces modèles fondamentaux ont été proposés dans la littérature, à savoir :

Modèles ensemblistes : Le modèle booléen étendu, le modèle flou.

Modèles algébrique : Le modèle vectoriel généralisé, le modèle LSI (Latent Semantic Indexing Model) et le modèle connexionniste.

Modèles probabilistes : Le modèle basé sur les réseaux bayésiens.

1.6. Evaluation des systèmes de recherche d'information

L'évaluation des systèmes de recherche d'information (SRI) est au cœur des problématiques de la recherche d'information. L'efficience et l'efficacité sont les deux critères pour évaluer un SRI.

L'efficience: Plus le temps de réponse est court et plus l'espace occupé par le système est faible, le système est considéré meilleur.

L'efficacité: c'est la qualité des résultats renvoyés par le système, plus les réponses du système correspondent à celles que l'utilisateur espère, le système est considéré meilleur. Ce critère reste le plus important pour évaluer et comparer les SRI entre eux.

Le modèle d'évaluation utilisé aujourd'hui est généralement S'appuie sur celui développé dans le projet Cranfield **[Cle 66]**. Ce modèle repose sur trois éléments essentiels, à savoir la collection de documents sur laquelle les recherches sont effectuées, un ensemble de requêtes de test et une liste des documents de la collection pertinents pour chacune des requêtes. Ce modèle inclut également des mesures de performance associées au silence et au bruit documentaire, notions bien connues des documentalistes : le rappel et la précision.

L'objectif principal d'un système de recherche d'information est de retrouver tous les documents pertinents et rejeter tous les documents non pertinents. Pour évaluer cet objectif, un certain nombre de mesures standards sont proposées dans la littérature, les plus utilisés sont, la précision moyenne ou les courbes de rappel-précision **[Bac 10]**.

Nous allons décrire quelques mesures utilisés pour l'évaluation des SRI.

1.6.1. Rappel et précision

Le rappel et la précision sont deux mesures de base pour évaluer les performances des SRI.

Taux de rappel: c'est le ratio entre le nombre de documents pertinents retrouvés et le nombre total de documents pertinents dans la base. Si de nombreux documents intéressants n'apparaissent pas dans la liste retournée par le système on parle de silence, c'est-à-dire la proportion de documents pertinents non trouvés.

$$rappel = \frac{|P \cap R|}{P} \tag{1.14}$$

Taux de précision : c'est le nombre de documents pertinents retrouvés rapporté au nombre total de documents retrouvés. Tous les documents non pertinents retrouvés constituent du bruit.

$$prcision = \frac{|P \cap R|}{R} \quad (1.15)$$

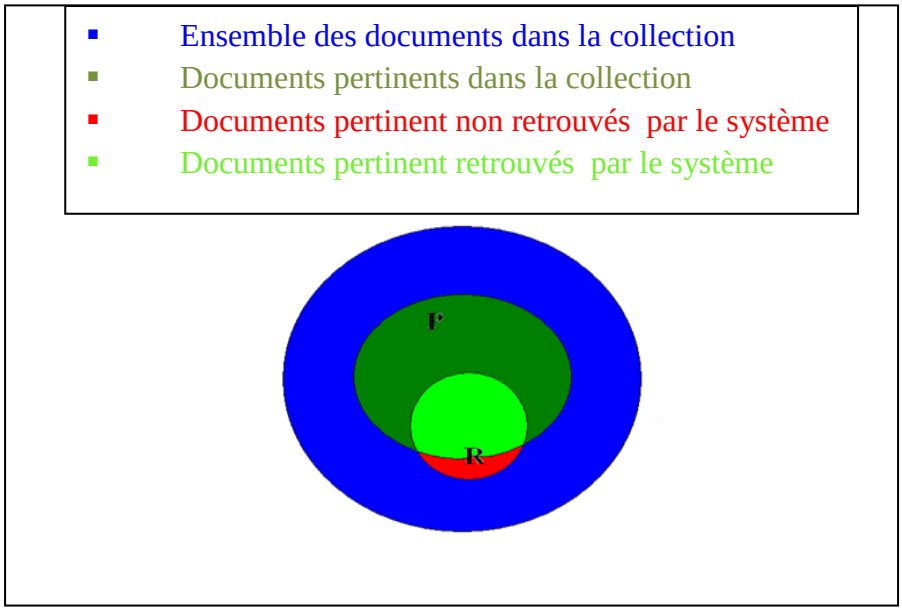

Figure 1. 2 : Précision et rappel

Courbe de rappel-précision

Les deux métriques ne sont pas indépendantes: quand l'une augmente, l'autre diminue. Il est facile d'avoir 100% de rappel: il suffit de donner toute les documents disponibles comme réponse à chaque requête. Cependant, la précision dans ce cas serait très basse. De même, il est possible d'augmenter la précision en donnant très peu de documents en réponse, mais le rappel se réduit. Il faut donc utiliser les deux métriques conjointement.

Le tableau ci-dessous illustre le calcul de précision et de rappel pour les 10 premiers documents renvoyés par un système pour une requête pour laquelle la collection contient 10 documents pertinents. La courbe de rappel-précision associée est tracée sur la figure 1.3 ci-dessous.

Pour évaluer la performance du système pour toutes les requêtes de la collection du test. Une précision moyenne à chaque niveau de rappel est calculée comme suit:

$$P(rp) = \sum_{i=1}^{N_q} \frac{P_i(rp)}{Nq} \quad (1.16)$$

	Requête 1		
Rang	Pertinent	Rappel	Précision
1	Oui	1/10 = 0.1	1/1 = 1.00
2	Oui	2/10 = 0.2	2/2 = 1.00
3	Non	2/10 = 0.2	2/3 = 0.66
4	Oui	3/10 = 0.3	3/4 = 0.75
5	Non	3/10 = 0.3	3/5 = 0.60
6	Oui	4/10 = 0.4	4/6 = 0.66
7	Non	4/10 = 0.4	4/7 = 0.57
8	Oui	5/10 = 0.5	5/8 = 0.62
9	Non	5/10 = 0.5	5/9 = 0.55
10	Oui	6/10 = 0.6	6/10 = 0.60

Tableau 1.3 : Calcul de précision et de rappel.

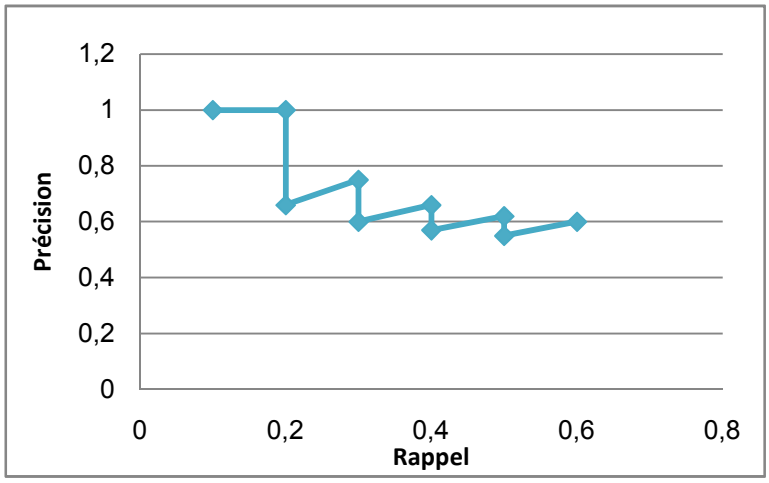

Figure 1. 3 : la courbe rappel-précision

Pour 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. Et pour calculer les valeurs de précision non obtenues à partir des valeurs de rappel, on utilise l'interpolation linéaire.

$$P_i^{'} = \max(p_i, p_j) \ \forall \ i < j \qquad (1.17)$$

Où $P_i^{'}$ est la précision interpolée au point de rappel *i*, et p_i est la vraie précision au point de rappel *i*.

1.6.2. Mesures alternatives

L'inconvénient principal des mesures de rappel et de précision est que ces deux mesures représentent des aspects différents de l'ensemble des documents retrouvés. L'utilisation d'une mesure unique combinant les propriétés de ces deux mesures serait peut être plus appropriée.

1.6.2.1. Mesure harmonique

Une mesure Harmonique H est une fonction qui combine les deux valeurs de précision et de rappel :

$$H(j) = \frac{2}{\frac{1}{R(j)}+\frac{1}{P(j)}} \quad (1.18)$$

La mesure Harmonique H est égale à 0 si aucun document pertinent n'est restitué et 1 si tous les documents restitués sont des documents pertinents.

La fonction H prend des valeurs élevée quand les valeurs de rappel et précision sont élevées. Ainsi, la mesure Harmonique garantie le compromis entre les deux mesures de rappel et précision.

1.6.2.2. Mesure d'évaluation « E »

La mesure d'évaluation E est proposée par Van Rijsbergen **[Rij 81]**, elle a pour but de permettre à l'utilisateur de spécifier laquelle des valeurs de précision et de rappel est plus intéressante. La fonction E est définie par:

$$\mathrm{E(j)} = \frac{1+\mathrm{b}^2}{\frac{\mathrm{b}^2}{\mathrm{R(j)}}+\frac{1}{\mathrm{P(j)}}} \quad (1.19)$$

Où b est un paramètre de l'utilisateur, lui permettant de spécifier l'importance du rappel et précision.

- Si b = 1, alors E(j) est le complément de la mesure Harmonique H(j).
- Plus les valeurs de b sont supérieures à 1: la précision est plus intéressante.
- Plus les valeurs de b sont inferieurs à 1: le rappel est plus intéressant.

1.6.3. Mesure NDCG

Une autre mesure fréquemment utilisée en RI est le NDCG (Normalized Discounted Cumulative Gain) **[Jär 02]**. Elle permet d'évaluer la capacité des SRI à renvoyer les documents pertinents en haut de la liste de résultats. Elle est définie de la façon suivante:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \quad (1.20)$$

$$\mathrm{DCG@k} = \sum_{\mathrm{i=1}}^{\mathrm{k}} \frac{2^{\mathrm{rel(i)}}-1}{\log\,(\mathrm{i}+1)}, \quad (1.21)$$

IDCG@) (également appelé DCG idéal) est la valeur maximale possible de DCG@k. La moyenne des valeurs $NDCG$ pour toutes les positions d'évaluation et pour toutes les requêtes est utilisée pour évaluer la performance du SRI.

1.6.4. Collection de référence(TREC)

Des campagnes d'évaluation ont été mises en place pour juger l'efficacité et évoluer la performance des systèmes de recherche d'information.

Le TREC est l'un des plus importants projets d'évaluation initié au tout début des années 90 par le NIST aux Etats-Unis, dans le but de proposer des moyens d'évaluation de systèmes documentaires sur des bases de documents conséquentes. Il est aujourd'hui co-sponsorisé par le NIST et l'ARPA (Defense Advanced Research Projects Agency), son objectif est d'encourager les travaux de recherche en informatique documentaire permettant l'accès à des bases volumineuses en fournissant :

- une base de test importante,
- des procédures d'évaluation uniformes,
- un forum pour les organismes intéressés par une comparaison de leurs résultats.

1.7. Conclusion

Comme nous l'avons vu dans ce chapitre, Les modèles de la recherche d'information présentés traitent les documents dans leur globalité et renvoi à l'utilisateur le document complet contenant l'information qu'il cherche, parfois l'utilisateur cherche une information précise, et il ne désire pas que cette information soit au milieu d'autres sujets. Avant l'apparition du XML, Plusieurs travaux ont été proposés concernant la granularité de l'information à renvoyer à l'utilisateur. Ces travaux ont cherché à découper un document textuel et renvoi à l'utilisateur des entités plus petites.

Le problème de granularité est résolu Avec l'apparition de nouveaux standards de présentation des documents tels que le XML. La granularité de l'information à renvoyer à l'utilisateur n'est plus le document entier mais un élément XML. Ces éléments sont caractérisés par d'autres informations que le contenu textuel, il s'agit de la structure.

Les SRI classiques ne tiennent pas en considération ces nouvelles caractéristiques. Nouveaux modèles cherchent à tirer parti de ces nouvelles caractéristiques, en combinant ces dernières avec l'information contenue dans l'élément lui-même, soit en améliorant les modèles de la RI classique ou en proposant de nouveaux modèles, spécialement, pour la RI-XML. Nous présentons dans le chapitre suivant les modèles et les techniques de la recherche d'information structurée.

CHAPITRE 2 : RECHERCHE D'INFORMATION STRUCTUREE

2.1. Introduction

La nature des collections de documents électroniques évolue. La représentation des documents passe de simples documents « texte plat » à des documents structurés ou semi-structurés contenant le contenue et la structure. Le contenue est le texte du document, ensemble de mots pour former des phrases. La structure est comment organiser logiquement un document (titre, section, paragraphe, etc.).

L'émergence du format XML (eXtensible Markup Language), conçus à l'origine pour faciliter l'échange et la standardisation des données, devenu comme format standard pour la représentation des documents structurés. Cette représentation soulève de nouvelles problématiques en Recherche d'Information (RI), tel que la pondération des termes, l'indexation utilisée ainsi que le modèle adopté pour la mise en correspondance de la requête et des éléments et les sources de pertinence utilisées pour calculer la pertinence des éléments XML, afin de renvoyer une liste triée des éléments à l'utilisateur. Dans ce chapitre nous présentons les différentes problématiques soulevées par la RI structurée, ainsi que les différentes solutions proposées dans la littérature qui utilisent d'autre sources de pertinences liée à la structure et au contenu des éléments pour répondre aux problématiques de la recherche d'information structurée.

2.2. Concepts de base des documents XML

2.2.1. Notion de structure (balise et élément)

La structure des documents XML 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 <balise>. Un élément est une unité syntaxique identifiée, délimitée par des balises de début et de fin , comme par exemple <balise> texte </balise>. Les éléments peuvent être imbriqués comme le montre l'exemple suivant :

```
<Article annee ="2011">
  <Titre> recherche d'information</Titre>
  <Section> <Titre> système de recherche d'information </Titre>
        <Paragraphe> un système de RI est… </Paragraphe> </Section>
  <Section> <Titre> recherche d'information structurée</Titre>
        <Paragraphe> xml.. </Paragraphe></Section>
</Article>
```

Figure 2. 1 : Exemple de document XML article.xml.

Les attributs des éléments sont intégrés à la balise de début en utilisant la syntaxe attribut = valeur. Par exemple, <balise attribut='valeur'> texte </balise>.

2.2.2. DTD (Document Type Defintion)

La DTD est un fichier décrivant la structure (grammaire) des documents, c'est un moyen pour vérifier la syntaxe d'un document. Il contient la liste des noms des éléments qui peuvent se produire dans un document, l'imbrication des éléments possible et les attributs de chaque type d'élément.

Contrairement à SGML, il n'est pas obligatoire d'associer une DTD à un document XML. Si les documents XML respectent un certain nombre de règles lexicales et syntaxiques, on parle des documents *bien formés* mais lorsque la grammaire d'un document est définie dans une DTD et que le document respecte cette DTD, on parle de document *valide*.

Par exemple, pour définir un élément ***élémentParent*** qui contient un autre élément ***élémentEnfant*** au moins une fois et contient un attribut obligatoire ***attr***, on écrit:

```
<!ELEMENT élémentParent (élémentEnfant)+>
<!ATTLIST élémentParent attr CDATA #REQUIRED>
<!ELEMENT élémentEnfant (#PCDATA)>
```

Figure 2. 2 : Exemple de DTD.

Malgré la facilité à écrire et à comprendre, les DTD présente quelques limitations :

- il ne s'agit pas d'un document XML
- il n'est pas possible de spécifier un type pour les données.

Le W3C a alors développé un autre format (***XML Schema***) pour but de remplacer les DTD. Ce format permet de définir des types de données, et par rapport aux DTD il offre la possibilité de spécifier un nombre déterminé d'occurrences d'un élément donné et permet la mise au point de grammaires modulaires en facilitant l'importation d'un schéma dans un autre.

2.2.3. DOM (Document Object Model)

DOM est une interface de programmation qui représente les documents XML en mémoire sous forme d'arbre. Grâce à ses fonctions, DOM permet de naviguer dans la structure des documents XML, ajouter, modifier et détruire des éléments avec un langage de programmation. Par exemple, si on prend le document *article.xml* de la figure 2.1 on trouvera l'arbre DOM correspondant sur la figure 2.3.

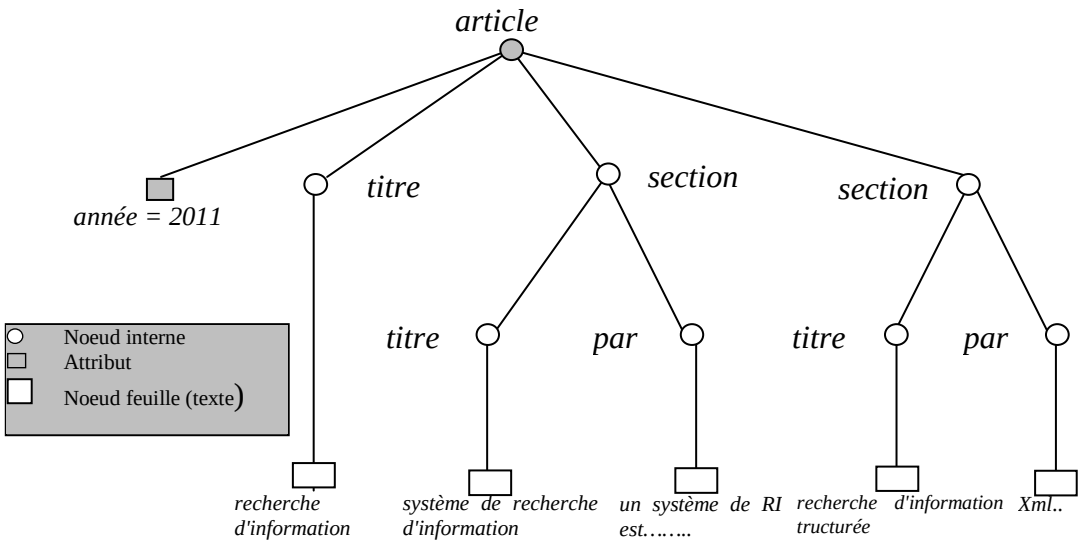

Figure 2. 3 : Exemple d'arbre DOM correspondant au document de la figure 2.1.

2.2.4. XPath

XPath est un langage permettant de sélectionner un ensemble d'éléments (sous arbres) d'un document XML répondant à certaines contraintes structurelles ou textuelles, Il possède une syntaxe simple et non ambigüe et implémente les types usuels (chaines, nombres, booléens, variables, fonctions).

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

- ***"Child ::"*** représente les enfants du nœud contextuel (abréviation : chaîne vide) ;
- ***"descendant-or-self::"*** représente les descendants du nœud contextuel ou ce nœud lui-même (abréviation : \/") ;
- ***"parent::"*** représente le parent du nœud contextuel (abréviation : \..") ;
- ***"ancestor::"*** représente les ancêtres du nœud contextuel ;
- …

2.3. Spécificité de la recherche d'information structurée

2.3.1. Granularité de l'information recherchée

Contrairement à la recherche d'information traditionnelle, Le but des SRI traitant les documents structurés est d'identifier des parties de documents les plus pertinentes à une requête donnée. Dans le cadre des documents XML, l'unité d'information est un nœud de

l'arbre du document (élément XML). La pertinence d'un élément vis-à-vis d'une requête est mesurée selon deux dimensions suivantes **[Chi 96]**:

Exhaustivité : une unité d'information est exhaustive à une requête si elle contient toutes les informations requises par la requête de l'utilisateur.

Spécificité: une unité d'information est spécifique si tout son contenu concerne la requête de l'utilisateur.

Dans des corpus 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.

2.3.2. Approches orientées données et Approches orientées documents

Les approches et les techniques proposées à la recherche dans les documents XML sont divisées en deux courants principaux **[Bou 08]**:

Approches orientées données : Les documents XML sont considérés comme des collections de données typées et relativement homogènes. Ces approches utilisent des techniques développées par la communauté de base de données.

Approches orientées documents : Les documents XML sont considérés comme des documents traditionnels (plat), c'est à dire que les balises sont utilisés uniquement pour décrire la structure logique des documents. Ces approches sont prises en charge par la communauté de la recherche d'information.

	SGBD	SRI
Besoin en information	Précis	Flou
Résultat	Exact	Inexact
Requête	SQL	Ensemble de mots clés
Modèle	Théorie des ensembles	Modèles de RI

Tableau 2.1 : Système de recherche d'information vs SGBD.

Alors que les deux communautés sont historiquement à l'origine de méthodes bien dissociées, la frontière entre les différentes approches pour la recherche dans des documents XML tend aujourd'hui à s'estomper. D'une manière générale, les solutions proposées par la communauté RI peuvent être utilisées comme "sur-couche" aux solutions orientées BD. Cette

sur-couche sert essentiellement à intégrer la notion de pertinence dans la recherche, en complétant les approches proposées par la communauté BD pour le stockage et l'interrogation des documents **[Sau 05]**.

Notre travail se positionne dans le domaine de la RI. Ainsi, les techniques de l'indexation, l'interrogation et le traitement des requêtes sont abordées sous l'angle des approches orientées documents.

2.4. Indexation des documents structurés

Dans les documents structurés le processus d'indexation est alors changé puisqu'il tient compte de la structure du document, ce qui engendre de nouvelles questions :

- Que doit-on indexer de la structure des documents ?
- Comment relier cette structure au contenu même du document ?
- Comment pondérer les termes d'indexation?

La manière la plus simple pour indexer les documents semi-structurés est de les considérer comme des documents plats et ne indexer que l'information textuelle, ce qui rend ce processus similaire à celui utilisé pour la RI traditionnelle. Cependant, aucune recherche sur la structure n'est plus possible, et la granularité renvoyée à l'utilisateur reste toujours le document dans son intégralité.

Selon **[Sau 05]**, un schéma d'indexation de documents XML devrait couvrir les aspects suivants :

- La reconstruction du document XML décomposé dans les structures de stockage;
- Le traitement des expressions de chemin sur la structure XML;
- L'accélération de la navigation dans des documents XML;
- Le traitement de prédicats vagues et précis sur le contenu du document XML;
- La recherche par mot-clés.

Selon le type de l'information manipulée (textuelle ou structurelle) on peut classer l'indexation des documents XML. Cette classification permet de mieux comprendre les différents enjeux soulevés par chaque type d'information.

