THÈSE EN COTUTELLE

Présentée devant

l'Université Paul Sabatier de Toulouse

en vue de l'obtention du

Doctorat de l'Université Paul Sabatier

Spécialité : **INFORMATIQUE**

*P*ar

Mohamed BEN AOUICHA

Une approche algébrique pour la recherche d'information structurée

---

Soutenue le 08 Janvier 2009, devant le jury composé de :

M. M. BOUGHANEM	Professeur à l'Université Paul Sabatier, Toulouse III	Directeur de thèse
M. M. ABID	Professeur à l'Université de Sfax	Directeur de thèse
M. M. TMAR	Maître Assistant à l'Université de Sfax	Co-directeur de thèse
M. P. GALLINARI	Professeur à l'Université Pierre et Marie Curie, Paris VI	Rapporteur
M. C. CHRISMENT	Professeur à l'Université Paul Sabatier, Toulouse III	Examinateur
Mme. R. FAIZ	Maître de Conférences, HDR, à l'Université du 7 novembre à Carthage	Rapporteur

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

Résumé

L'objectif principal d'un SRI classique est de retrouver les documents dont le contenu est conforme à une requête donnée. Dans cette optique, les documents sont représentés par un ensemble de mots-clés décrivant leurs contenus. La structure du document n'est pas prise en considération ni au niveau de la requête, ni au niveau de la réponse pour retourner les parties pertinentes : la réponse à une requête reste le document tout entier. Aujourd'hui, l'utilisation de l'information apportée par la structure devient une nécessité dans le domaine d'accès à l'information. Cette nécessité provient d'un type de document qui est très bien répandu sur Internet, utilisé comme un standard d'échange sur le Web : le langage XML (*eXtensible Markup Langage*) qui est utilisé comme format de données structurées sur le Web, et qui impose au SRI de retrouver des unités d'information qui ne sont pas nécessairement le document entier.

L'appariement document/requête doit alors être réalisé d'une façon telle que les granules documentaires dont la structure présente de légères différences avec la structure de la requête reçoivent un score. Il peut également être vu comme l'inverse de l'effort nécessaire pour la construction incrémentale d'un arbre à partir d'un autre.

Grâce à la flexibilité apportée par la phase d'indexation, nous avons défini un algorithme basé sur le principe de relaxation des requêtes, qui permet de comparer les arbres requête et documents et de retourner les sous arbres potentiellement pertinents. Selon les sous arbres retournés à chaque document, nous avons défini une fonction de ressemblance entre la requête et le document. Cette fonction est une agrégation du score provenant de la structure et celui provenant du contenu des documents XML traités.

L'algorithme que nous proposons pour la comparaison d'arbres permet de localiser les sous arbres similaires à l'arbre représentant la requête. Ce premier appariement permet de fournir des scores orientés structure dans une échelle hautement graduée. En effet, l'extension que nous appliquons sur les représentations du document et de la requête et qui constituent le point de départ de notre algorithme de recherche augmentent la structure par des liens de descendance pondérés. L'algorithme de recherche de structures pertinentes est basé sur les valeurs de ses liens. Les structures sont plus pertinentes que les fragments sont plus larges, plus ramifiés et plus réels (si les poids des liens de descendance sont plus élevés).

Le traitement du contenu est réalisé indépendamment de la structure. La séparation entre les deux types d'appariement permet de mesurer l'impact de chacun. Nous nous sommes basés sur les modélisations arborescentes des documents XML. Cette modélisation permet de façon naturelle de traiter la structure et le contenu d'un document XML d'une manière indépendante. L'objectif principal de notre méthode est de combiner l'information portée par la structure et celle portée par le contenu.

On distingue dans la littérature deux principes courants, la plupart des modèles de RI utilisent la propagation des scores : on propage le score d'un nœud pertinent à une requête (en terme de contenu) à ses ancêtres. Les autres modèles se basent sur la propagation du contenu : au lieu de la propagation du score, on propage le contenu d'un nœud à un autre via les liens de descendance et on calcule son score indépendamment. Nous adoptons deux manières pour propager le texte d'un nœud feuille à ces ancêtres, la première se base sur la profondeur (le seul critère considéré dans la littérature), la deuxième se base sur la profondeur et la largeur (exprimé en fonction du nombre de fils) d'un document XML.

Nous nous sommes basés sur la séparation entre le traitement du contenu et de la structure. Cette séparation nous a permis de tirer profit des techniques de comparaison d'arbres pour le traitement de la structure, et des techniques de recherche d'information pour traiter le contenu. D'autre part, elle nous a été bénéfique pour mesurer l'apport de chaque orientation de recherche. Par ailleurs, et dans des conditions réelles d'utilisation, la séparation entre le contenu et la structure permet une utilisation plus libérée du système, en effet l'utilisateur pourra orienter sa recherche vers le contenu ou la structure selon son besoin en information.

Le traitement séparé du contenu et de la structure de chaque élément XML engendre deux scores : un score pour le contenu et un score pour la structure. Leur combinaison en un score définitif permet de les ordonner selon leur pertinence potentielle. Nous avons développé deux techniques pour la combinaison des scores : une technique basée sur une combinaison linéaire et une deuxième technique basée sur les distributions des scores.

L'évaluation de notre modèle, grâce au prototype que nous avons développé, montre l'intérêt de nos propositions.

Mots-clés : Recherche d'Information, XML, arbres étendus, propagation de texte

Remerciements

Ce n'est pas par tradition que cette page figure au préambule de ce mémoire, mais c'est plutôt dicté par un devoir moral et une reconnaissance sincère. Je serais en effet ingrat si je n'exprime pas ma reconnaissance et ma gratitude à tous ceux qui, de près ou de loin, m'ont facilité ma tâche et m'ont permis de mener à bien ce travail.

Je tiens tout particulièrement à exprimer toute ma gratitude et mes vifs remerciements à