2.4.1. Indexation de l'information textuelle

La présence de l'information structurelle dans les documents XML pose une nouvelle problématique d'indexation textuelle concernant la liaison entre les informations textuelles

(termes) et les informations structurelles (balises). Au niveau de l'indexation des documents, d'après **[Hub 10]**, il est possible de distinguer les approches qui:

- indexent tous les éléments XML **[SIG 05]** ;
- indexent ou uniquement les éléments feuilles **[Gev 06] ;**
- excluent certaines balises comme celles relatives à la mise en forme **[Kek 05] ;**
- indexent seulement certains types d'éléments (balises) issus de la DTD **[Gov 03] ;**
- indexent uniquement les éléments au-dessus d'une certaine taille **[Hat 05][Kam 04]** ;

Dans **[Bou 08]** l'auteur a résumé les approches ci-dessus on deux approches principales qui peuvent répondre au problématique de l'indexation des documents XML, à savoir :

L'agrégation du contenu des nœuds (l'approche d'indexation des sous-arbres imbriqués), et l'indexation de tous les contenus des nœuds séparément (l'approche d'indexation dite des unités disjointes).

2.4.1.1. Sous-arbres imbriqués

Les approches de ce premier groupe considèrent que le texte complet de chaque nœud de l'index est un document atomique **[Sig 04][Kam 04]** et propagent certains termes ou tous les termes d'un élément donné vers ses ancêtres. Un inconvénient majeur de ces approches est que l'index contient de nombreuses informations redondantes, par exemple, le texte se trouvant au niveau n de la structure logique du document XML est indexé n fois **[Lal 09]**.

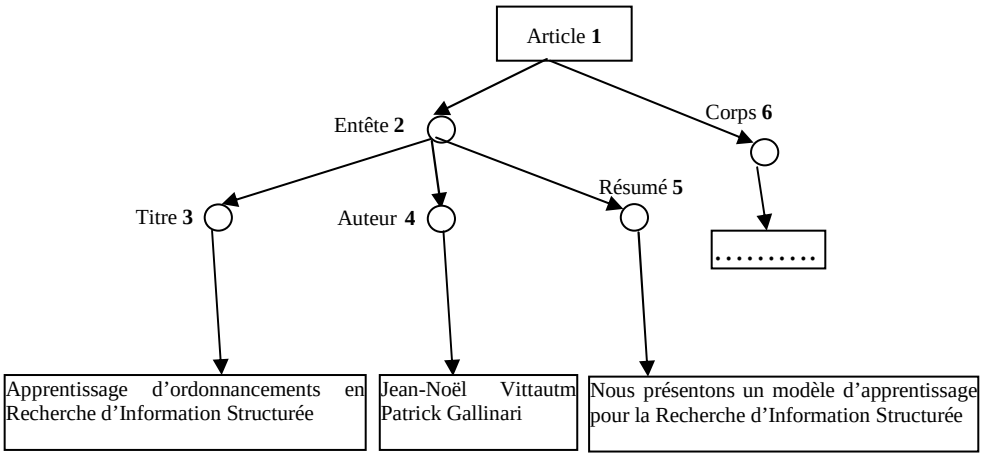

Figure 2. 4 : indexation des documents XML.

Les termes "apprentissage information " sont reliés aux nœuds /article/entête/titre, /article/entête, et /article.

2.4.1.2. Unités disjointes

Dans ces approches le document XML est décomposé en unités disjointes, donc le texte de chaque nœud de l'index est l'union d'une ou plus de ces parties disjointes **[Sau 06A] [Gev 06]**. Les termes des nœuds feuilles sont uniquement reliés au nœud parent qui les contient.

Si on reprend en exemple l'arbre de la figure 2.4 ci-dessus, les termes "apprentissage information " seront reliés uniquement au nœud /article/entête/titre.

2.4.2. Pondération des termes d'indexation

Les techniques de pondération des termes dans la RI classique cherche à rendre compte de l'importance de manière locale (tf) au sein du document et de manière globale (idf) au sein de la collection, s'ajoute en RI structurée l'importance du terme au niveau de l'élément qui le contient. Dans **[Wol 00]** et **[Gra 02]**, les auteurs proposent l'utilisation de l'ief (Inverse Element Frequency) et ils calculent un nouveau poids pour les termes en utilisant ief et la fréquence du terme dans chaque élément. D'autre calcul ief à travers des éléments du même type **[The 06].**

Dans **[Zar 04]**, le calcul du poids des termes est influencé par le contexte (l'unité d'indexation) dans lequel ils apparaissent. Ce calcul de poids est une adaptation de la formule tf-idf qu'on applique aux balises. Ainsi, les auteurs définissent le *tf-itdf (Term Frequency - Inverse Tag and Document Frequency)*, qui permet de calculer la force discriminatoire d'un terme *t* pour une balise b relative à un document d.

Des résultats expérimentaux produits par l'adaptation de BM25 en RI-XML **[Bro 08]** suggèrent qu'une meilleure performance est obtenue par l'estimation de *ief* pour tous les éléments de la collection au lieu pour chaque type d'élément **[Lal 09]**.

2.4.3. Indexation de l'information structurelle

L'indexation structurelle peut se faire selon des granularités variées, c'est à dire il n'est pas forcément d'utiliser toute l'information structurelle dans le processus d'indexation **[Sau 05]**. Dans la littérature on distingue trois types d'approches pour l'indexation de l'information structurelle : l'indexation basée sur des champs, l'indexation basée sur des chemins et l'indexation basée sur des arbres.

2.4.3.1. Indexation basée sur des champs

Il s'agit d'associer chaque terme du contenu au nom de la balise dans laquelle il apparaît pour permettre une recherche restreinte à certains champs. Les termes de l'index sont construits donc en combinant le nom du champ avec les termes du contenu.

Le tableau 2.2 montre un exemple d'indexation basée sur des champs pour le fichier de la figure 2.4.

Termes	Champ	
	Sans propagation	Avec propagation
Apprentissage	Titre	titre, entête, article
Patrick	auteur	auteur, entête, article
Recherche	titre, résumé	titre, entête, article, résumé
Modèle	Résumé	résumé, entête, article

Tableau 2.2 : Exemple d'indexation basé sur des champs.

2.4.3.2. Indexation basée sur des chemins

Contrairement à l'indexation basé sur des champs, cette technique remplace le nom de la balise par le chemin de l'élément basé sur XPath, c'est-à-dire d'indiquer le chemin d'accès logique plutôt que le nom de la balise comme illustre le tableau suivant:

Termes	Chemins
Apprentissage	article/entête/titre
patrick	Article/entête/auteur
Recherhche	article/entête/titre, article/entête/résumé
modèle	article/entête/résumé

Tableau 2.3 : Exemple d'indexation basé sur des chemins.

Cette indexation accélère le processus de recherche pour une information se situant dans plusieurs balises qui portent le même nom mais l'inconvénient majeur de cette méthode est qu'elle ne permet pas de décrire la relation de descendance des différents éléments du document XML.

2.4.3.3. Indexation basée sur des arbres

Contrairement à l'indexation par les chemins cette technique d'indexation permet d'assigner pour chaque nœud des valeurs de pré-ordre et post-ordre pour distinguer la relation ancêtre descendant et résoudre la difficulté posée par la technique basée sur des chemins. Plusieurs plans de numérotation des nœuds pour les documents XML ont été proposés dans la littérature. On peut par exemple citer

- l'index ANOR **[Lee 96]** ;
- l'approche EDGE **[Flo 99]**;
- l'approche BINARY **[Flo 99]**;
- l'index Xpath Accelerator [**Gru 02];**
- l'approche XFIRM **[Sau 05]**.

Ces modèles de numérotation doivent supporter deux opérations basiques **[Ali 07]** :

- **La décision** : Pour deux nœuds donnés, décider s'ils ont une relation spécifique, comme père-fils, ancêtre-descendant, frère-suivant, frère-précédent.
- **La reconstruction** : Pour un nœud donné, déterminer les identificateurs des nœuds de son voisinage comme, par exemple, le père, le frère suivant, le premier enfant, etc.

Termes	nœuds
Apprentissage	**3**
Patrick	**4**
Recherhche	**3, 5**
Modèle	**5**

Tableau 2.4 : Exemple d'indexation basé sur des arbres.

2.5. Modèles de recherche d'information structurée

La majorité des approches utilisées pour la RIS présentées dans la littérature sont généralement des adaptations des modèles traditionnels. Ces adaptations visent essentiellement à tenir compte de certaines spécificités des documents XML, à savoir les tailles très différentes que peuvent avoir les éléments XML, l'imbrication des éléments les uns dans les autres (non indépendance des éléments), ainsi que, les informations données par la structure elle-même.

Dans **[Bou 08]**, les différents modèles de la RIS sont classés selon deux types principaux:

- ***Approches par propagation des termes des documents*** : ces approches indexent des sous-arbres imbriqués et propagent les termes des nœuds feuilles dans l'arbre du document ;
- ***Approches par propagation des scores des éléments*** : ces approches indexent des unités disjointes et calculent des scores de pertinence au niveau des nœuds feuilles et ces scores sont ensuite propagés vers les nœuds internes.

2.5.1. Approche par propagation des termes des documents

2.5.1.1. Modèle vectoriel étendu

Le principe du modèle vectoriel appliqué à la RIS est de mesurer la similarité vectorielle de chaque élément à la requête. Les éléments sont représentés par des vecteurs de termes pondérés. Ces éléments sont renvoyés à l'utilisateur par ordre décroissant de pertinence.

Une des premières adaptations du modèle vectoriel est celle de Fuller et al **[Ful 93]**. La similarité d'un nœud vis-à-vis une requête est calculée par la formule suivante:

$$sim(q,n) = \propto (T) cosm(q,n) + \sum_{k=1}^{s} \frac{cosm(q,n_k)}{\beta^{k-1}} \qquad (2.1)$$

Où

- $\propto (T)$ est un facteur permettant de prendre en compte le type du nœud
- s est le nombre de nœuds enfants n_k de n,
- β est un paramètre permettant d'assurer que le nombre d'enfants n'introduit pas un biais dans la formule.

La fonction *cosm* est définie de la façon suivante :

$$cosm(q,n) = \sum_{i=1}^{T} \frac{w_i^q * w_i^n}{|n|} \qquad (2.2)$$

Où w_i^q , w_i^n représentent respectivement le poids du terme ti dans la requête q et dans le nœud n, et |n| le nombre de termes dans le nœud n.

La pertinence d'un nœud peut ainsi être calculée à part, puis combinée avec la pertinence des nœuds descendants. Le modèle peut être généralisé en permettant le traitement des requêtes orientées contenu et structure. L'idée de base est là encore d'appliquer le modèle récursivement à chaque sous-arbre de la hiérarchie pour ensuite effectuer un agrégat des scores.

Le modèle JuruXML **[Mas 02][Mas 03]** est une autre adaptation du modèle vectoriel, consiste à intégrer la structure des documents dans la mesure de similarité. Propose d'indexer les éléments selon leur type (un index par type d'élément) et d'appliquer ensuite le modèle vectoriel pour la pondération des éléments.

Les requêtes orientées contenu sont évaluées sur chacun des index et les résultats, qui ont été normalisés, sont ensuite fusionnés afin de fournir une liste unique de résultats à l'utilisateur.

Pour les requêtes structurées sont décomposées en ensemble de conditions de la forme (chemin, terme). Ensuite, une correspondance vague entre les chemins est calculée.

- Soit C_i^q la condition du chemin pour le terme ti
- Soit C_i^e le XPath du terme ti dans l'élément e.

La fonction de similarité entre les deux chemins est calculée alors:

$$cr\left(c_i^q, c_i^e\right) = \begin{cases} \frac{1+|c_i^q|}{1+|c_i^e|} = 1 \; si \; c_i^q \; est \; une \; sous - sequence \; de \; c_i^e \\ 0 \; sinon \end{cases} \quad (2.3)$$

Exemple :

C_i^q: (Article/paragraph, terme) C_i^e : (Article/section/chapitre/paragraphe, terme).

$cr\left(c_i^q, c_i^e\right) = \frac{3}{5} = 0.6.$

Ainsi le score final d'un élément est donné par

$$RSV(e,q) = \frac{\sum_{(t,c_i^q)\in q} \sum_{(t,c_i^e)\in e} w_q(t) * w_e(t) * cr(c_i^q, c_i^e)}{|q| * |e|} \quad (2.4)$$

Où $w_q(t)\; et \; w_e(t)$sont les poids du terme t dans q et e, $|q| \; et \; |e|$ sont les nombres de termes dans q et e.

Cette dernière approche, évaluée dans le cadre de la campagne INEX 2004 permet d'obtenir de bons résultats par rapport à l'ensemble des participants **[Bou 08]**.

2.5.1.2. Modèle de langue

Dans **[Kam 04]** les auteurs proposent une approche en basant sur les modèles de langue inspirée du modèle langue **[Pon 98].**

Concéderons une requête q = (t1,t2,…,tn) contient n termes ti , un élément *e* et le modèle de langue correspondant *Me,* les éléments sont triés par rapport à la probabilité que le modèle de langage de l' élément génère la requête. Ceci revient à estimer la probabilité *P(e,q)*:

$$P(e|q) = P(e)P(q|M_e) \quad (2.5)$$

Où $P(e)$ est la probabilité a priori de pertinence de l'élément e et $P(q|M_e)$ est la probabilité que la requête q est générés par le modèle de langage Me. En utilisant le lissage de Jelinek-Merce. La probabilité $P(q|M_e)$ est ainsi calculée de la façon suivante:

$$P(t1, t2, \dots, tn|Me) = \prod_{i=1}^{n} \lambda P\,(ti\,|e) + \,(1\, -\, \lambda)P\,(ti\,|C) \quad (2.6)$$

Où $P\,(ti\,|e)$ est la probabilité d'observer le terme t_i dans l'élément e, $P\,(t_i\,|C)$ est la probabilité d'observer le terme t dans la collection C et λ est un paramètre de lissage.

Le calcul des probabilités peut être réduit à la formule de calcul des scores ci-dessous, pour un élément e et une requête *q(t1,t2, ...,tn).*

$$S(e, t_1, \dots, t_n) = \beta.\log(\textstyle\sum_t tf(t,e)) + \sum_{i=1}^{n} log\left(1 + \frac{\lambda.tf(t_i,e).(\sum_t df(t))}{(1-\lambda).df(t_i).(\sum_t tf(t_i,e))}\right) \quad (2.7)$$

où

- tf(t, e) : la fréquence du terme t dans l'élément e,
- df(t) : le nombre d'éléments contenant t,
- λ : le poids donné au modèle de langage de l'élément en lissant avec le modèle de la collection,
- β : le paramètre servant à combler le fossé entre la taille de l'élément moyen et la taille de l'élément moyen pertinent.

Un autre critère de pertinence a été utilisé dans ce model est celui de la taille des éléments XML, une préférence en faveur des éléments de grande taille est intégrée comme suit, $P(e) = \frac{length(e)}{\sum_{e\prime} length(e')}$, tel que le dominateur est la somme de toutes les tailles des éléments de l'index.

Autre estimation de la probabilité a priori ont été proposés dans la littérature on peut citer les travaux de **[Hua 07]** qui utilisent des probabilités a priori des éléments en fonction de

leur position et de leur profondeur dans le document ou les travaux de **[Ash 07]** qui calcule la probabilité a priori on incluant le nombre de changements de sujet **[Lal 09]**.

2.5.1.3. Modèle XIVIR

Le modèle XIVIR **[Aou 09]** (XML Information retrieval based on VIRtual links) permet la recherche d'information dans des documents XML par la structure, par le contenu et par la combinaison des deux.

Recherche par le contenu : L'auteur adopte deux manières pour propager le texte se situant dans un nœud feuille à ces ancêtres, la première se base sur la profondeur, la deuxième se base sur la profondeur et la largeur (exprimé en fonction du nombre de fils) d'un document XML. La propagation de texte par profondeur consiste à traduire le contenu de chaque nœud feuille par un ensemble de termes pondérés, ceux-ci seront propagés vers les ancêtres de ce nœud tout en diminuant leurs poids en fonction de la distance parcourue au moment de la propagation. La propagation de texte basé sur la profondeur et la largeur se fait de la même manière que la première en ajoutant le nombre de fis des ancêtres à qui le terme t appartient.

Recherche par la structure : Dans un document XML, deux nœuds appartenant au même chemin ont une relation directe (parent/fils-direct) ou indirecte (parent/descendant). Afin de refléter l'importance de la relation entre les nœuds A et B, un poids est calculé pour chaque chemin. Si la relation est directe, le poids est égal à 1, sinon, le poids w est calculé comme suit:

$$w = exp\left(\lambda\left(1 - d(A,B)\right)\right) \tag{2.8}$$

Où $d(A,B)$ est la distance entre les deux nœuds A et B,et λ est un coefficient d'atténuation.

Pour la recherche par structure, le score (RSV_s) entre la requête q et le document d est calculé comme suit :

$$RSV_s(E_q, E_d) = \sum_{A_q \xrightarrow{w_q} B_q \in E_q \equiv A_d \xrightarrow{w_d} B_d \in E_d} w_q * w_d \tag{2.9}$$

Où E_q , E_d sont les ensembles de tous les chemins pondérés de la requête q et le document *d*. Soient A_q l'élément A dans la requête q et A_d est l'élément A dans le document d, $A_q = A_d$ signifie que A_q est l'équivalent de A_d.

Par exemple, $chapter \xrightarrow{0.37} p \equiv chapter \xrightarrow{1} p[2]$ sur la figure ci-dessous

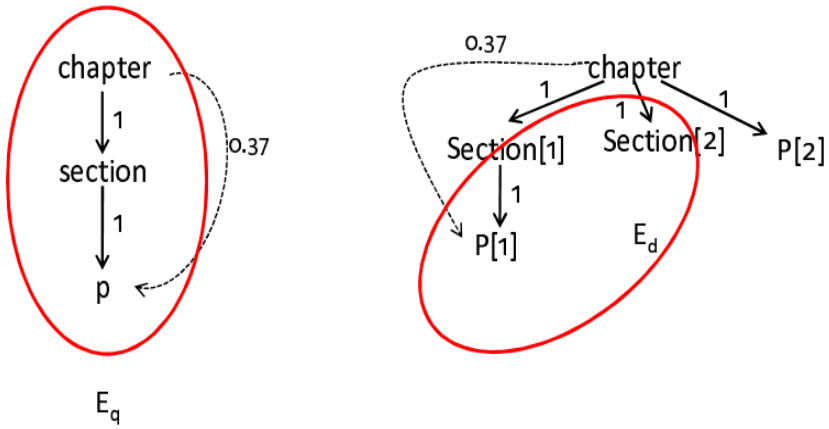

Figure 2. 5 : Exemple de recherche par structure avec le système XIVIR [Aou 09].

Le score selon la structure entre la requête et le document est :

$$RSV_s(E_q, E_d) = 2 + 0.37 * 0.37 = 2.14$$

Recherche par le contenu et la structure: Le score final de l'élément XMLvis à vis de la requête sera calculé en combinant les deux scores calculé selon les deux critères contenu et structure.

2.5.2. Approche par propagation des scores des éléments

2.5.2.1. Modèle GPX

Le Modèle GPX de Shlomo Geva (Queensland University of Technology) **[Gev 06]** considère que les nœuds feuilles contiennent l'essentiel de l'information textuelle. Un score est calculé pour ces nœuds, puis les valeurs sont propagées vers le haut jusqu'à la racine.

Le score de pertinence des feuilles est calculé avec la formule suivante

$$RSV(q, nf) = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \quad (2.10)$$

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

Le score des éléments feuilles est alors utilisé pour calculer celui de leurs ancêtres. Il s'agit de propager la pertinence des éléments en utilisant la formule suivante.

$$RSV(q,n) = D(n)\sum_{l=1}^{n} RSV(q,nc_l) \quad (2.11)$$

Avec *n* le nombre d'enfants jugés pertinents de l'élément.

$$D(n) = \begin{cases} 0.49\ si\ n = 1 \\ 0.99\ sinon \end{cases} \quad (2.12)$$

$RSV(q,nc_l)$ est le score de pertinence de lème élément fils.

Ce système a obtenu des bons résultats dans la campagne d'évaluation INEX 2005 **[Gev 06]**.

2.5.2.2. Modèle XFIRM

Le principe du modèle XFIRM (XML Flexible Information Retrieval Model) **[Sau 05]** est de calculer des valeurs de pertinence pour les différents nœuds feuilles (unité disjointe c'est à dire les nœuds contenant du texte). Par la suite Ces valeurs sont propagées vers les nœuds ancêtres.

Ce modèle permet de traiter les deux types de requêtes orientées contenu (CO) et orientées contenu et structure(CAS). Le traitement des requêtes orientées contenu est effectué comme suit :

Une première étape consiste à évaluer la similarité des nœuds feuilles de l'index vis-à-vis de la requête (calcul des poids des nœuds feuilles) et une seconde étape consiste à rechercher les sous-arbres pertinents en effectuant la propagation des poids des feuilles dans l'arbre du document.

***Calcul du score des nœuds feuilles* :** Les scores des nœuds feuilles sont calculés grâce à la fonction de similarité *rsv(q, nf).*

$$rsv(q,nf) = \sum_{i=1}^{T} w_i^q w_i^{nf} \quad (2.13)$$

Avec $w_i^q = tf_i^q \ \ et\ w_i^{nf} = tf_i^{nf} * ief_i * \ idf_i$

Où:

- $w_i^q\ et\ w_i^{nf}$ sont respectivement le poids du terme i dans la requête q et le nœud feuille nf.
- $tf_i^q et\ tf_i^{nf}$ sont respectivement la fréquence du terme i dans la requête q et dans le noeud feuille nf.

- $idf_i = \log(\frac{|D|}{|d_i|})$ permet d'évaluer l'importance du terme i dans la collection de documents, avec |D| le nombre total de documents de la collection et |di| est le nombre de documents contenant i.
- $ief = \log(\frac{|NF|}{|nf_i|})$ permet d'´evaluer l'importance du terme i dans la collection de nœuds feuilles, où |*NF*| est le nombre total de nœuds feuilles de la collection, et |*nfi*| est le nombre de nœuds feuilles contenant i.

Propagation de la pertinence des nœuds feuilles : Il semble intuitif que plus grande est la distance entre un nœud et son ancêtre, moins il contribue à sa pertinence. Cette intuition est modélisée par l'utilisation dans la fonction de propagation du paramètre dist(n, nfk), qui représente la distance entre le nœud n et un de ses nœuds feuilles nfk dans l'arbre du document, c'est-`a-dire le nombre d'arcs séparant les deux nœuds.

Il paraît aussi intuitif que plus un nœud possède de nœuds feuilles pertinents, plus il est pertinent. Il faut introduire alors le paramètre $\left|F_n^p\right|$, qui est le nombre de nœuds feuilles descendants de n ayant un score positif, dans la fonction de propagation.

La pertinence pn d'un nœud peut être calculée en intégrant la pertinence du document selon la formule suivante:

$$p_n = \rho * \left|F_n^p\right| . \sum_{nf_k \in F_n} \alpha^{dist(n,nf_k)-1} * RSV(q, nf_k) + (1-\rho) * p_{racine} \qquad (2.14)$$

$\rho \in [0..1]$ est un paramètre, fixé manuellement après plusieurs expérimentations, servant de pivot et permettant d'ajuster l'importance de la pertinence du nœud racine. Les nœuds sont ensuite renvoyés à l'utilisateur par ordre décroissant de pertinence.

Ce modèle a montré de bonnes performances au sein de la campagne d'évaluation INEX.

2.6. Evaluation des systèmes de recherche d'information structurée

L'évaluation de la performance d'un SRIS, se base généralement sur les mesures de rappel et de précision (étudié à la section 1.6.1). La compagne INEX (INitiative for the Evaluation of XML retrieval) est aujourd'hui la seule compagne pour l'évaluation de la RIS dans les documents XML.

2.6.1. Compagne d'évaluation

INEX est définie comme une initiative pour évaluer les systèmes de recherche dans les documents XML. Cette initiative a été lancée en 2002 et a subi différentes évolutions depuis son lancement. INEX fournit des collections de tests constituées d'un corpus de documents XML, de requêtes orientées vers la recherche XML et des réponses associées (jugements de pertinences).

2.6.1.1. Collection de référence

La compagne d'évaluation INEX fournit une collection de documents balisés au format XML. Jusqu'en 2004, INEX a utilisé une collection composée de 12107 articles balisé au format XML provenant de 18 magazines ou revues différentes de la "IEEE computer Society" publiés de 1995 à 2004, soit 494 Mo et 8 million d'éléments. En 2005, un total de 4712 nouveaux articles de la période 2002-2004 ont été ajoutés à la collection qui donne un total de 16819 articles, 764 Mo et 11 million d'éléments (cette dernière est la collection utilisée pour évaluer notre approche).

En 2006, la collection s'est renouvelée elle est composée de plus de 659,388 articles de l'encyclopédie Wikipedia, plus de 60 Go (4.6 Go sans images) et 52 million d'éléments [**Den 06]**.

2.6.1.2 Requêtes

Les requêtes de la collection sont crées par les différents participants et elles représentent les besoins en information des utilisateurs. Deux types de besoins en matières de l'information sont proposés dans INEX'2004, à savoir :

- contenu seulement (type de requête CO: Content Only).
- des requêtes comportant des informations sur le contenu et la structure (type de requête CAS, Content And Structure).

En 2005 les deux formes de requêtes sont regroupées et sont devenus CO+S (Content-Only + Structure) et les sujets de requête comportent aujourd'hui deux formulations différentes :

- Le champ **<*inex_topic*>** : contient deux attributs, l'un indique le numéro de la requête (***topic_id***) et l'autre le type de la requête (***query_type***).

- le titre (champ **<*title*>**) exprime le besoin en information de l'utilisateur sous. forme de simples mots-clés:

-le titre structuré (champ **<*castitle*>**) exprime le même besoin en information de l'utilisateur mais en y ajoutant des précisions sur la structure des documents. Le figure 2.6 illustre un exemple de ce type CO+S.

<**inex_topic topic_id**="231" **query_type**="CO+S">
<**title**>markov chains in graph related algorithms</**title**>
<**castitle**>//article//sec[about(.,+"markov chains" +algorithm+graphs)] </castitle>
<**description**>Retrieve information about the use of markov chains in graph theory and in graphs-related algorithms.
</**description**>
<**narrative**>**I** have just finished my Msc. in mathematics, in the field of stochastic processes. My research was in a subject related to Markov chains. My aim is to find possible implementations of my knowledge in current research. I'm mainly interested in applications in graph theory, that is, algorithms related to graphs that use the theory of markov chains. I'm interested in at least a short specification of the nature of implementation (e.g. what is the exact theory used, and to which purpose), hence the relevant elements should be sections, paragraphs or even abstracts of documents, but in any case, should be part of the content of the document (as opposed to, say, vt, or bib).</**narrative**>
</**inex_topic**>

Figure 2. 6 : Exemple de requête de type CO+S issue de la compagne d'évaluation INEX'2005.

2.6.1.3 Jugements de pertinence

Un système de recherche d'information est efficace s'il récupère tous et seulement les documents pertinents pour une requête donnée. Pour savoir les documents pertinents parmi les non pertinent d'une requête, une phase permet d'obtenir des jugements de pertinence où pour chaque élément du document ou le document en totalité est jugé à la main (par les participants) pour chaque requête.

En 2002, INEX défini deux dimensions pour la mesure de pertinence, basée sur le degré de pertinence et la couverture des éléments. En INEX 2004 ces deux dimensions ont été remplacées par deux autres dimensions **[Chi 96]:** la première dimension mesure l'exhaustivité et la seconde dimension mesure la spécificité d'un élément pour une requête donnée.

La notion d'exhaustivité décrit jusqu'à quel point l'élément discute du sujet de la requête. On distingue pour ce critère d'évaluation quatre niveaux d'exhaustivité :

- Pas exhaustif (0): l'élément ne traite pas du tout le sujet de la requête
- Marginalement exhaustif (1): l'élément traite marginalement le sujet de la requête.
- Assez exhaustif (2): L'élément aborde le sujet de la requête, mais pas de manière exhaustive (pas tous les aspects du sujet).
- Très exhaustif (3): L'élément aborde le sujet de la requête de manière exhaustive (tout les aspects du sujet).

La notion de spécificité décrit jusqu'à quel point l'élément renvoyé par le système traite de toute l'information recherchée. On distingue aussi pour ce critère d'évaluation quatre niveaux de spécificité :

- Pas spécifique : le sujet de la requête n'est pas un thème de l'élément
- Marginalement spécifique : le sujet de la requête est un thème mineur de l'élément
- Assez spécifique : le sujet de la requête est un thème majeur de l'élément
- Très spécifique : le sujet de la requête est le seul thème de l'élément

Comme les deux dimensions ne sont pas tout à fait orthogonales (par exemple, lorsque l'élément n'est pas exhaustif, il ne peut pas être spécifique, et inversement). Alors Il y a 10 valeurs possibles et pas 16. Le degré de pertinence d'un élément jugé par les participants est donnée par La combinaison des deux dimensions, soit la paire (e, s), avec (e, s) $\in$ ES et ES = {(0, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)}.

En 2005 les mesures ont été changées. L'exhaustivité est mesurée en utilisant 4 niveaux : très exhaustive (e = 2), peu exhaustive (e = 1), pas exhaustive (e = 0) et "trop petite" (e =?). La spécificité qui traduit le degré avec lequel un élément traite de toute l'information recherchée, spec(ei) dans [0,1], elle est calculé comme le ratio entre le nombre de caractères sélectionnés (*highlighted*) contenus dans l'élément $rsize(e_i)$ et le nombre total de caractères contenus dans l'élément $rsize(e_i)$.

$$spec(e_i) = \frac{rsize(e_i)}{size(e_i)} \quad (2.15)$$

La figure 2.7 montre un exemple de jugement de pertinence pour les éléments du fichier « *ex/2003/x1055.xml* » par rapport à la requête numéro 202 de la collection INEX 2005.

```
<file collection="ieee" name="ex/2003/x1055">
  <passage start="/article[1]/bdy[1]/sec[4]/ss1[3]/st[1]"
 end="/article[1]/bdy[1]/sec[4]/ss1[4]/p[13]/text()[5].229" size="11651"/>
  <element path="/article[1]/bdy[1]/sec[4]" exhaustivity="2" size="20392" rsize="11651"/>
  <element path="/article[1]/bdy[1]" exhaustivity="2" size="36339" rsize="11651"/>
  <element path="/article[1]" exhaustivity="2" size="41659" rsize="11651"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]" exhaustivity="2" size="3691" rsize="3691"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/st[1]" exhaustivity="?" size="55" rsize="55"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/p[1]" exhaustivity="?" size="666" rsize="666"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/p[2]" exhaustivity="?" size="382" rsize="382"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/p[3]" exhaustivity="?" size="487" rsize="487"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/p[2]/it[1]" exhaustivity="?" size="3" rsize="3"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[4]" exhaustivity="1" size="7961" rsize="7961"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[4]/st[1]" exhaustivity="?" size="16" rsize="16"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[4]/p[1]" exhaustivity="?" size="295" rsize="295"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/fig[1]" exhaustivity="?" size="64" rsize="64"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/fig[1]/art[1]" exhaustivity="?" size="1" rsize="1"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/fig[1]/no[1]" exhaustivity="?" size="2" rsize="2"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[3]/fig[1]/fgc[1]" exhaustivity="?" size="63" rsize="63"/>
```

Figure 2. 7 : Extrait d'un fichier de jugement de pertinence explique l'exhaustivité et la spécificité des éléments XML.

A partir de 2006, la dimension exhaustivité a été supprimée, et la spécificité reste la seule dimension pour mesurer le degré de pertinence des éléments XML **[Lal 06]**.

2.6.2. Mesures d'évaluation

Plusieurs mesures d'évaluation ont été proposées. Ces mesures utilisent une valeur unique combinant l'exhaustivité et la spécificité. Pour calculer cette valeur, plusieurs fonctions d'agrégation ont été définies:

L'agrégation stricte : pour évaluer si un SRI est capable de retrouver des éléments très spécifiques et très exhaustifs

$$f_{strict}(s,e) = \begin{cases} 1 \; si \; s = 1 \; et \; e = 2 \\ 0 \; sinon \end{cases} \quad (2.16)$$

L'agrégation généralisée: la fonction d'agrégation est généralisée est calculée par le simple produit entre s et e :

$$f_{généralisée}(s, e) = s * e \tag{2.17}$$

Les métriques utilisées en INEX 2005 sont le *gain cumulé étendu normalisé* (nxCG) et la courbe *d'effort-précision* par rapport au *gain-rappel*.

2.6.2.1. Gain cumulé étendu (XCG)

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

$$xCG(i) = \sum_{j=1}^{i} xG(j) \tag{2.18}$$

Où $xG(j)$ est le score de l'élément classé par le système à la position j.

Pour obtenir le gain cumulé étendu normalisé, on compare ce gain avec le gain cumulé idéale (le gain que normalement le système doit atteindre).

$$nxCG(i) = \frac{xCG(i)}{xCI(i)} \tag{2.19}$$

où $xCI(i)$ est le gain cumulé idéale.

Par analogie au gain cumulé, on définit l'effort-précision $\mathrm{ep(r)}$:

$$ep(r) = \frac{e_{ideal}}{e_{run}} \tag{2.20}$$

Où e_{ideal} est le rang pour lequel le gain cumulé est atteint par la courbe idéale et e_{run} est le rang pour lequel le gain cumulé est atteint par le système. Si cette valeur égale à 1 (c'est-à-dire une performance idéale du système), 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 :

$$gr(i) = \frac{xCG(i)}{xCI(n)} \tag{2.21}$$

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 moyenner les valeurs d'effort-précision pour chaque rang auquel un élément pertinent est renvoyé **[Sau 06B]**.

2.6.2.2. HiXEVAL

Une étude publiée dans **[Kas 07]** montre que la méthode XCG est insuffisante. Ceci s'est traduit par l'adaptation de nouvelles mesures qui ne sont autres que les définitions traditionnelle de la précision et du rappel. Lesdites mesures ont été initiées par Pehcevski et Thom **[Peh 06]**, c'est-à-dire HiXEVAL, et sont finalisées par **[Kam 08]**. Ces mesures ont été considérées comme des mesures officielles au niveau de INEX depuis 2007 **[Lal 09]**.

L'objectif d'un SRI dans les documents XML consiste à restituer les éléments qui contiennent autant d'informations pertinentes que possible et peu d'informations non pertinentes que possible. La précision et le rappel peut être modifiée par les fonctions suivantes :

$$précision = \frac{quantité\ d'information\ pertinenete\ restituée}{quatité\ total\ de\ l'information\ restituée} \tag{2.22}$$

$$rappel = \frac{quantité\ d'information\ pertinenete\ restituée}{quatité\ total\ de\ l'information\ pertinente} \tag{2.23}$$

Soit ei l'élément classé a la position i, rsize(ei) la taille de l'information pertinente dans ei pour une requête donnée. size(ei) représente la taille de ei (nombre de caractère), et Trel la taille de l'information pertinente dans la collection.

La précision peut être calculée comme suit:

$$P@r = \frac{\sum_{i=1}^{r} rsize(ei)}{\sum_{i=1}^{r} size(ei)} \tag{2.24}$$

Cette définition assure que pour atteindre une valeur de précision élevée, de rang r, les éléments trouvés jusqu'à r doivent contenir le minimum d'informations non pertinentes (maximiser rsize pour chaque ei).

Le rappel quant à lui correspond à la formule suivante :

$$R@r = \frac{1}{Trel} . \sum_{i=1}^{r} rsize(ei) \tag{2.25}$$

Cette définition assure que pour atteindre une valeur de rappel élevée, de rang r, les éléments trouvés jusqu'à r doivent contenir le maximum d'informations pertinentes.

2.7. Conclusion

Dans ce chapitre, nous avons présenté les principaux éléments de la recherche d'information structurée. En premier lieu, nous avons présenté un aperçu sur les documents structurés. Ainsi, nous avons présenté les problématiques et les enjeux soulevés par la RI structurée relatifs à la présentation de l'information structurel dans ces documents.

Différentes approches et modèles d'indexation ont été proposées dans la littérature, ainsi que des modèles pour l'interrogation des corpus de documents XML ont été développé. Ces modèles utilisent des fonctions de score pour ordonner les éléments retrouvés avant les envoyer à l'utilisateur, chaque fonction de score contient une ou plusieurs sources de pertinence. Plusieurs paramètres tel que λ, α, ρ …etc. ont été utilisés et fixés manuellement d'une manière empirique pour définir le poids de chaque source de pertinence. On peut signaler que l'utilisation de plusieurs paramètres influence l'efficience du modèle, aussi bien que certaines sources ne sont pas très utiles et leur utilisation peut affecter l'efficacité du modèle. La difficulté est alors de trouver les meilleures sources de pertinence et de savoir les combiner pour que le modèle soit efficient et efficace et améliore la performance de la recherche d'information structurée.

L'utilisation des techniques d'apprentissage dans le contexte de la RI et la RI structurée peut résoudre ce problème. L'objectif ici est d'apprendre les paramètres des fonctions d'ordonnancement on se base sur des donnée d'apprentissage afin de construire un model efficace de recherche. Ce model sera utilisé ensuite pour calculer la pertinence des éléments pour des nouvelles requêtes.

Le chapitre suivant sera consacré à l'étude des différents travaux de recherche menés pour l'application de l'apprentissage d'ordonnancement en RI et en RIS.

CHAPITRE 3 : APPRENTISSAGE D'ORDONNANCEMENT EN RECHERCHE D'INFORMATION

3.1. Introduction

L'objectif principal de l'apprentissage automatique est d'apprendre une fonction de prédiction, qui va induire, pour chaque entrée, une valeur désirée correspondante en sortie.

Ces dernières années, les techniques d'apprentissage automatique ont été utilisés pour entraîner le modèle de Ranking, et un nouveau domaine de recherche appelé apprentissage d'ordonnancement ou apprentissage des fonctions d'ordonnancement, en anglais « Learning to Rank », a progressivement émergé. Particulièrement au cours des dernières années, l'apprentissage d'ordonnancement a connu un succès réel et s'est rapidement développé pour devenir l'un des domaines de recherche les plus actifs et une thématique phare dans le domaine de la recherche d'information.

Dans L'apprentissage d'ordonnancement une fonction de score est apprise à partir d'un ensemble d'apprentissage étiqueté. Cette fonction prend en entrée les éléments à ordonner et en sortie le score de ces éléments. L'ordre est ensuite prédit en triant les éléments selon les scores croissants ou décroissants. Contrairement aux cas de la classification ou de la régression, la valeur du score prédit pour une entrée donnée n'est pas importante mais seuls les scores relatifs entre les éléments sont importants. Pour effectuer l'apprentissage de fonctions d'ordonnancement, il nécessite donc de définir de nouveaux critères d'erreurs, des algorithmes pour optimiser l'erreur, et une théorie permettant de montrer que la fonction apprise sera performante sur les nouvelles données.

Dans ce chapitre, nous présentons l'apprentissage automatique, en plus particulier l'apprentissage automatique supervisé où nous avons détaillé les quatre éléments clé de ce type. Par la suite, nous expliquons, le cadre formel de l'apprentissage supervisé des fonctions d'ordonnancement en détaillant les principales approches telles que : l'approche pointwise, pairwise et listwise. Nous évoquons enfin les méthodes adaptés à la RIS.

3.2. Apprentissage automatique

L'apprentissage automatique (de l'anglais machine Learning et aussi appelé apprentissage statistique ou apprentissage machine), est un des champs d'étude de l'intelligence artificielle, et la discipline scientifique concernée par le développement des algorithmes qui permettent à une machine d'évoluer et améliorer sa performance à partir d'un ensemble de données d'apprentissage. L'algorithme d'apprentissage est optimisé alors à partir

d'une base d'apprentissage (ensemble d'observations) et qui essaye d'en extraire automatiquement les régularités statistiques.

Les algorithmes d'apprentissages peuvent se catégoriser selon le mode d'apprentissage qu'ils emploient :

Apprentissage non-supervisé : Apprentissage par lequel un programme peut, par inférence, découvrir seul de nouveaux concepts, trouver de nouvelles lois, de nouveaux faits (*par exemple clusttering*).

Apprentissage supervisé : Apprentissage par lequel un programme découvre de nouvelles lois capables de (classer/prédire) correctement les exemples de l'ensemble d'apprentissage mais aussi des exemples nouveaux ne faisant pas partie de cet ensemble.

L'apprentissage supervisé est l'objet principal de ce mémoire.

3.2.1. Apprentissage automatique supervisé

Dans l'apprentissage supervisé, l'objectif et d'apprendre une relation probabiliste $p(x,y)$ entre les exemple de l'espace d'entrée $x \in X$ et les sorties désirées de l'espace de sortie $y \in Y$. la modélisation de cet apprentissage est donc portée sur les éléments clés suivants.

1. ***L'espace d'entrée,*** qui contient Les objets (observations) concerné par l'apprentissage. Généralement les objets sont représentés par des vecteurs de caractéristiques, extraits selon des applications différentes. Le vecteur de caractéristiques est une représentation numérique dans un espace vectoriel X, typiquement $X \subset R^d$ pour d fixé.

2. ***L'espace de sortie,*** qui contient le résultat d'apprentissage par rapport aux objets d'entrée. Il y a deux définitions liés mais différentes de l'espace de sortie dans l'apprentissage automatique. La première est l'espace de sortie de la tâche, ce qui est fortement dépendante de l'application. Par exemple, dans la régression, l'espace de sortie est l'espace de nombres réels R; dans le classement l'espace de sortie est un ensemble discret {1, 2,. . . , K}. La seconde est l'espace de sortie afin de faciliter le processus d'apprentissage. Cela peut différer de l'espace de sortie de la tâche. Par exemple, lorsqu'on utilise la régression pour résoudre le problème de la classification, l'espace de sortie qui facilite l'apprentissage est l'espace de nombres réels, mais pas un ensemble discret.

3. ***Espace des hypothèses,*** qui définit la classe des fonctions H permettant d'approximer la relation probabiliste qui relie l'espace d'entrée X et l'espace de sortie Y.

Autrement dit, ces fonctions exploitent les vecteurs de caractéristiques des objets d'entrée, et faire des prédictions selon le format de l'espace de sortie.

4. ***Fonction d'erreur (risque* ou *coût),*** Pour apprendre l'hypothèse optimale, une base d'apprentissage est généralement utilisée, qui contient un certain nombre d'objets et leurs étiquettes, échantillonnés à partir de l'espace d'entrée et de sortie. La fonction d'erreur mesure l'accord entre la prédiction *f(x)* et la sortie désiré *y* pour un couple *(x,y).*

Une fonction $L : Y \times Y \rightarrow R_+$. Intuitivement, *L(f(x), y)* mesure la proximité entre la réponse prédite et la réponse réelle. C'est donc généralement une distance sur l'ensemble *Y*.

En classification, l'erreur instantanée généralement considérée est :

$$L_c(f((x), y) = [\![f(x) \neq y]\!] \quad (3.1)$$

Ou $[\![P]\!]$ vaut 1 si le prédicat **P** est vrai, 0 sinon.

En régression, la fonction d'erreur instantanée couramment utilisée est la distance sur *R* définie par le carré de la valeur absolue :

$$L_r(f((x), y) = |f(x) - y|^2 \quad (3.2)$$

Il est clair que la fonction de perte joue un rôle fondamental dans l'apprentissage automatique car elle mesure la compréhension de l'algorithme (c'est-à-dire quelle prédiction est bonne et celle qui ne l'est pas). Et avec cette fonction de perte, un risque empirique peut être définie sur l'ensemble d'apprentissage, et l'hypothèse optimale est généralement (mais pas toujours) apprise avec la minimisation du risque empiriques.

Nous représentons graphiquement la relation entre ces quatre éléments dans la Figure 3.1.

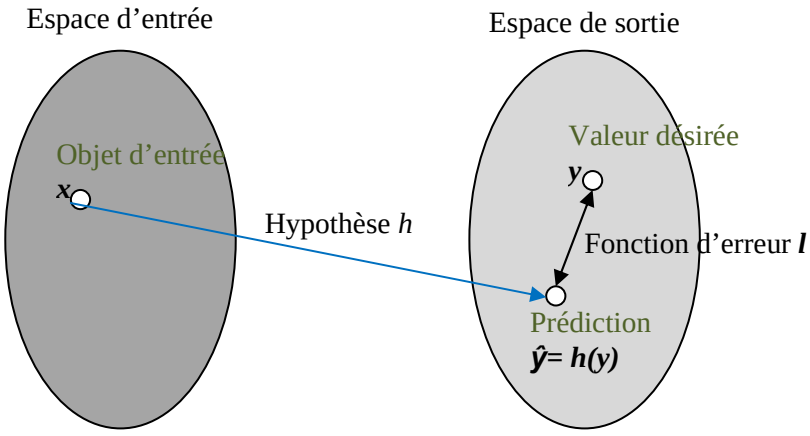

Figure 3. 1 : Apprentissage automatique supervisé.

3.2.2. Ensembles de données et généralisation du modèle

L'efficacité des algorithmes d'apprentissage est mesurée en vérifiant leurs capacités de généralisation, soit leurs capacités à entraîner un modèle à partir d'un ensemble d'apprentissage *(Training Set)* et qui retourne de bonnes réponses lorsque de nouvelles données de l'ensemble de test *(test Set)* lui sont présentées.

L'ensemble d'entrainement S et l'ensemble de test S_{test} sont semblable et contenant des observations issues du même processus. Puisque *S* et S_{test} sont indépendants, on peut utiliser *S* pour trouver une fonction f et évaluer sa performance en calculant son erreur sur S_{test}.

3.3. Apprentissage des fonctions d'ordonnancement

En recherche d'information, la tâche principale des systèmes de recherche d'information consiste à trouver des documents qui sont pertinents pour une requête d'utilisateur donnée. Ces documents sont ensuite présentés classé (ordonné) en fonction de leur pertinence. La pertinence est déterminée par une fonction d'ordonnancement.

L'apprentissage d'ordonnancement est une approche d'apprentissage automatique appliquée à la recherche d'information permettant de développer automatiquement des fonctions d'ordonnancement à partir de données d'apprentissage. Il s'agit de considérer une collection de couples (requête, document) avec leurs jugements de pertinence. Le modèle de Ranking est appris à l'aide d'un algorithme d'apprentissage sur ce jeu de données. Ce modèle est ensuite utilisé pour prédire la pertinence de nouvelles paires (requête, document) et fournir l'ordre total.

La figure 3.2 ci-dessous montre le cadre général de l'apprentissage d'ordonnancement. Étant donné que l'apprentissage d'ordonnancement et une sorte d'apprentissage automatique supervisé, une base d'apprentissage est alors nécessaire. La création de la base d'apprentissage est très similaire à la création de la base de test pour l'évaluation. Supposons qu'il existe une collection de *n* requêtes pour l'apprentissage, notées par $q_i(i=1,\ldots,n)$, leurs documents associés représentés par des vecteurs de caractéristiques $X^{(i)} = \{x_j^{(i)}\}_{j=1}^{m}$ (où m est le nombre de documents associés à la requête q_i), qui décrivent la similarité entre le document et la requête, et $Y = (y_i^1, \ldots, y_i^m)$ sont les jugements de pertinence des documents par rapport à la requête. Ensuite, un algorithme d'apprentissage est utilisé pour apprendre le modèle de Ranking en combinant les caractéristiques de telle sorte que le modèle de Ranking

peut prédire avec la plus grande précision possible l'étiquette réelle, en terme de la fonction d'erreur. Dans la phase de test, le modèle appris dans la phase d'apprentissage est appliqué sur les nouvelles requêtes pour trier les documents et renvoyer la liste ordonnée correspondante à l'utilisateur comme réponse à sa requête.

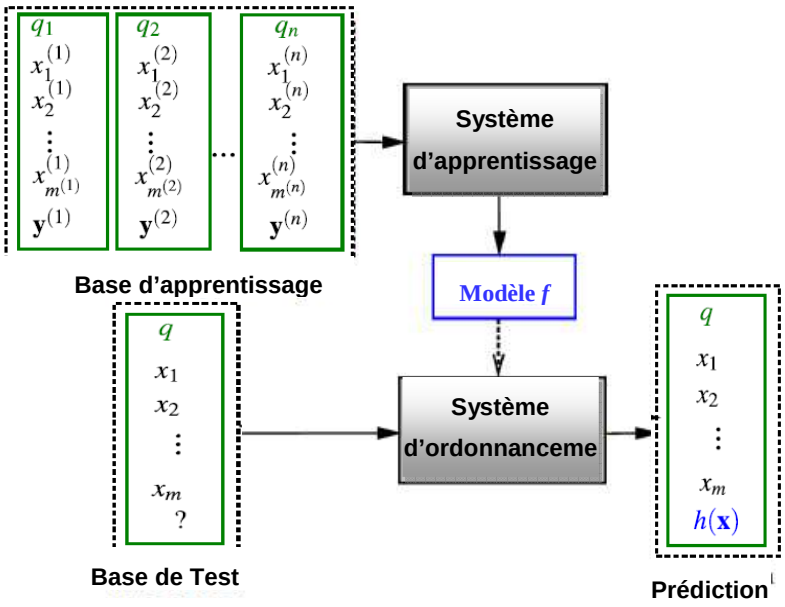

Figure 3. 2 : Le cadre générale de l'apprentissage d'ordonnancement [Liu 11].

Plusieurs algorithmes d'ordonnancements ont été développés ces dernières années dans le cadre de l'apprentissage d'ordonnancement. Chaque algorithme utilise une technique d'apprentissage et une fonction d'erreur différente de l'autre. Dans **[Liu 11]** l'auteur a catégorisé ces algorithmes d'ordonnancement en trois approches selon la façon de considérer les documents en entrée du système d'apprentissage.

- Approche par document seul (Pointwise Approach) : un seul document en entrée de l'algorithme.
- Approche par paire de documents (Pairwise Approach) : une paire de document en entrée de l'algorithme.
- Approche par Liste de documents (Listwise Approach) : la liste de tous les documents associés à une requête est l'entrée de l'algorithme.

Le tableau ci-dessous illustre une comparaison entre ces trois approches :

Catégorie	Pointwise	
	Régression	**Classification**
Espace d'entrée	Un seul document comme instance d'apprentissage	
Espace de sortie	Variable à prédire ***Y*** est un score (note) de pertinence	-Dichotomique {-1,1} (non pertinent, pertinent) - Présenter plusieurs classes (non pertinent, pertinent faible, pertinence moyenne, pertinence forte)
Fonction d'ordonnancement	$f(x_j)$	$f(x_j)$
Fonction d'erreur	$L_r(f((x),y) = \lvert f(x) - y\rvert^2$	$L_c(f((x),y) = [\![f(x) \neq y]\!]$
Catégorie	**Pairwise**	**Listwise**
Espace d'entrée	Paire de documents (x_u, x_v)	Liste de documents $X = \{x_j\}_{j=1}^{m}$
Espace de sortie	Préférence $y_{u,v}$	Liste ordonnée documents π_y
Fonction d'ordonnancement	$2.I_{\{f(x_u)>f(x_v)\}} - 1$	$Sort \circ f(X)$
Fonction d'erreur	$L(f, x_u, x_v, y_{u,v})$	$L(f, X, \pi_y)$

Tableau 3.1 : Résumé des approches de l'apprentissage de fonction d'ordonnancement [Liu11].

3.3.1. Approche pointwise :

L'approche pointwise est la première approche considérée par les recherches en apprentissage d'ordonnancement.

L'espace d'entrée de l'approche contient un vecteur de caractéristique de chaque document seul. L'espace de sortie contient le degré de pertinence de chaque document de l'entrée. Différents types de jugements peuvent être converties en étiquettes réelle en termes de degré de pertinence :

- Si le jugement est directement donné comme degré de pertinence *lj*, l'étiquette réelle pour le document *xj* est définie comme *yj = lj*.
- Si le jugement est donnée comme préférence par paires, *lu,v* on peut obtenir l'étiquette réelle d'un document en comptant le nombre de documents classés au-dessus de ce document.
- Si le jugement est donné en tant que *πl* (ordre total des documents), on peut obtenir l'étiquette en utilisant une fonction de mappage. Par exemple, la position du document dans *πl* peut servir cette étiquette.

L'espace des hypothèses contient des fonctions qui prennent le vecteur de caractéristiques d'un seul document en entrée et essayent de prédire le degré de pertinence de ce document.

Dans cette approche plusieurs méthodes d'apprentissage automatique ont été appliquées directement au problème d'ordonnancement à savoir, la classification la régression et la régression ordinale. La minimisation de leurs fonctions de perte a été utilisée pour améliorer la performance du modèle de Ranking.

Plusieurs algorithmes ont été développés dans cette approche à savoir :

3.3.1.1. Ordonnancement par régression

Cossock et Zhang **[Cos 06]** ont résoulu le problème d'ordonnancement en le réduisant à la régression. Étant donné $X = \{x_j\}_{j=1}^{m}$ un ensemble de documents associés à la requête d'apprentissage q, et l'ensemble des étiquettes $Y = \{y_j\}_{j=1}^{m}$ de ces documents, supposons qu'une fonction de score f est utilisée pour classer ces documents. La fonction de perte définie est l'erreur des moindres carrés (c'est-à-dire le carré de la différence entre l'étiquette de référence et le score prédit) pour optimiser le classement des premiers résultats d'une liste.

$$L(f; x_j; y_j) = (y_j - f(x_j))^2 \qquad (3.3)$$

La figure 3.3 montre la courbe de la fonction d'erreur des moindres carrés. De la figure on peut constater que l'erreur vaut zéro (il n'ya pas de perte), si est seulement si la sortie de la fonction de score $f(x_j)$ est exactement égal aux étiquettes y_j. Sinon, la perte augmente de façon quadratique. Autrement dit, pour un document pertinent avec l'étiquette 1, si la sortie de la fonction de prédiction est de 2 ce qui semble être une prédiction encore plus forte de la pertinence de ce document, il y aura une certaine perte !

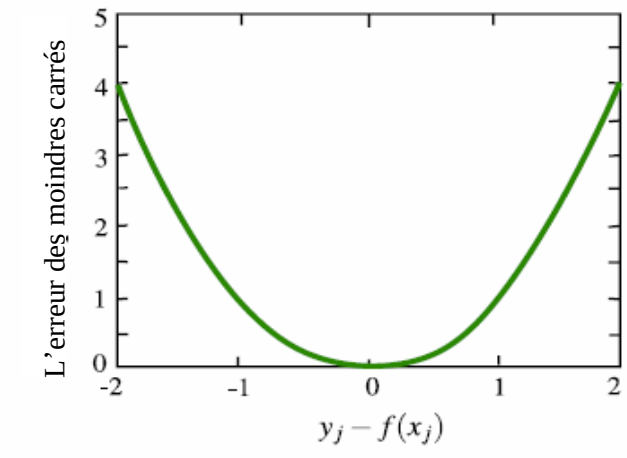

Figure 3. 3 : Fonction d'erreur des moindres carrés.

3.3.1.2. Ordonnancement par classification binaire

Plusieurs travaux ont utilisé les modèles de classification pour calculer la pertinence des documents et traiter le problème de ranking en recherche d'information, par exemple, **[Gey 94] [Coo 92]** et **[Nal 04]**. Ici nous prenons **[Nal 04]** à titre d'exemple pour expliquer l'idée de base. Il s'agit d'une application directe des SVM proposé par Vapnik **[Vap 99]**. Celle-ci cherche à calculer l'hyperplan permettant de séparer correctement les exemples positifs (doc pertinents) et les exemples négatifs (doc non pertinents) en maximisant la marge le plus possible entre l'hyperplan et les documents les plus proches de celui-ci.

Soit $\{x_j\}_{j=1}^{m}$ un ensemble de documents et leurs jugements de pertinence $Y = \{y_j\}_{j=1}^{m}$ associés à la requête q, dans ce cas les étiquettes sont binaires (c.-à-d. $y_j = +1$ pour tous les documents pertinents et $y_j = -1$ pour tous les documents non pertinents.

Lorsqu'on applique les SVM sur le problème d'ordonnancement, on construit un modèle sous forme d'une équation linéaire, c.-à-d $f(x) = W^T x$. D'où la formulation mathématique suivante :

$$min \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m^{(i)}} \xi_j^{(i)}$$

$$s.c \quad W^T x_j^{(i)} \leq -1 + \xi_j^{(i)}, \ if\ y_j^{(i)} = -1 \qquad (3.4)$$

$$W^T x_j^{(i)} \geq 1 - \xi_j^{(i)}, \ if\ y_j^{(i)} = +1$$

$$\xi_j^{(i)} \geq 0, \ j = 1, \dots, m^{(i)}, i = 1, \dots, n,$$

Avec $x_j^{(i)}$ est le j ème document et $y_j^{(i)}$ son étiquette par rapport à la requête q_i, $m^{(i)}$est le nombre de documents associé à la requête q_i.

L'objectif de la fonction d'optimisation ci-dessus est de déterminer si chaque document de la base d'apprentissage est classé dans la classe appropriée. La fonction d'erreur utilisée est le Hinge Loss (l'erreur charnière), c.-à-d. $L(f(x_j); y_j) = max(0,1 - y_j f(x_j))$.

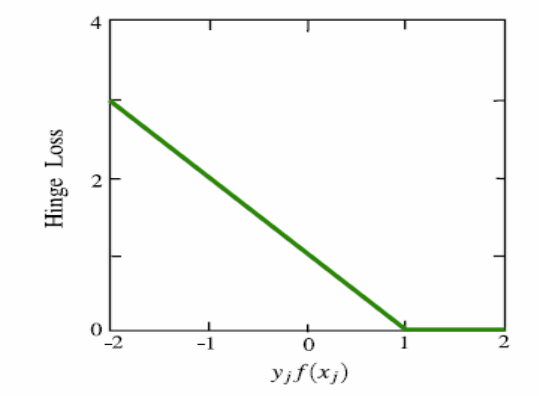

Figure 3. 4 : Fonction d'erreur charnière (Hinge Loss).

3.3.1.3. Régression ordinale

Les algorithmes basés sur la régression ordinale peuvent être considéré comme des algorithmes de classification multi-classes mais ils prennent en considération les relations d'ordre existant entre les classes lors de l'apprentissage du modèle d'ordonnancement.

Considérant k classes, l'objectif de la régression ordinale est de trouver les k-1 seuils $b_r(b_1 < b_2 < \cdots < b_{k-1})$ qui séparent les k classes ainsi que d'apprendre une fonction de score *f* capable de répartir correctement les prédictions dans chaque classe.

Prank **[CRA 02]** est l'algorithme le plus connu en régression ordinale c'est un algorithme itératif dédié à cette tâche. Considérant un vecteur de poids *w* et des seuils initiaux $b_r \; \forall r \in \{1, \dots, k-1\}$, à chaque itération *t* l'algorithme d'apprentissage prend une instance x_j associée à une requête q. La prédiction $\hat{y}_j = \arg\min_k (W^T x_j - b_r < 0)$ est calculée et comparée à l'étiquette réelle y_j. Si l'algorithme a fait une erreur (c.-à-d. $y_j \neq \hat{y}_j$), le vecteur *w* et les seuils b_r sont alors modifiés pour que l'étiquette prédite soit la plus proche possible de l'étiquette réelle. Le processus est alors itéré jusqu'à ce que l'algorithme d'apprentissage converge.

La figure suivante illustre la règle de modification de *W* et les seuils b_r .

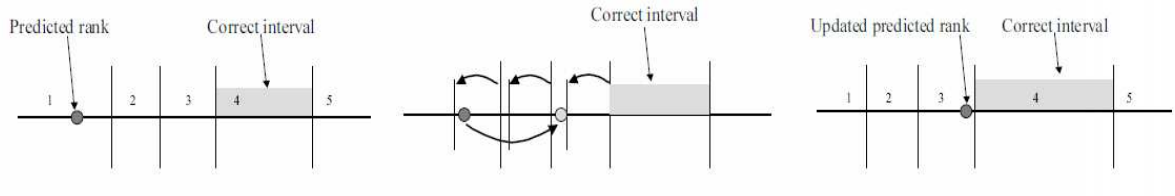

Figure 3. 5 : Une illustration de la règle de modification dans Prank [CRA 02].

3.3.1.4. Limite de l'approche pointwise

Cette approche suppose que le degré exact de la pertinence de chaque document est l'objectif de la prédiction et ne tient pas en compte de l'ordre entre les documents (la position des documents dans la liste). Par ailleurs, l'approche ne fait pas usage que certain documents sont effectivement associés à la même requête. Sachant que les mesures d'évaluation des SRI (NDCG ou MAP) se basent sur les positions des documents dans la liste renvoyée par le système et par rapport à chaque requête.

3.3.2. Approche pairwise

Comme nous avons vu dans la section précédente, les approches pointwise tentent de prédire le score de pertinence des documents en se basent sur les jugements de pertinence de chaque document tout seule. Au contraire, les approches pairwise visent la prédiction de l'ordre relatif entre chaque paire de documents. En ce sens, ce sont des approches plus proches au concept du ranking (ordonnancement) que les approches pointwise. Le but de ces approches est de déterminer pour chaque paire de documents lequel est plus pertinent que l'autre. Formellement, cela revient à minimiser le nombre de paires de documents mal ordonnés.

L'espace d'entrée de l'approche contient une paire de documents $\left(x_u^{(i)}, x_v^{(i)}\right)$ associés à la requête q_i représentés par leurs vecteurs de caractéristiques. L'espace de sortie contient le jugement de préférence $y_{u,v}$ entre chaque paire de documents qui prend des valeurs dans $\{+1, -1\}$. Différents types de jugements de pertinence peuvent être convertis en étiquettes réelles en termes de préférence par paire :

- Si le jugement de pertinence donné comme degré de pertinence l_j, l'étiquette réelle pour la paire (x_u, x_v) est définie comme $y_{u,v} = 2.I_{\{l_u > l_v\}} - 1$ ($I_{\{X\}}$ vaut 1 si la proposition X est vrai, 0 sinon).

- Si le jugement de pertinence est directement donné comme préférence par paires, $l_{u,v}$ on peut l'utiliser directement comme étiquette réelle $y_{u,v} = l_{u,v}$

- Si le jugement de pertinence est donné en tant que *πl* (ordre total des documents), l'étiquette réelle pour la paire (x_u, x_v) est définie comme $y_{u,v} = 2.I\{\pi_l(u) < \pi_l(v)\} - 1$. Avec $\pi_l(.)$ est la position de l'objet dans la liste.

L'espace des hypothèses contient des fonctions qui prennent le vecteur de caractéristiques d'une paire de documents en entrée et essayent de prédire l'ordre relatif entre ces deux documents.

Plusieurs algorithmes d'ordonnancement pairwise ont été proposés dans la littérature de l'apprentissage de fonction d'ordonnancement, la plupart de ces algorithmes utilisent des classifieurs pour classifier l'ordre pour chaque paire de documents. On peut citer, RankNet **[Bur 05],** un algorithme basé sur les réseaux de neurones. RankBoost **[Fre 03]** une adaptation de Adaboost**[Fre 95]** pour les paires de documents. Enfin, Ranking SVM **[Joa 02][Joa 06],** un algorithme basé sur les SVM.

3.3.2.1. RankBoost

La méthode RankBoost [**Fre 03]** est une adaptation d'Adaboost **[Fre 95]** pour la classification des paires de documents. La seule différence entre RankBoost et AdaBoost est que les objets dans RankBoost sont des paires de documents $(x_u^{(i)}, x_v^{(i)})$ alors que dans AdaBoost sont des documents individuels.

Le but de RankBoost est de déterminer itérativement une fonction $H_T = \sum_{t=1}^{T} \alpha_t h_t$ *(classifieur fort),* en cherchant à chaque itération *t* une fonction h_t *(classifieur faible)* et son poids α_t. La pondération D_t qui assigne plus de poids aux paires de documents pour lesquelles la fonction h_t se trompe.

$$D_{t+1}(x_u^{(i)}, x_v^{(i)}) = \frac{1}{Z_t} D_t(x_u^{(i)}, x_v^{(i)}) \exp\left(\alpha_t\left(h_t\left(x_u^{(i)}\right) - h_t\left(x_v^{(i)}\right)\right)\right) \quad (3.5)$$

A l'itération *t,* Le classifieur faible minimise la fonction de coût suivante :

$$Z_t = \sum_{i=1}^{n} \sum_{u,v:y_{u,v}^{(i)}=1} D_t\,(x_u^{(i)}, x_v^{(i)}) \exp\left(\alpha_t\left(h_t\left(x_u^{(i)}\right) - h_t\left(x_v^{(i)}\right)\right)\right). \quad (3.6)$$

La fonction du modèle de Ranking donnée par RankBoost est : $f(x) = \sum_t \alpha_t h_t(x)$

3.3.2.2. Ranking SVM

Ranking SVM est l'un des premiers algorithmes d'apprentissage d'ordonnancement, proposé par **[Joa 02].** L'idée de base de Ranking SVM est de transformer le problème de Ranking en classification par paires et d'employer les techniques SVM **[Vap 95]** pour effectuer la tâche d'apprentissage.

Les techniques SVM cherchent à trouver l'hyperplan optimal, qui sépare deux classes, pour lequel la distance aux points les plus proches (marge) est maximale (Figure 3.6).

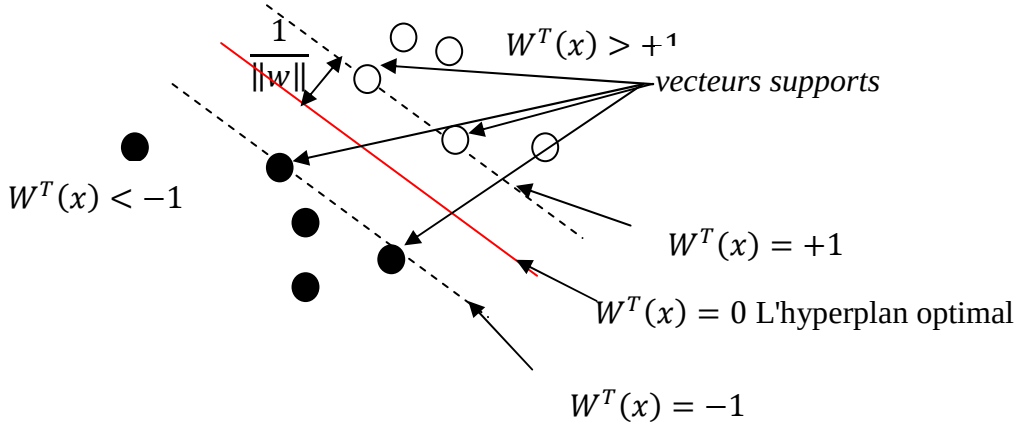

Figure 3. 6 : Machine à vecteurs de support SVM.

Figure 3.7 montre un exemple de Transformation du problème de Ranking en classification par paires. Supposons que x_1, x_2, x_3 trois objets avec trois degrés de pertinence. La différence entre deux vecteurs de caractéristiques à différents degrés construisent les six(06) nouveaux vecteurs de caractéristiques des paires d'objets ($x_1 - x_3, x_1 - x_2, x_2 - x_3$ avec l'étiquette +1 et ($x_3 - x_1, x_2 - x_1, x_3 - x_2$) avec l'étiquette -1.

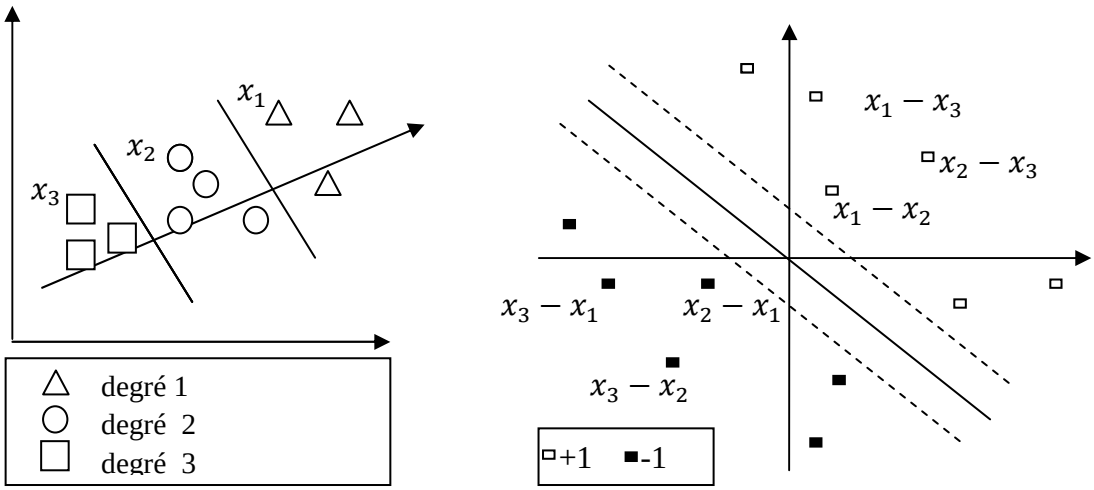

Figure 3. 7 : Transformation du problème de Ranking en classification par paires.

Généralisation pour toutes les requêtes

Etant donnée *n* requêtes d'apprentissage $\{q_j\}_{j=1}^{n}$, avec leurs paires de documents associés. L'ensemble $(x_i^1, x_i^2)_{\ i=1..N}$ sont les N paires de documents représentés par leurs vecteurs de caractéristiques, y_i est l'étiquette réelle correspondante ($y_i = 1$ si le document x_i^1 est plus pertinent que le document x_i^2), $et\ y_i = -1$ sinon. L'objectif alors est d'apprendre une fonction linéaire $f(x) = W^T(x)$ telle que si $f(x_i^1, x_i^2) > 0$ alors x_i^1 doit se classer au-dessus de x_i^2 dans la liste ordonnée.

La formulation mathématique de Ranking SVM est la suivante :

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.c } y_i\,\langle w, x_i^1 - x_i^2\rangle \geq 1 - \xi_i, \qquad (3.7)$$

$$\xi_i \geq 0, \quad i = 1, \ldots, N.$$

Ranking SVM résout un problème d'optimisation quadratique (QP) par la minimisation de la fonction d'erreur (*hinge loss*) :

$$\min \textstyle\sum_{i=1}^{N}[1 - y_i\langle w, x_i^1 - x_i^2\rangle]_+ + \lambda\|w\|^2, \qquad (3.8)$$

où $[z]_+ = \max(0, z)$, dénommée fonction *hinge*, est une fonction convexe. (Voir Figure 3.4). Autrement dit Ranking SVM utilise la fonction $[z]_+$ pour approximer le 0-1 loss.

La minimisation sous contrainte ci-dessus peut être résolue avec les multiplicateurs de Lagrange et le problème équivalent est le suivant :

$$\max_{\propto} \sum_{i=1}^{N} \propto_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \propto_i \propto_j y_i y_j \langle x_i^1 - x_i^2, x_j^1 - x_j^2\rangle$$

$$s.c\ \textstyle\sum_{i=1}^{N} \propto_i y_i = 0, \qquad (3.9)$$

$$0 \leq \propto_i \leq C, \text{ i=1,…,N.}$$

Où C est un paramètre de régularisation qui permet d'intervenir sur le compromis entre l'erreur sur la base d'apprentissage et la généralisation du modèle, N est le nombre des paires de documents dans la base d'apprentissage.

La solution optimale utilisée comme une fonction de ranking est la suivante :

$$f(x) = \textstyle\sum_{i=1}^{N} \propto_i^* y_i \langle \boldsymbol{x}, x_i^1 - x_i^2\rangle. \qquad (3.10)$$

Plusieurs méthodes d'optimisation peuvent être utilisées : on peut citer SMO (optimisation minimale et séquentielle) qui est un algorithme rapide proposé par **[Pla 98]** pour résoudre le problème de programmation quadratique des SVMs.

3.3.2.3. Avantage des approches pairwise

Comme nous avons vu dans cette section, les approches pairwise tentent de prédire l'ordre relatif entre chaque paire de documents. Dans le cas extrême, si cet ordre est juste pour toutes les paires de documents, la liste de tous les documents est bien ordonnée. Malgré que ces approches ne prennent pas en considération que ces paires de documents n'appartiennent pas à la même requête et le nombre des paires de documents se diffèrent d'une requête a une autre (problème que le modèle privilège une requête par rapport à une autre), ces approches ont donné des bonnes performances en qualité de recherche d'information.

3.3.3. Approche listwise

L'espace d'entrée de l'approche contient un ensemble de documents $x = \{x_j\}_{j=1}^{m}$ associés à la requête q_i représentés par leurs vecteurs de caractéristiques. L'espace de sortie contient la liste ordonnée de touts les documents (permutation). Les étiquettes réelles sont affectées aux documents à condition que la permutation des documents soit respectée.

L'espace des hypothèses contient des fonctions qui prennent les vecteurs des caractéristiques de tous les documents en entrée et cherchent de prédire la liste ordonnée des documents selon leurs scores de pertinence. Les algorithmes de cette approche sont répartis en deux sous-catégories : ceux minimisant une fonction d'erreur définie à partir d'une mesure d'évaluation de la RI comme la MAP ou le NDCG (sont appelées méthodes d'optimisation directes) et ceux minimisant une autre fonction de cout non explicitement liée aux mesures d'évaluations de la RI.

Les travaux de la première catégorie suivent la logique suivante selon **[Liu 11]**: puisque la performance des algorithmes est évaluée à partir de mesures de RI, les algorithmes doivent apprendre les fonctions en maximisant ces mesures. Considérant une mesure m, les algorithmes cherchent ainsi à minimiser soit une approximation de la quantité 1-m **[Tay 08] [Zoe 08]**, soit une approximation de la borne supérieure de la quantité 1-m **[Xu 07][Yue 07]**. Les travaux de la deuxième catégorie cherchent quant à eux à minimiser le nombre de permutations entre la liste de référence et la liste restituée par l'algorithme **[Léa 12]**. De nombreuses fonctions de perte ont été proposées dans ce but **[Cao 07] [Xia 08].**

Nous allons détailler l'algorithme ListMLE à titre d'exemple.

3.3.3.1. ListMLE

Dans ListMLE **[Xia 08]**, la fonction d'errer est définie par l'utilisation de la distribution de probabilité sur les permutations. Plusieurs modèles ont été proposées pour représenter cette distribution de probabilité de permutation, à savoir, le modèle Plackett-Luce **[Luce 59] [Pla 75]** et le modèle Mallows **[Mal 75]**. ListMLE applique le modèle Plackett-Luce à l'apprentissage de fonction d'ordonnancement.

Étant donné les scores des documents fournis par la fonction *f*, $S = \{s_j\}_{j=1}^{m}$ où $s_j = f(x_j)$, le modèle Plackett-Luce définit une probabilité pour chaque permutation possible π des documents:

$$p(\pi|S) = \prod_{j=1}^{m} \frac{\varphi(S_{\pi^{-1}(j)})}{\sum_{u=1}^{m} \varphi(S_{\pi^{-1}(u)})} \tag{3.11}$$

Où $\pi^{-1}(j)$ désigne le document classé à la j^{ème} position de la permutation π et φ est une fonction de transformation, qui peut être linéaire, exponentielle ou sigmoïde.

Supposons qu'il y a trois documents A, B, C dont les scores sont S_A, S_B, S_C avec $(S_A > S_B > S_C)$, la probabilité des permutations ABC et CBA sont :

$$P(ABC \mid S) = \left(\frac{S_A}{S_A + S_B + S_C}\right)\left(\frac{S_B}{S_B + S_C}\right)\left(\frac{S_C}{S_C}\right)$$

$$P(CBA \mid S) = \left(\frac{S_C}{S_C + S_B + S_A}\right)\left(\frac{S_B}{S_B + S_A}\right)\left(\frac{S_A}{S_A}\right)$$

La fonction d'erreur utilisée dans ListMLE est la fonction de vraisemblance (*Likelihood function*).

$$L(f; x; \pi_y) = -\log P(\pi_y | f(w, x)). \tag{3.12}$$

Où

$$P(\pi_y \mid f(w, x)) = \prod_{j=1}^{k} \frac{exp(f(x_{\pi^{-1}(j)}))}{\sum_{u=1}^{m} \exp\left(f(x_{\pi^{-1}(j)})\right)} . \tag{3.13}$$

La fonction précédente calcul l'erreur pour une entrée (dans listwise l'entrée est une liste de document, dans ce cas le nombre de document est *k*). L'erreur pour toutes les requêtes est : $L(f; x_i; \pi_{y_i}) = -\sum_{i=1}^{m} \prod_{j=1}^{k} \frac{exp(f(x_{\pi^{-1}(j)}))}{\sum_{u=1}^{m} \exp\left(f(x_{\pi^{-1}(j)})\right)}$; m est le nombre de requêtes.

Le modèle ListMLE utilise les réseaux de neurone et la technique de la descente de gradient pour résoudre le problème de minimisation.

3.3.3.2. Avantage des approches listwise

Contrairement aux approches pairwise qui visent la prédiction de l'ordre relatif entre chaque paire de documents et ne prennent pas en considération la requête dans laquelle appartient chaque paire de documents. Les approches listwise visent à prédire une liste ordonnée de tous les documents de chaque requête. Autrement dit, ces approches visent directement l'objectif des systèmes de recherche d'information.

3.4. Apprentissage d'ordonnancement et RIS

Dans le domaine de la recherche d'information, les algorithmes d'ordonnancement ont été utilisés avec succès dans des tâches comme la méta-recherche, le résumé automatique, filtrage collaboratif et la classification des documents…etc. Néanmoins, peu de travaux concernant l'utilisation de ces méthodes dans la tache de la recherche d'information structurée. Dans **[Vit 06]** les auteurs ont proposé d'utiliser les caractéristiques proviennent de l'élément XML, de son document et de son élément parent puis ils utilisent une méthode d'apprentissage d'ordonnancement pairwise pour combiner les trois caractéristiques. Dans **[Gao 09]** une méthode d'apprentissage d'ordonnancement Listwise appelée ListBM a été utilisé pour apprendre les paramètres *k1* et *b* de la fonction OKAPI BM25 **[Rob 97]**. Les résultats de l'évaluation montrent que LearningBM25 améliore les performances du système mieux que NormalBM25 dans plusieurs taches de la recherche d'information structurée. Dans ce qui suit nous allons détailler ces deux méthodes.

3.4.1. Modèle pairwise pour la RIS

Le premier modèle a utilisé l'apprentissage d'ordonnancement en recherche d'information structurée est le modèle développé par **[Vit 06]** qui utilise une approche pairwise pour apprendre le modèle de Ranking.

Etant donnée *n* requêtes d'apprentissage $\{q_i\}_{i=1}^{n}$, avec leurs paires d'éléments associées (x_i, x_i') et l'étiquette réelle y_i correspondante à chaque paire d'éléments ($y_i = 1$ si l'élément x_i est plus pertinent que l'élément x'_i), $et\ y_i = -1$ sinon. L'objectif alors est d'apprendre une fonction linéaire $f(x) = \sum_{k=1}^{d} w_k x_k$, où *d* est le nombre des caractéristiques utilisées. Si $f(x_i) > f(x'_i)$ alors x_i doit se classer au-dessus de x'_i ($x_i \succ x'_i$) dans la liste ordonnée.

Le coût d'ordonnancement (fonction d'erreur) utilisé est la fonction exponentielle utilisée dans RankBoost **[Fre 03]** telle que La formulation mathématique est la suivante :

$$L(f; x; x'; y_i) = e^{-y_i(f(x)-f(x'))} \quad (3.14)$$

Et le risque empirique est :

$$Re = \sum_{x,x':x\prec x'} e^{f(x)-f(x')} \quad (3.15)$$

Les composantes du gradient de *Re* sont :

$$\frac{\partial Re}{\partial w_k} = \sum_{x,x':x\prec x'} (x_k - x'_k) e^{f(x)-f(x')} \quad (3.16)$$

Pour minimiser $w \rightarrow Re$, on utilise la méthode de la descente de gradient qui donne le vecteur *w* optimale pour le modèle. Et la fonction d'ordonnancement $f(x) = \sum_{k=1}^{d} w_k x_k$ est la fonction utilisée pour ordonner les éléments pour des nouvelles requêtes.

Trois caractéristiques ont été utilisées dans cette approche, À savoir, Le score BM25 de l'élément, le score BM25 du parent et le score BM25 du document entier. La base d'apprentissage est l'ensemble des requêtes et jugements produits par INEX en 2003 et 2004. Ceux qui ont été produit en 2005 ont servi de base de test. L'évaluation des résultats a montré que le modèle proposé pour ordonner des éléments XML améliore de manière significative le modèle de base Okapi (BM25), qui est déjà connu pour être efficace en RI sur des documents plats.

3.4.2. Modèle listwise pour la RIS (ListBM).

Dans ce modèle **[Gao 09]** les paramètres *k1* et *b* du modèle BM25 ont été appris en utilisant les méthodes d'apprentissage d'ordonnancement. On peut classer ListBM dans la catégorie Listwise, car l'entrée de ce modèle d'apprentissage est une liste ordonnée des éléments XML.

Dans la base d'apprentissage, il y a un ensemble de requêtes $Q = \{q_i\}_{i=1}^{m}$. Chaque requête q_i est associée à une liste ordonnée des documents pertinents $D^i = \{d_j^i\}_{i=1}^{n}$ où d_j^i est le j-ième document pertinent pour la requête q_i. $R^i = \{r_j^i\}_{i=1}^{n}$ est l'ensemble des *n* premiers éléments XML ordonnés par la fonction BM25 normale ({*k1*,*b*}={4,0.8}).

La fonction d'erreur à minimiser est définie pour la requête q_i comme suit :

$$L(D^i, R^i) = \frac{mn^2}{\sum_{j=1}^{n} rank_j^i} \tag{3.17}$$

$$rank_j^i = \begin{cases} 0 \; si \; le \; jième \; élément \; de \; R^i \; n'apparait \; pas \; dans \; D^i \\ k \; si \; le \; jième \; élément \; de \; R^i \; apparait \; dans \; le \; kième \; element \; de \; D^i \end{cases} \tag{3.18}$$

Où *m* est le nombre de requêtes, *n* est le nombre de documents pertinents pour la requête q_i, $rank_j^i$ est divisé en deux partie : (1) si le j-ième élément de R^i n'appartient à aucun document pertinent de D^i , alors $rank_j^i$ est mis à 0 ; (2) si le j-ième élément de R^i appartient au k-ième document pertinent de D^i , alors $rank_j^i$ est mis à k.

Le même algorithme ListBM peut apprendre les deux paramètres *k1* et *b* séparément. L'algorithme suivant décrit la procédure d'apprentissage pour le paramètre *k1*.

Algorithm 1. ListBM: The process of learning k_1

Input: query Q, relevant documents D
Parameter: k_1, number of iterations T,
Initialize: b=0.8, $\sum_{i=1}^{m} \mathbf{L(D^i, R^i)}$=+∞
for t=1 **to** T **do**
  for i=1 **to** m **do**
    search the q^i using BM25, get R^i;
    compute $\mathbf{L(D^i, R^i)}$
    update $\mathbf{k_1} += \frac{1}{\mathbf{L(D^i, R^i)}}$
  end for
  if $\sum_{i=1}^{m} \mathbf{L(D^i, R^i)}$ increases, then break;
end for
output the pair of {k_{min}, **min** $\sum_{i=1}^{m} \mathbf{L(D^i, R^i)}$; }

Figure 3. 8 : ListBM : algorithme pour apprendre k*1* de BM25 [Gao 09].

Lors de l'apprentissage de *k1* le paramètre *b* est mis à 0.8, et pour apprendre *b* le paramètre *k1* est mis à 4. les résultats de l'apprentissage montrent que les meilleurs paramètres {*k1*,*b*} pour le modèle de Ranking est {35,0.8}.

L'évaluation du modèle montre que l'efficacité de la recherche par LearningBM25 est meilleure que NormalBM25 pour les trois taches de la recherche d'information structurée (Best in Context , Focused et Thorough).

3.5. Evaluation des algorithmes d'apprentissage d'ordonnancement

L'évaluation des algorithmes d'apprentissage d'ordonnancement sur leur capacité à ordonner les documents pertinents au dessus des documents non pertinent se fait via différentes mesures d'évaluation de RI (MAP, NDCG), voir section 1.4. L'algorithme le plus performant sera celui pour lequel la valeur de la mesure est la plus grande.

Collections de référence

La collection LETOR **[Liu 07]**, publiée au début de 2007, est la première collection de référence développée exclusivement pour l'évaluation des algorithmes d'ordonnancement, elle contient deux versions LETOR 3.0 et LETOR 4.0. Pour construire cette collection, deux collections de référence en recherche d'information TREC et OHSUMED ont été utilisées pour extraire les requêtes et les documents correspondants, des caractéristiques qui décrivent la similarité entre le document et la requête ont été prédéfinie et pré-calculée. Pour les jugements de pertinence, des experts humains donnent l'étiquette réelle pour chaque couple document-requête.

Après l'émergence et la réussite de l'apprentissage des fonctions d'ordonnancement pour la RI, Yahoo ![1] et le moteur de recherche Russe (Yandex[2]) ont publié leurs bases de données internes pour les utiliser dans l'apprentissage et l'évaluation des modèles de Ranking. Dans le même cadre, Microsoft research a également publié un ensemble de données d'apprentissage. Le tableau 3.2 montre les statistiques des collections disponibles et utilisées par la communauté de l'apprentissage des fonctions d'ordonnancement.

Collection	Nombre de Documents	Nombre de Requêtes	Nombre de Caractéristiques	Pertinence	Année
Letor 3.0/ Gov	568 k	575	64	2	2008
Letor 3.0/Ohsumed	16 k	106	45	3	2008
Letor 4.0	85 k	1692	46	3	2009
Yandex	213 k	20267	245	5	2009
Yahoo ![3]	883 k	36251	700	5	2010
Microsoft Research	3771 k	31531	136	5	2010

Tableau 3.2 : statistiques des collections de l'apprentissage d'ordonnancement.

[1] http://learningtorankchallenge.yahoo.com/
[2] *www.**yandex**.ru/*
[3] http://learningtorankchallenge.yahoo.com/datasets.php

3.6. Conclusion

Dans ce chapitre, nous avons fait un rapide tour d'horizon des approches d'apprentissage de fonction d'ordonnancement pointwise, pairwise et listwise. Dans les trois cas, nous avons vu que les différentes tâches d'apprentissage consistent à apprendre des paramètres ou des fonctions d'ordonnancement à partir des données d'apprentissage supervisé, afin de prédire l'ordre des documents par rapport a une requête donnée.

Il est observé que, les approches listwise et pairwise ont relativement des meilleures performances que les approches pointwise. Car ces approches visent directement l'objectif des systèmes de recherche d'information, qui est la prédiction de l'ordre des documents et non pas les scores. Récemment, dans « Yahoo Learning to Rank Challenge[4]», LambdaMART **[Qia 10]**, qui est une approche pairwise a obtenu la meilleure performance, malgré que dans la littérature en considère que les approches Listwise sont plus performant que les approches Pairwise.

Nous avons aussi constaté que la collection LETOR facilite l'évaluation et la comparaison entre plusieurs algorithmes d'apprentissage où les caractéristiques sont déjà définies et les jugements de pertinence sont attribués. Si on veut adapter ces algorithmes sur d'autre tache, comme par exemple la recherche d'information XML où les caractéristiques de la collection d'apprentissage sont absentes. En effet, la difficulté réside dans le choix des caractéristiques et l'attribution des jugements de pertinence pour construire cette collection d'apprentissage et apprendre le modèle d'ordonnancement.

Dans le chapitre suivant, nous allons expliquer comment adapter les algorithmes d'apprentissage d'ordonnancement sur la recherche d'information structurée, en précisant les caractéristiques choisis, les éléments XML échantillonnés et la méthode de l'étiquetage des couples élément-requêtes adoptée afin de construire une collection d'apprentissage et apprendre le modèle de Ranking.

[4] ***learningtorankchallenge.yahoo****.com*

CHAPITRE 4 : APPROCHE PROPOSEE ET EVALUATION DES RESULTATS

4.1. Introduction

Nous avons présenté dans les chapitres précédents un état de l'art des travaux sur la recherche d'information et la recherche d'information structurée. Afin de pouvoir retrouver de l'information pertinente au sein des documents XML, plusieurs modèles d'interrogation et de recherche ont été proposés. Tous ces modèles cherchent à utiliser les sources de pertinence pour retrouver les unités d'information (éléments XML) les plus spécifiques et exhaustives au besoin de l'utilisateur. Aussi, un état de l'art sur les méthodes d'apprentissage d'ordonnancement en recherche d'information a été présenté. Ces méthodes ont pour objectif de combiner les caractéristiques (sources de pertinence), qui décrivent la similarité entre les requêtes et les documents, pour apprendre automatiquement le modèle de Ranking permettant d'ordonner les documents du plus pertinent au moins pertinent.

Dans la suite de ce rapport, nous proposons une approche de recherche d'information structurée basée sur les techniques d'apprentissage d'ordonnancement en combinant plusieurs caractéristiques de pertinence des éléments XML.

Cette proposition, utilise l'algorithme Ranking SVM, un des meilleurs algorithmes d'apprentissage d'ordonnancement pour construire un modèle de Ranking pour la recherche d'information structurée. Quatre caractéristiques ont été utilisées, à savoir le score de l'élément, de son document, la proximité entre les termes de la requête dans l'élément et enfin la taille de l'élément.

Nous allons décrire dans ce chapitre la démarche utilisée pour utiliser l'algorithme Ranking SVM en détaillant les expérimentations effectuées et l'évaluation des résultats obtenus.

4.2. Modélisation de l'approche proposée

Nous présentons dans cette partie le modèle général d'ordonnancement qui sera adapté à la RIS. Comme pour n'importe quelle technique d'apprentissage supervisé, nous avons besoin d'un ensemble d'exemples étiquetés (base d'apprentissage) afin d'apprendre le modèle de Ranking. Cette base contient les trois éléments suivants :

- ***Les instances*** : Les instances de l'algorithme d'apprentissage sont les couples élément-requête.
- ***Le Vecteur de caractéristiques*** : le vecteur de caractéristique de chaque instance est construit par les sources de pertinence.

- ***L'étiquette réelle*** : L'étiquette réelle de chaque instance est le jugement de pertinence.

Formulation

Supposons que, Q est l'ensemble de *n* requêtes, E est l'ensemble d'éléments XML et $Y = \{1,2, \dots, l\}$ est l'ensemble d'étiquettes. Il existe un ordre total entre les étiquettes tel que $l \succ l-1 \succ \cdots \succ 1$, où $\succ$ désigne la relation d'ordre. Étant donné un ensemble de requêtes $\{q_1, \dots q_m\}$ sélectionnées aléatoirement à partir de l'ensemble de requêtes Q et $E_i = \{e_1^i, \dots, e_k^i\}$ est l'ensemble de k éléments associés à la requête q_i. La base d'apprentissage est créée comme un ensemble de couple élément-requête. Pour chaque couple $(q_i, e_j^i) \in QXE$:

- une étiquette y_j^i, représentant le degré de pertinence de l'élément e_j^i par rapport à la requête q_i, est affectée par un système d'étiquetage.
- Un extracteur de caractéristique produit un vecteur de caractéristique contient nb caractéristiques (sources de pertinence), qui décrits la similarité entre q_i et e_j^i.

Les couples élément-requêtes, leurs vecteurs de caractéristiques et les jugements de pertinence correspondant sont les entrées de l'algorithme d'apprentissage. La sortie est une fonction d'ordonnancement, f, où $f(q_i, e_j^i)$, est censée donner le vrai jugement de pertinence pour l'élément e_j^i par rapport à la requête q_i.

A partir de cet ensemble d'apprentissage, l'algorithme d'apprentissage d'ordonnancement tente d'apprendre une fonction de Ranking, qui est une combinaison de toutes les caractéristiques $f(e,q) = \sum_{i=1}^{nb} w_i * c_i$, tel que, w_i est le poids de chaque caractéristique. Cette fonction peut attribuer un score à chaque instance (élément-requête), de telle sorte que lors du tri des éléments selon ces scores les plus pertinents apparaissent au début de la liste de classement.

Dans la phase de test, les données de test se composent de nouvelles requêtes $\{q_{m+1}, \dots q_n\}$ et leurs éléments associés E_i. Nous créons le vecteur de caractéristiques pour chaque couple élément-requête et la fonction de Ranking résultante de la phase apprentissage est appliquée pour attribuer des scores aux éléments associés de chaque requête. Enfin trier ces éléments selon ces scores, puis renvoyer la liste ordonnée. La figure suivante présente la modélisation de l'approche proposée.

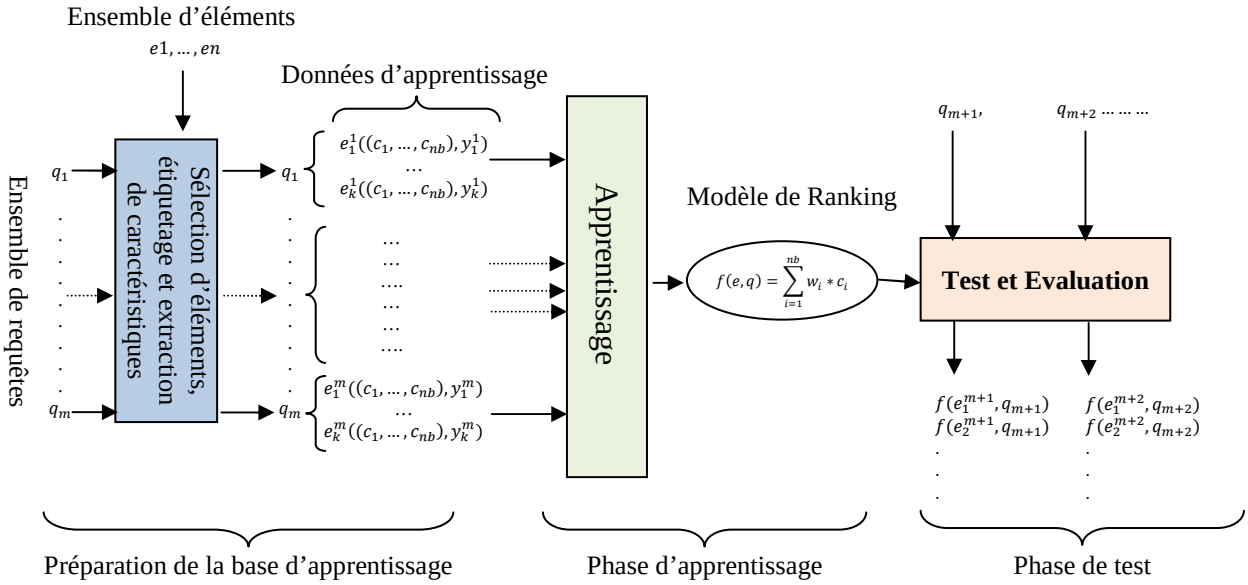

Figure 4. 1 : Modélisation de l'approche proposée pour l'apprentissage d'ordonnancement en RIS.

4.3. Algorithme choisi

Ranking SVM **[Joa 06],** décrit à la section 3.3.2.2, est l'un des algorithmes de référence pour l'apprentissage des fonctions d'ordonnancement. Nous avons choisi cet algorithme puisque son évaluation sur de nombreux jeux de données de la collection LETOR montre que sa performance est excellente **[Liu 11]** et plusieurs implémentations sont mises à disposition librement et maintenues par l'auteur sur sa page personnelle[5].

Le principe de l'algorithme est de transformer le problème d'ordonnancement en classification par paire. Comme montre l'exemple suivant :

Supposons trois éléments e1, e2 et e3 chaque élément avec le vecteur de caractéristiques correspondant et l'étiquette réelle associée pour chaque requête q1 et q2. L'ordre de ces trois éléments par rapport à leurs jugements de pertinence est comme suit :

- Pour la requête q1 l'ordre est : e1 > e2 > e3.
- Pour la requête q2 l'ordre est: e2 > e3 > e1.

[5] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Requêtes	Instances (éléments)	Vecteurs de caractéristiques	Etiquettes (jugements de pertinence)
q_1	e1	x_1^1	3
	e2	x_2^1	2
	e3	x_3^1	1
q_2	e1	x_1^2	1
	e2	x_2^2	3
	e3	x_3^2	2

Tableau 4.1 : exemple de trois éléments et leurs vecteurs de caractéristiques.

Apres la transformation en classification par paire, les instances seront des paires d'éléments et les étiquettes ne seront plus des jugements de pertinence mais représentent l'ordre relatif entre les éléments, si e1 est plus pertinent à e2 alors la paire d'élément (e1, e2) est étiqueté par +1, sinon -1, comme montre le tableau suivant.

Requêtes	Instance (paires d'éléments)	Vecteurs de caractéristiques	Etiquettes (classes)
q_1	(e1, e2)	$x_1^1 - x_2^1$	**+1**
	(e1, e3)	$x_1^1 - x_3^1$	**+1**
	(e2, e3)	$x_2^1 - x_3^1$	**+1**
q_2	(e1, e2)	$x_1^2 - x_2^2$	**-1**
	(e1, e3)	$x_1^2 - x_3^2$	**-1**
	(e2, e3)	$x_2^2 - x_3^2$	**+1**

Tableau 4.2 : les nouvelles instances avec leurs vecteurs de caractéristiques après la transformation par paire.

Pour classer les paires positives dans la classe +1 et les paires négatives dans la classe -1, l'algorithme Ranking SVM minimise la fonction objectif suivante **[Liu 11].**

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \qquad (4.1)$$

$$\text{s.c } y_i \langle w, x_i^1 - x_i^2 \rangle \geq 1 - \xi_i, \qquad \xi_i \geq 0, \quad i = 1, \dots, N.$$

Où N est le nombre des paires d'éléments, w est le vecteur des poids des caractéristiques, C est un paramètre de régularisation qui permet d'intervenir sur le compromis entre l'erreur sur la base d'apprentissage et la généralisation du modèle

L'algorithme d'apprentissage Ranking SVM est résumé comme suit.

Paramètre C

Entrée : les paires d'éléments $\left\{\left((x_i^1, x_i^2), y_i\right)\right\}, i = 1 \ldots., N$ /* x_i^1, x_i^2 sont les vecteurs de caractéristiques des paires d'éléments, N est le nombre des paire d'éléments*/

Résolution du dual lagrangien : $\max_{\propto} \sum_{i=1}^{N} \propto_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \propto_i \propto_j y_i y_j \langle x_i^1 - x_i^2, x_j^1 - x_j^2 \rangle$

Pour trouver $\propto_i^*$, (i=1,…,N)

Sortie : $f(x) = \sum_{i=1}^{N} \propto_i^* y_i \langle x, x_i^1 - x_i^2 \rangle$ /* le modèle de Ranking */

Figure 4. 2 : Algorithme Ranking SVM.

4.4. Caractéristiques utilisées

Dans cette section, nous introduisons la représentation des caractéristiques des éléments XML utilisées dans notre approche. Quatre caractéristiques sont utilisées pour construire le modèle : le score BM25 de l'élément, le score BM25 du document, la taille de l'élément et la proximité des termes de la requête dans l'élément.

4.4.1. Score du document (BM25Doc)

Tenir compte du contexte d'un élément XML pour estimer sa pertinence peut améliorer la RIS. Par exemple, un élément d'un document pertinent et ne contient pas tous les termes de la requête, est susceptible d'être plus pertinent que s'il est contenu dans un document non pertinent. Le processus de combiner le score de l'élément et celui de son contexte est appelée contextualisation où bien « à base de contexte ». Le contexte peut être, le parent de l'élément, quelques ancêtres de l'élément ou bien le document entier. La technique de contextualisation la plus courante consiste à utiliser le document contenant l'élément (c'est à dire l'élément racine) comme contexte **[Lal 09]**.

Dans la recherche d'information, BM25 **[Rob 97]** est une des plus citées fonction de score utilisée pour estimer la pertinence des documents pour une requête donnée. elle est basée sur le modèle probabiliste. Dans notre approche, ce score est utilisé comme une caractéristique pour estimer le score des documents à une requête donnée.

$$BM25(d,q) = \sum_{t \in q} idf(t,d) \frac{(k1+1)tf(t,d)}{k1.\left((1-b)+b\frac{len(d)}{avdl}\right)+tf(t,d)} \quad (4.2)$$

$$idf(t,d) = log \frac{|D|}{df(t)} \quad (4.3)$$

Où $|D|$ le nombre de documents dans la collection; $df(t)$est le nombre de documents contenant le terme *t*; $tf(t,d)$ est la fréquence de terme *t* dans le document *d*; $len(d)$ désigne la longueur du document et *avdl* est la longueur moyenne des documents.

K1 et *b* sont des paramètres libres. Nous avons mis k1 = 2,5 et b = 0,8.

4.4.2. Score de l'élément (BM25Elem)

Quoique, à l'origine, la fonction BM25 est utilisée pour calculer le score des documents dans la RI des documents plats. Elle a été introduite dans la RI XML ces dernières années et les modèles qui l'utilisent comme une fonction de base, fonctionnent le mieux dans INEX **[Gao 11]**. De ce fait, nous allons utiliser la fonction du score BM25 des éléments comme une caractéristique dans notre approche.

$$BM25(e,q) = \sum_{t \in q} ief(t) \frac{(k1+1)tf(t,e)}{k1.\left((1-b)+b\frac{len(e)}{avel}\right)+tf(t,e)} \quad (4.4)$$

$$ief(t,e) = log\frac{|E|}{ef(t)} \quad (4.5)$$

Où $|E|$ le nombre d'éléments dans la collection; $ef(t)$est le nombre d'éléments contenant le terme *t*; $tf(t,e)$ est la fréquence de terme *t* dans l'élément *e*; $len(e)$ désigne la longueur de l'élément et *avel* est la longueur moyenne des éléments.

K1 et *b* sont des paramètres libres. Nous avons mis k1 = 2,5 et b = 0,8.

Calcul de « ief »

En raison du nombre important des éléments XML, et la différence de taille entre ces éléments. Des approches calculent *ief* pour chaque type d'élément et concédèrent que $|E|$ est le nombre d'éléments d'un type donné. D'autres approches calculent un seul *ief* pour tous les types d'éléments sélectionnés de la collection (un grand sous ensemble des éléments). Les résultats expérimentaux produits par BM25 adapté à la RIS suggèrent qu'une meilleure performance est obtenue par l'estimation de *ief* pour tous les éléments au lieu par type d'élément **[Lal 09]**.

Dans notre approche nous avons calculé *ief* pour tous les types d'éléments sélectionnés de la collection. Le tableau suivant montre Le sous ensemble des types d'éléments sélectionnés comme défini en **[Lu 05].**

Nom du continue	Type d'élément
Article	Article
Titre de l'article	Atl
Résumé de l'article	Abs
Corps du texte	Bdy
Section	sec, ss1, ss2, ss3
Titre de section	St
Paragraphe	ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3
Bibliographie	Bib
Annexe	Bm

Tableau 4.3 : Types d'éléments sélectionnés de la collection.

4.4.3. Proximité des termes de la requête (ProxElem)

Il est intuitif que, plus les termes de la requête se retrouvent proches dans un élément alors plus cet élément est pertinent et doit être positionné en tête de la liste des réponses retournées par le système de recherche d'information. L'utilisation de la distance entre les termes de la requête dans les modèles de la RI a amélioré la qualité des résultats trouvés **[Ste 06].** Cette mesure a été adaptée au contexte de la RI dans les documents XML de plusieurs façons **[Abb 08] [Bro 08] [Bei 07]**. Dans notre approche, nous calculons la distance entre deux mots-clés de la requête *ti* et *tj* dans un élément *e* comme la distance minimale entre toutes les occurrences de ti et tj dans l'élément *e*.

$$Dis(e, t_i, t_j) = min_{dist}(e, t_i, t_j), \quad (4.6)$$

$$ProxElem(e, t_i, t_j) = \frac{1}{Dist(e,t_i,t_j)}, \quad (4.7)$$

Pour une requête qui contient les mots clés $(t_1, \dots, t_n)$, $ProxElem(e, q)$ est calculée par :

$$ProxElem(e, q) \sum_{i=1..n, j=1..n, i \neq j} \frac{1}{Dist(e,t_i,t_j)} \quad (4.8)$$

4.4.4. Taille de l'élément (SizeElem)

Pour tenir compte de la grande différence des tailles d'éléments dans les documents XML, et en particulier le nombre importants des petits éléments (éléments feuilles) par rapport aux éléments de taille plus grande (articles ou éléments intermédiaires) dans les collections des documents structurés. Plusieurs méthodes ont été adoptées dans la RIS pour résoudre ce problème. Dans **[Kam 04]** une préférence en faveur des éléments de grande taille a été établie et un seuil a été défini pour la suppression des éléments de petite taille de l'index. Nous trouvons aussi dans **[Sig 06]** une analyse faite sur la distribution de la taille des éléments évalués strictement pertinents dans la collection INEX 2005. Cette analyse montre que les éléments de taille 56-100 termes (bins =7-8, bin est un intervalle) représentent la majorité des éléments pertinents. La figure 4.3 montre cette analyse avec :

Micro : le pourcentage des éléments pertinents dans chaque intervalle.

Macro : le pourcentage des éléments pertinents dans chaque intervalle en moyenne des requêtes.

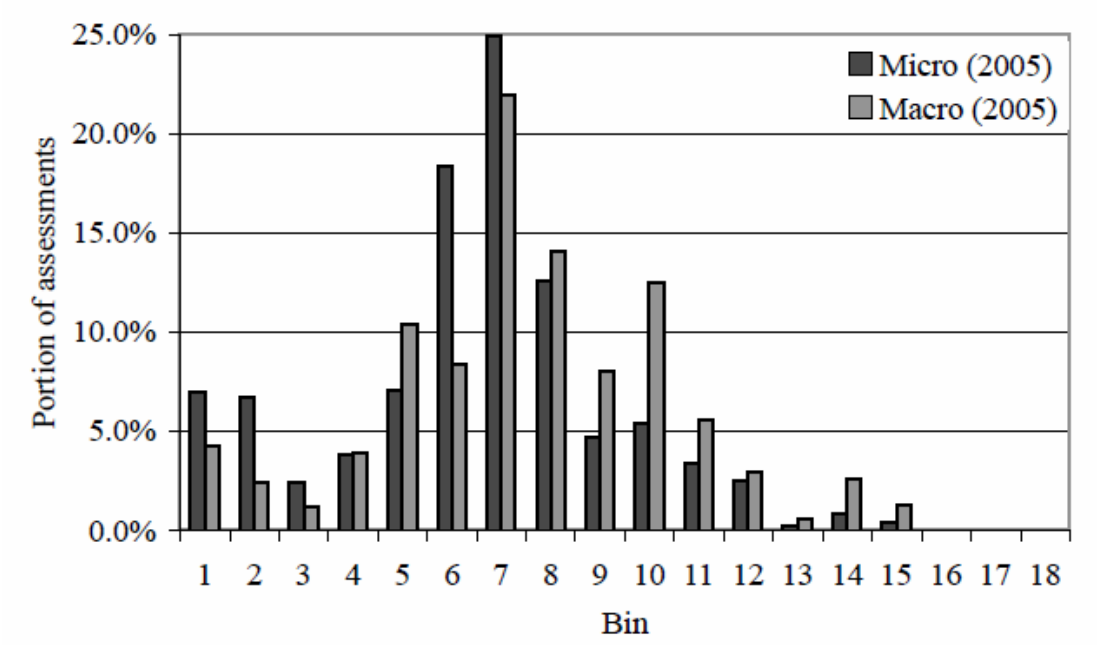

Figure 4. 3 : Distribution de la taille des éléments évalués strictement pertinent dans la Collections INEX 2005[Sig 06].

Proposition d'un nouveau score

Dans notre approche nous avons proposé un nouveau score pour la taille de l'élément (SizeElem) qui est basé sur l'analyse précédente et les deux propriétés suivante :

Propriété 1 : Plus un élément a des éléments fils pertinents, plus son exhaustivité pour la requête est élevé (exhaustivité se propage dans l'arbre), mais il est moins spécifique en raison de sa grande taille.

Propriété 2 : Plus la taille d'un élément pertinent est petite, plus il est spécifique à la requête mais son exhaustivité est faible.

Donc, les éléments les plus équilibrés (ceux qui ont une bonne exhaustivité et une bonne spécificité en même temps) sont situés au milieu de l'arbre XML et ont une taille moyenne. Pour cette raison, nous introduisons un nouveau score de taille qui attribue un score faible aux éléments ayant une petite et grande taille. Et attribuer un score élevé pour les d'éléments dont la taille est proche de la moyenne. Ce score est calculé comme suit:

$$SizeElem = \begin{cases} \frac{len(e)}{IdealLenth} & si\ len(e) \leq IdealLenght \\ \frac{len(e)-(MaxLenght+1)}{IdealLenght-(MaxLenght+1)} & si\ \ len(e) \geq IdealLenght \end{cases} \quad (4.9)$$

Où $len(e)$ est la longueur de l'élément e (nombre de termes) ; $MaxLenght$ représente la longueur maximale des éléments dans la collection et $IdealLenght$ représente la longueur idéale de l'élément à trouver.

Dans les expérimentations, nous avons mis $IdealLenght = 100\ terme$.

Nous détaillons par la suite les expérimentations et l'évaluation des résultats obtenus de notre approche.

4.5. Expérimentations et évaluation

Dans cette section, nous allons présenter la collection INEX utilisée pour évaluer notre approche. Nous décrirons ensuite l'échantillonnage des éléments et des requêtes pour l'apprentissage. Enfin, l'évaluation de notre approche sera introduite.

4.5.1. Données expérimentales

Nous nous appuyons pour l'évaluation de notre approche sur la collection de test fournie dans le cadre de la compagne d'évaluation INEX 2005 (INitiative for the Evaluation of XML Retrieval). Cette collection composée d'articles scientifiques provenant de l'IEEE Computer Society, balisés au format XML. Elle comporte 16819 articles publiés de 1995 à 2004 provenant de 21 magazines ou revues différentes ayant une taille totale d'environ 700 mégaoctets. La collection contient au total 11 millions d'éléments.

4.5.1.1. Indexation

Nous avons utilisé pour l'indexation l'outil Restad6 , un outil d'indexation et de stockage de documents XML dans une base de données relationnelle, pour faire de la recherche d'information. Plusieurs modifications ont été faites sur l'outil afin de faciliter l'évaluation de notre approche tels que :

- Lors de l'indexation de la collection, l'algorithme de Porter **[Por 80]** est utilisé pour la lemmatisation des termes.

- Une liste de mots vides est aussi consultée pour éliminer les termes non significatifs, qui n'apportent pas ou peu de sens au contenu des éléments, comme par exemple les pronoms ou les déterminants.

- Ajouter plusieurs champs nécessaires pour notre approche tels que, *ief* de l'élément idf, nombre de terme dans l'élément…etc.

La collection est indexée selon la stratégie d'indexation en sous arbre imbriqué (section 2.4.1.1). Cette stratégie est adoptée, parce que notre approche se base sur la propagation des termes, et les sources de pertinences utilisées sont calculées au niveau de chaque élément.
Les index sont stockés sous forme de tables dans une base de données relationnelle postgreSQL. La base de données contient quatre tables principales : la table Documents,

[6] https://github.com/ymoreau/Restad
http://restad.sourceforge.net

la table éléments, la table termes et la table Index_Inverse. Le schéma de ces tables est détaillé dans le tableau suivant :

Table	Description
Documents	***Documents (id_doc int, doc_name varchar(256),text text)*** ***Id_doc*** est l'identifiant unique de chaque document ***doc_name*** est le nom de fichier du document, ***text*** est le text du document sans les balises.
Termes	***Termes(id_token int, token varchar(256), ief , idf)*** ***id_token*** est l'identifiant unique de chaque terme. ***Token*** est le terme. ***ief*** est le ief terme. ***idf*** est le idf du terme
Eléments	***Elémént (id_tag bigint, id_doc int, id_tag_name int, nb_token, tag_num int, parent int, start int, end int);*** ***id_tag*** est l'identifiant unique de chaque élément. ***id_doc*** est l'identifiant du document contenant l'élément. ***Nb_token*** est le nombre de terme dans l'élément. ***tag_num*** est le numéro de l'élément dans l'élément parent (s'incrémente par type d'élément) ***parent*** est l'identifiant de l'élément parent. ***Start*** la position du début de l'élément ***End*** la position de la fin de l'élément
Index_Inverse	***Index_Inverse (id_token int, id_doc int, positions int[])*** ***id_token*** est l'identifiant unique de chaque terme. ***id_doc*** est l'identifiant du document contenant le terme. ***positions*** les positions du terme dans le document.

Tableau 4.4 : Les tables principales de la base de données.

4.5.1.2. Traitement des requêtes

Dans le cadre d'INEX, deux types de requêtes sont introduites:

- les requête de type CAS (*Content And Structure*) qui porte sur la structure (type d'élément) et le contenue des éléments à rechercher.
- les requêtes de type CO (Content Only) qui décrivent uniquement le contenu souhaité des éléments XML recherchés.

Les requêtes CO+S peuvent servir aux deux types CO et CAS. Il y a 87 requêtes dont 29 sont des requêtes de type CO. Par exemple, la requête 202 (figure 4.4) recherche les éléments contenant les mots clés suivants : **ontologies, case et study**

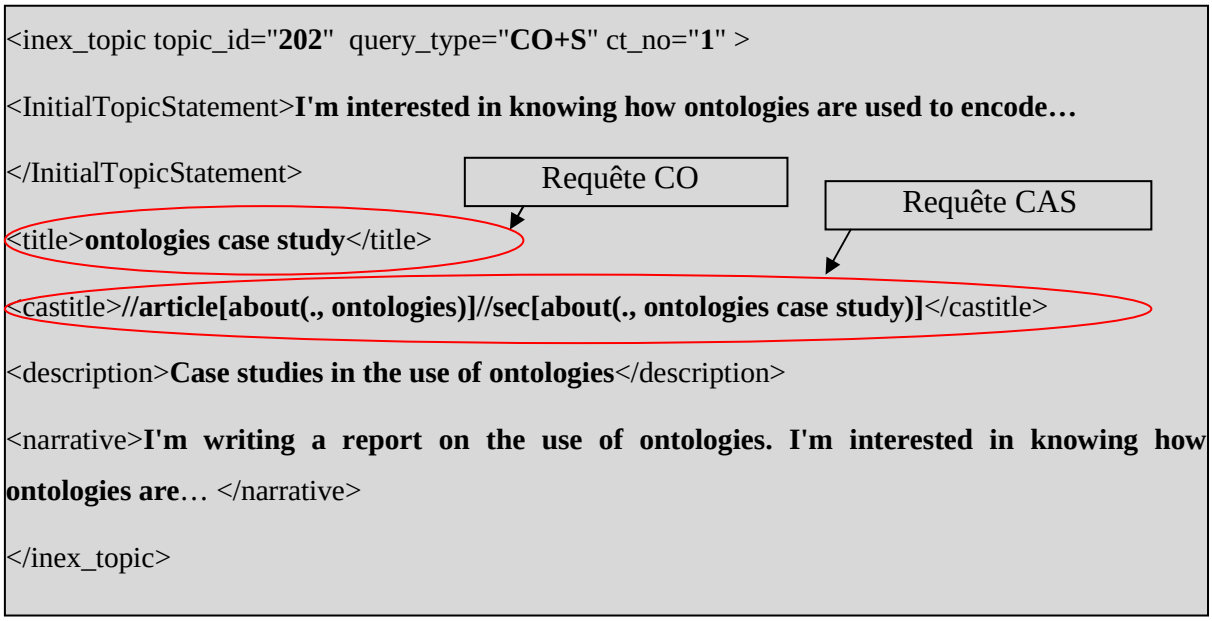

Figure 4.4 : Requête de type CO+S.

4.5.2. Echantillonnage des requêtes d'apprentissage

Nous étudions dans notre approche la tâche thorough d'INEX (chevauchement des éléments est permis), c'est-à-dire on peut renvoyer à l'utilisateur deux éléments imbriqués (par exemple, un paragraphe et sa section). Les 29 requêtes de type CO, qui ont des jugements de pertinence, numérotées de 202 à 241 sont utilisées.

Afin de rendre l'évaluation plus complète, nous avons utilisé dans notre expérience le « k-fold cross validation » de la validation croisée, avec k=3. L'ensemble de requête est divisé en trois parties avec environ le même nombre de requêtes, noté S1, S2 et S3. Pour chaque étape, deux parties sont utilisés pour l'apprentissage du modèle d'ordonnancement et la partie restante pour évaluer la performance du modèle appris (voir tableau 4.5). La

moyenne des résultats des trois différentes expériences de la validation croisée est finalement utilisée comme résultat de l'évaluation de notre approche.

Partie	Apprentissage	Test
Partie1	S1, S2	S3
Partie2	S1, S3	S2
Partie3	S2, S3	S1

Tableau 4.5 : Partitionnement des requêtes pour trois-fold cross validation.

4.5.3. Echantillonnage des éléments d'apprentissage

En raison du nombre important des éléments XML dans la collection (11 millions d'élément), il n'est pas possible d'extraire les vecteurs de caractéristiques et étiqueter tous ces éléments. Une stratégie raisonnable consiste à échantillonner certains éléments qui pourraient être plus pertinents, puis extraire les vecteurs de caractéristiques et faire l'étiquetage des couples élément-requête correspondants. Dans notre expérience, tout d'abord, le modèle BM25 est utilisé pour ordonner tous les éléments de chaque requête. Les 1500 premiers éléments de chaque requête ont été sélectionnés pour l'extraction de caractéristiques et l'étiquetage.

4.5.4. Etiquetage des couples élément-requêtes

L'objectif de l'étiquetage de données est d'associer à chaque couple élément-requête une étiquette, qui décrit le niveau de pertinence de l'élément par rapport à la requête. Deux méthodes d'étiquetage ont été utilisées dans le domaine de l'apprentissage d'ordonnancement. Dans la première, les jugements sont déterminés par des experts humains, dans la deuxième ils sont estimés à partir des clics des utilisateurs sur les documents. Nous avons utilisé la première méthode dans notre approche.

Dans la collection que nous allons utiliser pour l'apprentissage et le test, les jugements de pertinence sont définis, par des experts humains (participants de la compagne INEX), suivant deux dimensions : l'exhaustivité et la spécificité **[Chi 96]**. L'exhaustivité tient compte de la présence ou de l'absence de l'information recherchée dans un élément XML, elle est mesurée en utilisant 4 niveaux : très exhaustive (e = 2), peu exhaustive (e = 1), pas exhaustive (e = 0) et "trop petit" (e =?). La spécificité traduit le degré avec lequel un élément traite toute l'information recherchée, elle est mesurée dans un intervalle continu [0,1] où s=1

représente un élément totalement spécifique. Plusieurs fonctions de quantification sont appliquées pour agréger les deux dimensions, exhaustivité et spécificité, en une seule valeur unique qui représente le degré de pertinence des éléments.

Pour assigner chaque élément avec une étiquette nous avons utilisé la fonction d'agrégation "généralisée" $f_{gen}(e,s) = s * e$. Cette fonction donne une valeur réelle entre 0 et 2.

4.5.4.1. Problème d'étiquetage

L'étiquetage direct avec la fonction généralisée associe à chaque couple élément-requête une étiquette différente des autres, ce qui donne un nombre très grand des étiquettes.

Dans notre approche nous allons sélectionner 1500 éléments pour chaque requête. Si *K* éléments de cette ensemble sont jugés pertinents et chaque élément a une étiquette différente des autres, alors le nombre des étiquettes est égale à *K* et le nombre des paires d'élément pour chaque requête est :

$$Nb\_paires\_par\_req = C_k^2 = \frac{k!}{2!(k-2)!} \quad (4.10)$$

Ce nombre très grand des paires d'éléments peut influencer le temps d'exécution et la performance de l'algorithme d'apprentissage. Par exemple, si deux éléments qui ont deux jugements de pertinence très proche avec deux étiquette différente, l'algorithme ne converge pas facilement et tente inutilement de trouver une fonction qui ordonne correctement ces deux éléments, portant l'ordre dans ce cas n'est pas important (pertinence très proche).

4.5.4.2. Solution proposée

Pour éviter ce problème nous avons proposé d'utiliser quatre niveaux de jugement de pertinence et d'associer aux éléments qui ont des jugements très proches la même étiquette.

Les quatre niveaux de jugement de pertinence utilisés sont les suivants :

Niveau 1(excellent) : les éléments qui ont une exhaustivité égale à 2 et plus de 75% de leur texte est pertinent (spécificité >= 0.75).

Niveau 2 (bon) : les éléments qui ont une exhaustivité égale à 2 mais 50% à 75% uniquement de leur texte est pertinent (0.5 <= spécificité < 0.75).

Niveau 3 (Moyen) : les éléments qui ont une exhaustivité égale à 2 et moins de 50% de leur texte est pertinent (spécificité < =0.5) ou tous les éléments qui ont une exhaustivité égale à 1.

Niveau 4 (Mauvais) : les éléments qui ne contiennent aucune information pertinente (spécificité=0) ou jugé comme trop petit (exhaustivité = ?).

Le tableau 4.6 montre les étiquettes avec leurs degrés de pertinence correspondant.

$f_{gen}(e,s) = s * e$	Etiquette	Degré de pertinence
$1.5 \leq f_{gen}(e,s) \leq 2$	3	excellent
$1 \leq f_{gen}(e,s) < 1.5$	2	Bon
$0 < f_{gen}(e,s) < 1$	1	Moyen
$f_{gen}(e,s) = 0$	0	Mauvais

Tableau 4.6 : Etiquetage des couples élément-requêtes.

4.5.5. Apprentissage et tests

Après la construction de l'ensemble d'apprentissage en suivant les étapes précédentes, le résultat est un fichier contenant les requêtes, les éléments correspondants et leurs étiquettes comme montre la figure 4.5.

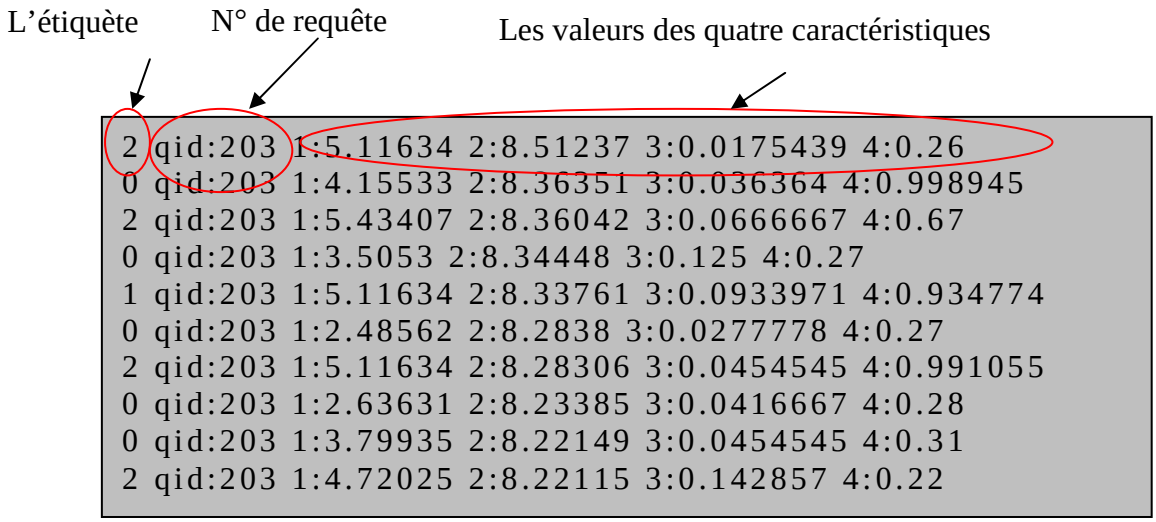

Figure 4. 5 : Fichier d'apprentissage *train.txt*.

L'algorithme d'ordonnancement Ranking SVM est utilisé pour apprendre le modèle de Ranking en utilisant comme entrée le fichier *train.txt*. Le résultat est un fichier *model.txt* qui contient les poids des caractéristiques.

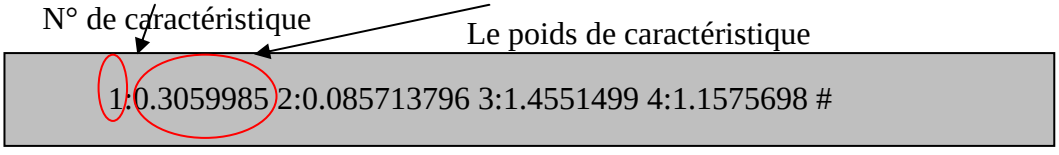

Figure 4. 6 : Fichier résultat de l'apprentissage *model.txt*.

Une fois la phase d'apprentissage est terminée, un ensemble de test, qui contient des requêtes inconnues pour la méthode d'apprentissage, sert à évaluer les performances en généralisation du modèle appris.

La préparation des données de test se fait de la même manière que dans la phase d'apprentissage sauf l'étape de l'étiquetage. Dans ce cas la, le modèle appris est appliqué pour associer à chaque couple élément-requête un score, trier les éléments en décroissance par rapport au score et retourner la liste ordonnée correspondante pour toutes les requêtes de test.

4.5.6. Evaluation des résultats

L'évaluation de notre approche se fait à travers deux mesures : la courbe d'effort-précision par rapport au gain-rappel et le gain cumulé étendu normalisé (nxCG). Ce dernier est le gain cumulé par rapport à un seuil, décrit à la section 2.5.3 du chapitre2. L'évaluation des résultats a été faite par l'outil Evalj[7].

Afin d'évaluer la performance de notre approche et mesurer l'impact de chaque caractéristique, plusieurs expérimentations ont été réalisées.

4.5.6.1. Combinaison de toutes les caractéristiques

Les figures 4.7 et 4.8 montrent les résultats obtenus sur les mesure Maep et nxCG de notre approche en combinant toutes les caractéristiques (BM25Elem, BM25Doc, ProxElem, et SizeElem). Ces résultats sont comparés avec toutes les expérimentations officielles relatives aux requêtes de type CO, soumises par l'ensemble des participants à INEX 2005, pour la tache thorough. On peut voir par cette mesure que les performances de notre approche sont mieux par rapport aux autres expérimentations, en particulier, au début de la liste de classement. Un bon système de recherche d'information structurée positionne les éléments les plus pertinents au début de la liste.

[7] Projet JAVA dans lequel toutes les mesures utilisées durant INEX 2005 ont été implémentées par Benjamin Piwowarski. Cette application peut être téléchargée à partir de l'adresse : https://sourceforge.net/projects/evalj

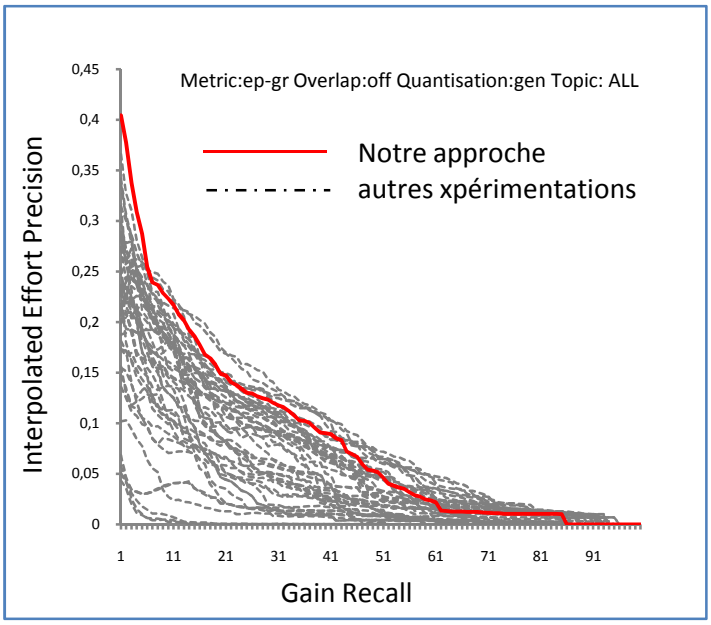

Figure 4. 7 : Courbe de ep/gr de notre approche et des résultats officiels de la campagne d'évaluation INEX 2005, tâche CO, quantification fgen.

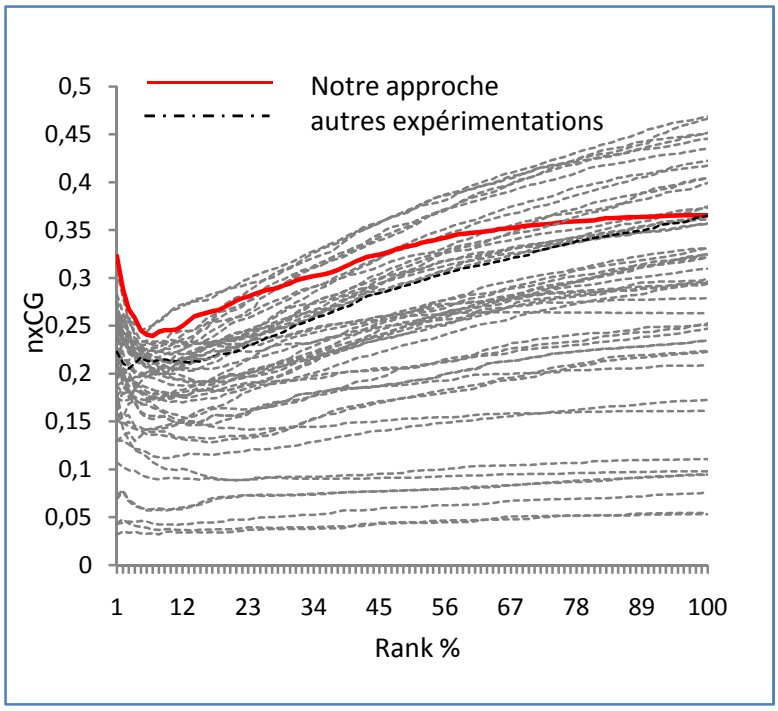

Figure 4. 8 : Comparaison avec les participants officiels d'INEX'2005 selon la mesure nxCG, tâche CO, quantification fgen.

Le tableau 4.7 montre les résultats obtenus sur la mesure ManXCG (*Mean average normalized extended cumulated gain*) de notre approche, BM25 (modèle de base) et les expérimentations officielles de l'université de Kaiserslautern et du laboratoire informatique de Paris 6, Lip6 **[Vit 06]**, qui sont les meilleures expérimentations soumises à la compagnie

INEX 2005, pour la tâche thorough. Pour cette mesure, notre approche est plus performante que ces expérimentations.

Nom de l'expérimentation	MAnxCG[r]									MAep
	1	2	3	4	5	10	15	50	100	
Notre Approche	**0.3895**	**0.3755**	**0.3611**	**0.3496**	**0.3439**	**0.3317**	**0.3228**	**0.3104**	**0.2899**	**0.0838**
Univ de Kaiserslautern	0.3073	0.3209	0.3293	**0.3282**	**0.3245**	**0.3124**	**0.3083**	**0.2986**	**0.2772**	0.0518
Lip6	**0.3845**	**0.3572**	**0.3401**	0.3253	0.3156	0.2874	0.2755	0.2624	0.2425	0.0756
BM25 (Modèle de Base)	0.2737	0.2560	0.2461	0.2399	0.2372	0.2286	0.2229	0.2187	0.2128	0.0560

Tableau 4.7 : Evaluation des résultats de notre Approche, BM25 et les deux meilleures expérimentations soumises à la compagnie INEX 2005 sur la mesure MAnxCG.

Le tableau montre aussi que les résultats du laboratoire Lip6 sont mieux par rapport aux résultats de l'université de Kaiserslautern pour ManXCG@1~3 (la performance du système pour les trois premiers éléments trouvés), mais pour ManXCG@4~100 les résultats de l'université de Kaiserslautern sont les mieux. Par contre, notre approche donne des bons résultats pour tout ManXCG@1~100.

4.5.6.2. Etude de l'impact de chaque caractéristique

Pour mesurer l'impact de chaque caractéristique sur la performance de notre approche. Nous avons évalué les résultats obtenus en enlevant à chaque fois une caractéristique et en combinant les trois restantes. Le tableau 4.8 montre les résultats trouvés pour chaque combinaison.

D'après les résultats, nous constatons que la meilleure performance de l'approche est obtenue en combinant toutes les caractéristiques. Les résultats montrent clairement que la suppression du BM25Elem ou SizeElem donne des mauvais résultats (par exemple MAnxCG@1 a démunie de 0,3895 à 0,2239 (-42.52%) lorsque on supprime BM25Elem et à 0,2200 (-43.52%) lorsque on supprime SizeElem). Alors l'impact de ces deux caractéristiques est très élevé par rapport à celui de BM25Doc et ProxElem. Il apparaît également que l'impact de BM25Elem est très fort sur tous les résultats trouvés MAnxCG@1~100 (-33.74% pour MAnxCG@1, -33.74% pour MAnxCG@2 et -29.51% pour

MAnxCG@50) Alors que l'impact de SizeElem est très fort sur les cinq premiers résultats uniquement MAnxCG@1~5 mais l'impact est moyennement fort pour MAnxCG@10~100 (-8.07% uniquement pour MAnxCG@100) . Il est clair aussi que l'impact de ProxElem est important comparativement à l'impact de BM25Doc.

Nous avons constaté aussi que toutes les caractéristiques utilisées ont un impact très élevés sur les premiers résultats de la liste (42.52 % pour BM25Elem, 43.52% pour SizeElem, 26.44% pour ProxElem et 16.61% pour Bm25Doc). Ca montre que nous avons choisi les bonnes caractéristiques et que notre approche a abouti aux objectifs et résultats attendus.

Le classement des quatre caractéristiques selon leurs impacts à partir des résultats de l'approche est comme suit :

1- Le score BM25 de l'élément(BM25Elem)
2- La taille de l'élément (SizeElem)
3- La proximité des termes de la requête dans l'élément (ProxElem).
4- Le score du document contenant l'élément (BM25Doc)

Caractéristique enlevée	MAnxCG[r]								
	1	2	3	4	5	10	15	50	100
BM25Elem	0.2239	0.2488	0.2479	0.2459	0.2457	0.2352	0.2273	0.2188	0.2106
	-42.52%	-33.74%	**-31.35%**	**-29.66%**	**-28.55%**	**-29.09%**	**-29.58%**	**-29.51%**	**-27.35%**
SizeElem	0.22	0.2297	0.2482	0.2526	0.2552	0.2632	0.2653	0.2701	0.2665
	-43.52%	**-38.83%**	-31.27%	-27.75%	-25.79%	-20.65%	-17.81%	-12.98%	-8.07%
ProxElem	0.2865	0.2964	0.3011	0.2983	0.2986	0.2922	0.2868	0.282	0.2658
	-26.44%	-21.07%	-16.62%	-14.67%	-13.17%	-11.91%	-11.15%	-9.15%	-8.31%
BM25Doc	0.3248	0.3284	0.3311	0.3215	0.3181	0.3151	0.3099	0.3034	0.2799
	-16.61%	-12.54%	-8.31%	-8.04%	-7.50%	-5.00%	-4.00%	-2.26%	-3.45%
Aucun	**0.3895**	**0.3755**	**0.3611**	**0.3496**	**0.3439**	**0.3317**	**0.3228**	**0.3104**	**0.2899**

Tableau 4.8 : Résultats montre l'impact de chaque caractéristique.

Ces résultats confirment les résultats d'études précédentes qui ont montré que l'utilisation de plusieurs sources de pertinence améliore la recherche d'information et justifie l'utilisation des techniques d'apprentissage d'ordonnancement.

4.6. Conclusion

Dans cette expérimentation, notre objectif était de combiner plusieurs caractéristiques (sources de pertinence) afin d'obtenir de meilleures performances du système de recherche et surtout de mieux répondre aux besoins de l'utilisateur. Nous avons proposé une approche qui permet de combiner certaine caractéristiques en utilisant les méthodes d'apprentissage d'ordonnancement.

Les principales conclusions que l'on peut tirer de toutes les expérimentations sont les suivantes :

- L'utilisation de l'apprentissage des fonctions d'ordonnancement avec le bon choix des caractéristiques, nous a permis d'observer des améliorations significatives pour la recherche d'information structurée.

- Une deuxième conclusion est celle qui concerne la combinaison des caractéristiques. Nous avons remarqué que la combinaison de toutes les caractéristiques proposées dans notre approche donne des meilleurs résultats. Ainsi que l'impact de certaines caractéristiques est plus important que d'autres.

- La multiplication et la diversification des sources de pertinence en recherche d'information dans les documents structurées justifie et motive en grande partie l'utilisation des méthodes d'apprentissage d'ordonnancement.

CONCLUSION GÉNÉRALE

Synthèse

Les travaux que nous avons présentés dans ce mémoire s'inscrivent dans le cadre de la recherche d'information, et plus particulièrement dans le cadre de la recherche d'information structurée. Nous avons exposé le problème de recherche d'information dans des documents structurés et nous avons évoqué la nécessité de développer des systèmes, efficaces et efficients, permettant d'identifier les éléments les plus pertinents dans un document par rapport à une requête donnée. Afin de faciliter la recherche d'information d'un utilisateur, ces techniques doivent renvoyer à l'utilisateur une liste ordonnée des éléments pertinents dans laquelle les plus pertinents doivent être classés au dessus des moins pertinents. Pour ce faire, des solutions pour l'interrogation des documents ainsi que le tri des unités d'information résultats ont été proposées.

Tous Les modèles d'interrogation proposés définissent une fonction de score qui calcul la pertinence entre la requête et l'élément XML. La plus part de ces fonctions combinent plusieurs sources de pertinence et le poids de chaque source est trouvé manuellement d'une manière empirique. Ces derrières années et avec l'apparition de l'apprentissage d'ordonnancement, les modèles sont appris automatiquement en évitant le paramétrage manuelle.

Nous avons introduit les trois grandes approches pour l'apprentissage d'ordonnancement, l'approche pointwise, pairwise et listwise. Nous avons introduit quelques algorithmes représentatifs pour chaque approche, en discutant leurs avantages et inconvénients, et l'évaluation de leur efficacité sur la collection de référence LETOR. Nous avons constaté que les algorithmes pairwise et listwise sont plus performants que les approches pointwise. Car ces dernières visent directement l'objectif de la RI qui est la prédiction de l'ordre entre les documents et pas le score de chaque document.

Résumé de la contribution

L'objectif de notre travail est de proposer une adaptation d'un algorithme d'apprentissage d'ordonnancement dans le cadre de la recherche d'information structurée. Pour ce faire il faut définir les caractéristiques à utiliser, les éléments et les requêtes pour l'apprentissage et la méthode pour l'étiquetage des données.

Pour le choix des caractéristiques nous avons choisi quatre caractéristiques (sources de pertinence) sur la base de leurs efficacités en RIS : Le score BM25 de l'élément, le score BM25 du document, la proximité des termes de la requête dans l'élément et enfin la taille de l'élément. En ce qui concerne les éléments d'apprentissage, nous avons sélectionnée les 1500 premiers éléments ordonnés par la fonction BM25. La technique de validation croisée *k-fold cross-validation* a été utilisée pour designer l'ensemble des requêtes d'apprentissage et l'ensemble des requêtes de tests. Pour l'étiquetage de données, nous avons utilisé la fonction généralisée qui combine l'exhaustivité et la spécificité de l'élément XML afin d'attribuer une étiquette à chaque couple élément-requête sélectionné. Enfin l'algorithme Ranking SVM est utilisé pour l'apprentissage.

La collection d'apprentissage et de tests est INEX 2005 constituée de 16819 documents de revues de IEEE, pour un total de 11 million d'éléments. Après l'évaluation de notre approche, les résultats obtenus ont été comparés aux soumissions officielles de la campagne INEX 2005, et présentent des performances intéressantes comparées à celles des meilleurs participants.

Nous avons constaté que la combinaison des quatre caractéristiques donne de bons résultats par rapport aux autres combinaisons. Ainsi, chaque caractéristique a son importance. Les résultats montrent aussi que l'impact du score de l'élément et la taille de l'élément dans la fonction de score est plus élevé comparativement à l'impact du score du document et de la proximité des termes de la requête dans l'élément.

En résumé, nous avons montré grâce à notre proposition et évaluation que l'utilisation des méthodes d'apprentissage d'ordonnancement avec le bon choix des caractéristiques améliore significativement les performances de la recherche d'information structurée.

Perspectives

Les perspectives à ces travaux sont nombreuses :

- Nous souhaitons pour les futurs travaux, d'étendre l'évaluation de notre approche sur une grande collection, telle que la collection qui contient plus de 650000 articles de l'encyclopédie Wikipedia **[Den 06],** environ 4,5 Go.
- L'utilisation d'autres caractéristiques liées à la structure et le contenue des éléments XML qui peuvent améliorer la RIS. Par exemple, le score des éléments voisins de l'élément XML est une caractéristique qui peut mesurer la spécificité de l'élément par rapport à la requête.

- Dans le cas où il y a un grand nombre de caractéristiques, il est intéressant de développer des méthodes automatiques pour objectif de sélectionner les meilleures caractéristiques avant de passer à la phase d'apprentissage.
- Enfin, une perspective très importante est de développer des méthodes d'apprentissage d'ordonnancement, spécialement, dédiées à la RIS qui prennent en compte la structure des éléments XML et les métrique d'évaluation dans leurs fonctions d'erreur.

BIBLIOGRAPHIE

[Abb 08] Abbaci, F., Francq, P.: XML Components Ranking: Which Relevant Ranking Criteria? Which Relevant Criteria Merging? First IEEE International Conference on the Applications of Digital Information and Web Technologies (ICADIWT'08), Ostrava, Czech Republic.

[Ada 74] Adamson, G., Boreham, J.: The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information Storage and Retrieval, vol 10, P253-60 (1974).

[Ali 07] Alilaouar, A. Interrogation flexible de documents semi-structurés. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, octobre 2007.

[And 01] Anderson, J.D., Perez-Carballo, J. The nature of indexing: How humans and machines analyze messages and texts for retrieval. Part 1&2. (Research and the nature of human indexing) Information processing and management 37 (2): (2001) 231-277.

[Aou 09] Aouicha, M. B. Une approche algébrique pour la recherche d'information structurée. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, janvier 2009.

[Arv 05] Arvola,P., Junkkari,M., Kekäläinen,J. "Generalized contextualization method for XML information retrieval," in Proc. CIKM, 2005, pp.20-27

[Ash 07] Ashoori, E., Lalmas, M., and Tsikrika, T. Examining topic shifts in contentoriented XML retrieval. International Journal on Digital Libraries 8, 1 (2007), 39–60.

[Bac 10] Baccini,V, Déjean,S., Kompaore, N. Mothe,J. Analyse des critères d'évaluation des systèmes de recherche d'information. Technique et Science Informatiques, Hermès Science Publications, 2010.

[Bei 07] Beigbeder, M.: ENSM-SE at INEX 2007: Scoring with proximity. ; in proc. INEX, 2007, pp. 53–55.

[Bou 08] Boughanem M., J. Savoy, « Recherche d'information : état des lieux et perspectives », Hermés, Paris, 2008.

[Bro 07] Broschart, A., Schenkel, R., Theobald, M., and Weikum, G. TopX @ INEX 2007. In Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, Selected Papers (2008).

BIBLIOGRAPHIE

[Bro 08] Broschart, A., Schenkel, R., Theobald, M.: Experiments with Proximity-Aware Scoring for XML Retrieval at INEX 2008. ;In INEX(2008)29-32.

[Bur 05] Burges, C.J.C., Shaked, T., Renshaw, E., Lazier A., Deeds, M., Hamilton, N., Hullender G., Learning to rank using gradient descent, Proceedings of the 22nd International Conference on Machine Learning (ICML), 2005, p. 89-96.

[Cao 07] CAO Z., QIN T., LIU T.Y., TSAI M.F., LI H., Learning to rank : From pairwise approach to listwise approach, Proceedings of the 24th International Conference on Machine Learning (ICML), 2007, p. 129-136.

[Cha 12] Chaa M., Nouali O., Bal K:"Learning to Rank in XML Information Retrieval: Which Feature Improve the Best?" in: proc. icdim, Macau, China, pp. 336–340 (2012)

[Chi 96] Chiaramella Y., Mulhem P., Fourel F.: A Model for Multimedia Information Retrieval, 1996,. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow

[Cle 66] Cleverdon, C. W., Mills, J., Keen, E. M., « Factors determining the performance of indexing systems », Cranfield, UK: Aslib Cranfield Research Project, College of Aeronautics (Volume 1: Design; Volume 2: Results), 1966.

[Coo 92] Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1992), pp. 198–210 (1992)

[Cra 02] Crammer, K., Singer, Y.: Pranking with ranking. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 641–647 (2002).

[Cos 06] Cossock D., Zhang T., Subset ranking using regression, Proceedings of the 19th Conference on Learning Theory (COLT), 2006, p. 605-619.

[Den 06] Denoyer, L., and Gallinari, P. The Wikipedia XML Corpus. SIGIR Forum 40, 1 (2006), 64–69.

[Flo 99] Florescu,D., Kossmann,D. Storing and querying XML data using an RDMBS. IEEE Data Engineering Bulletin, 22(3) :27–34, 1999

[Fra 92] Frakes, W. B. Stemming Algorithms, pages 131–160. Dans Frakes W B, Baeza- Yates, R. (eds) Prentice Hall, New jersey, 1992.

[Fre 03] Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4, 933–969 (2003)

[Fre 95] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1995)

BIBLIOGRAPHIE

[Fuh 02] N.Fuhr,N.,Govert,N., Kazai, G., Lalmas,M. editors. Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX'02),Schloss Dagstuhl, Germany,2002.

[Ful 93] Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structural answers for a large structured document collection. In Proceedings of ACM SIGIR 1993, Pitthsburgh, pages 204–213, 1993.

[Gao 11] Gao, N., Deng, Z., Yu, H., Jiang, J.: ListOPT: Learning to Optimize for XML Ranking. ;In PAKDD (2)(2011)482-492

[Gao 09] Gao, N., Deng, Z., Xiang, Y., Yu, H.: ListBM: A Learning-to-Rank Method for XML Keyword Search. ;In INEX(2009)81-87

[Gev 06] Geva, S. GPX - Gardens Point XML IR at INEX 2005. In Advances in XML Information Retrieval and Evaluation, 4th InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle,Germany, Revised Selected Papers (2006), pp. 240–253

[Gey 94] Gey, F.C.: Inferring probability of relevance using the method of logistic regression. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994), pp. 222–231 (1994)

[Gov 03] Gövert,N., Fuhr,N., Abolhassani, M. et Großjohann, K. Content-oriented XML retrievalwithHyREX. In Proceedings of the FirstWorkshop of the INitiative for the Evaluation of XML Retrieval (INEX), pages 26–32.

[Gra 02] Grabs, T., Sheck, H.J. : "ETH Zürich at INEX : Flexible Information Retrieval from XML with PowerDB-XML". In INEX 2002 Workshop Proceedings, p. 35-40, Germany, 2002.

[Gru 02] Grust,T. Accelerating XPath location steps. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA. In M. J. Franklin, B. Moon, and A. Ailamaki,editors, ACM Press. 2002.

[Has 05] HASSLER, M., BOUCHACHIA A., « Searching XML documents – preliminary work », p. 30-42, novembre 2005.

[Hat 05] Hatano, K., Kinutani, H., Amagasa, T.,Mori, Y., Yoshikawa,M. et Uemura, S. Analyzing the properties of XML fragments decomposed from the INEX document collection. In Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, volume 3493 de Lecture Notes in Computer Science, pages 168–182.

BIBLIOGRAPHIE

[Her 00] Herbrich, R., Obermayer, K., Graepel, T.: Large margin rank boundaries for ordinal regression. In: Advances in Large Margin Classifiers, pp. 115–132 (2000)

[Hie 98] Hiemstra, D. 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.

[Hua 07] Huang, S. Watt, D. Harper, M. Clark. Compact Representation in XML Retrieval. In Comparative Evaluation of XML Information Retrieval Systems (INEX 2006), pp. 65-72, 2007.

[Hub 10] Hubert G. : "Recherche d'information et contexte", Habilitation à diriger des recherches, Université Paul Sabatier - Toulouse 3, 2010.

[Jär 02] Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ;ACM Trans. Inf. Syst.(2002)422-446.

[Joa 02] Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), pp. 133–142 (2002).

[Kam 04] Kamps, J., Rijke, M.D., Sigurbjörnsson, B.: Length normalization in XML retrieval. ;In SIGIR(2004)80-87.

[Kam 08] Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., and Robertson, S. INEX 2007 Evaluation Metrics. In Focused access to XML documents, 6th InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, Selected Papers (2008).

[Kaz 07] Kazai, G. Choosing an Ideal Recall-Base for the Evaluation of the Focused Task: Sensitivity Analysis of the XCG Evaluation Measures. In Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval,INEX 2006, Dagstuhl Castle, Germany, Revised and Selected Papers (2007), pp. 35–44.

[Kek 05] Kekäläinen, J., Junkkari,M., Arvola, P. et Aalto, T. (2005). TRIX 2004 - struggling with the overlap. In Fuhr, N.,Lalmas, M.,Malik, S. et Szlávik, Z., éditeurs : Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers, volume 3493 de Lecture Notes in Computer Science, pages 127–139.

BIBLIOGRAPHIE

[Lal 06] Lalmas, M., Piwowarski, B.: INEX 2006 relevance assessment guide. In: INEX 2006 Pre- proceedings (2006).

[Lee 96] Lee, Y., Yoo, S., Yoon, K.. Index structures for structured documents. In ACM Workshop on XML and IR, Bethesda, pages 91–99. 1996.

[Liu 11] Liu, T.Y., Learning to Rank for Information Retrieval, Springer, 2011.

[Liu 07] Liu T.Y., Xu J., Qin T., Xiong W., Li H., LETOR : Benchmark dataset for research on learning to rank for information retrieval, Proceedings of SIGIR 2007 Worshop on Learning to Rank for Information Retrieval.

[Léa 12] Léa, L. Ordonnancement des résultats sur les moteurs de recherche : principes, limites et applications au géoréférencement (short paper). Dans VSST, Ajaccio, mai 2012.

[Lu 05] Lu, W., Robertson, S.E., MacFarlane, A. Field-Weighted XML Retrieval Based on BM25. ;In INEX(2005)161-171.

[Luc 59] Luce, R.D. Individual Choice Behavior. Wiley, New York (1959)

[Mal 75] Mallows, C.L. «Non-null ranking models». Biometrika 44, 114–130 (1975)

[Mar 60] Maron, M., Kuhns, J. On relevance, probabilistic indexing and information retrieval. Journal of the Association for Computing Machinery, 7 :pages 216–244, 1960.

[Mar 10] Marini, J.L. Capitalisation d'expériences pour l'indexation et la recherche d'information dans le domaine de la gestion électronique de documents. Thèse de doctorat, Université Jean Moulin Lyon 3, septembre 2010.

[Mas 02] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. Maarek, and A. Soffer. JuruXML-an XML retrieval system at INEX'02. In Proceedings of INEX 2002, Dagstuhl, Germany, pages 73–80, 2002.

[Mas 03] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML components. In Proceedings of INEX 2003, Dagstuhl, Germany, 2003.

[Nal 04] Nallapati, R.: Discriminative models for information retrieval. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004), pp. 64–71 (2004).

[Peh 06] Pehcevski, J., andThom,J. A. Hixeval: HighlightingXMLretrieval evaluation. In Advances in XML Information Retrieval and Evaluation, 4th InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers (2006), pp. 43–57.

BIBLIOGRAPHIE

[Pla 75] Plackett, R.L. (1975) «The analysis of permutations». Applied Statistics 24(2), 193–202.

[Pon 98] Ponte, J. M., and Croft,W. B. A language modeling approach to information retrieval. In SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia (1998), pp. 275–281.

[Por 80] Porter, M. F. An algorithm for suffix stripping. Program 14, 1980.

[Qia 10] Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, 2010.

[Rij 81] Rijsbergen, C. V. (1981). Retrieval effectiveness. In Information retrieval experiments. Butterworths, London, Karen Sparck-Jones.

[Rob 77] Robertson, S. The probability ranking principle in IR. Journal of Docu-mentation, 33(4): pages 294–304, 1977.

[Rob 94] Robertson, S. E., Walker, S. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Proceedings of SIGIR, pages 232–241, 1994.

[Rob 97] Robertson, S. E. Overview of the okapi projects, Journal of Documentation, Vol. 53, No. 1, 1997, pp. 3-7.

[Sal 70] Salton, G. The SMART retrieval system : Experiments in automatic document processing. Prentice Hall, 1970.

[Sal 71] Salton, G. A comparison between manual and automatic indexing methods. Journal of American Documentation, 20(1) :pages 61–71, 1971.

[SAL 83] Salton G., Mcgill M., « Introduction to Modern Information Retrieval. », New York : McGraw-Hill, , 1983.

[Sau 05] Sauvagnat, K. Modèle flexible pour la recherche d'information dans des corpus de documents semi-structurés. Thèse de doctorat, Université Paul Sabatier, juin 2005.

[Sau 06a] Sauvagnat, K., Hlaoua, L., and Boughanem, M. XFIRM at INEX 2005: Ad-Hoc and Relevance FeedbackTracks. In Advances in XML Information Retrieval and Evaluation, 4th InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers (2006), pp. 88–103.

[Sau 06b] Sauvagnat K., Boughanem M. Propositions pour la pondération des termes et l'évaluation de la pertinence des éléments en recherche d'information structurée. RevueI3 -Information Interaction Intelligence - Volume 6, n°2.2006.

BIBLIOGRAPHIE

[Sig 04] Sigurbjornsson, B., Kamps, J., and de Rijke, M. An Element-Based Approch to XML Retrieval. In Proceedings INEX 2003Workshop (2004), pp. 19–26.

[Sig 05] Sigurbjörnsson, B., Kamps, J. et de Rijke, M. (2005). Mixture models, overlap, and structural hints in XML element retrieval. In Advances in XML Information Retrieval, Third InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Revised Selected Papers, volume 3493 de Lecture Notes in Computer Science, pages 196–210.

[Sig 06] Sigurbjornsson, B.: Focused Information Access using XML Element Retrieval. SIKS dissertation series 2006-28, University of Amsterdam (2006)

[Sin 96] Singhal, A., Buckley, C., Mitra, M. Pivoted document length normalization. In SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pages 21–29. ACM Press, 1996.

[Ste 06] Stefan Büttcher, Charles L. A. Clarke, Brad Lushman: Term proximity scoring for ad-hoc retrieval on very large text collections. SIGIR 2006: 621-622.

[Tay 08] Taylor M., Guiver J., SoftRank : optimising non-smooth rank metrics, Proceedings of the 1st International Conference on Web Search and Data Mining, 2008, p. 77-86.

[The 06] Theobald, M., Schenkel, R., andWeikum, G. TopX and XXL at INEX 2005. In Advances in XML Information Retrieval and Evaluation, 4th InternationalWorkshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers (2006), pp. 282–295.

[Vap 95] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Berlin (1995)

[Vap 99] Vapnik V.N., Statistical learning theory, Wiley Inter Science, 1999.

[Vit 06] Vittaut, J., Gallinari, P.: Machine Learning Ranking for Structured Information Retrieval. ;In ECIR(2006)338-349.

[Wol 00] Wolff, J.E. , Flörke, H., Cremers, A.B., "Searching and browsing collections of structural information". In Proc of IEEE advances in digital libraries, pp. 141-150. Washington, 2000.

[Xia 08] Xia, F., Liu T.Y., Wang J., Zhang W., LI H., Listwise approach to Learning to rank – Theory and Algorithm, Proceedings of the 25th International Conference on Machine Learning (ICML), 2008.

BIBLIOGRAPHIE

[Xu 07] Xu J., Li H., AdaRank : A boosting algorithm for information retrieval, Proceedings of the 30th annual international SIGIR Conference on Research and Development in Information Retrieval, 2007, p.107-114.

[Yue 07] Yue Y., Finley T., Radlinski F., Joachims T., A support vector method for optimizing average precision, Proceedings of the 30th annual international SIGIR Conference on Research and Development in Information Retrieval, 2007, p. 271-278.

[Zar 04] Zargayouna, H. : Contexte et sémantique pour une indexation de documents semi-structurés. CORIA 04. pp. 161-177, 2004.

[Zip 49] Zipf, G. Human Behaviour and the Principle of Least Effort. Addison-Wesley, 1949.

[Zoe 08] Zoeter O., Taylor M., Snelson E., Guiver J., Craswell N., Szummer M., A decision-theoretic framework for ranking using implicit feedback, SIGIR Workshop on Learning to Rank for Information Retrieval, 2008.

RÉSUMÉ

L'adoption accrue de XML comme format standard pour représenter les documents structurés nécessite le développement des systèmes, efficients et efficaces, capable de retrouver les éléments XML pertinents à une requête d'utilisateur. Ces éléments sont ensuite présentés ordonnés en fonction de leur pertinence par rapport à la requête. Généralement la stratégie adoptée consiste à combiner plusieurs sources de pertinences dans une seule fonction de score et le poids de chaque source est donné manuellement selon des méthodes empiriques.

Il est connu que, dans la recherche d'information classique, compte tenu de plusieurs sources de pertinences et l'utilisation des méthodes d'apprentissage d'ordonnancement en combinant ces sources de pertinences, améliore la performance des systèmes de recherche d'information.

Dans ce travail, certains caractéristiques de pertinences des éléments XML, ont été définies et utilisées pour l'apprentissage d'ordonnancement dans les documents structurés. Notre objectif est de combiner ces caractéristiques afin d'obtenir la bonne fonction d'ordonnancement et montrer l'impact de chaque caractéristique dans la pertinence de l'élément XML.

Des expérimentations sur une grande collection de la compagne d'évaluation de la recherche d'information XML (INEX) ont montré la performance de notre approche.

Mots clés: XML, Recherche d'information structurée, Apprentissage d'ordonnancement, Ranking SVM, BM25.

ABSTRACT

The increased adoption of XML as the standard for representing a document structure requires the development of tools to retrieve and rank effectively relevant elements of the XML documents to a user's query. Usually the strategy adopted involves the combination of multiple sources of relevance into a single ranking and the weights of sources are manually tuned.

It's known that, in classical information retrieval, considering multiple sources of relevance and the use of learning to rank methods to combine theses sources of relevance improves information retrieval.

In this work, some relevance features of XML elements, are defined and used in a learning to rank approach for XML information retrieval. Our aim is to combine theses features to derive good ranking function and show the impact of each feature in the relevance of XML element.

Experiments on a large collection from the XML Information Retrieval evaluation campaign (INEX) showed good performance of the approach.

Keywords: XML, structured information retrieval, learning-to-rank, Ranking SVM, BM25.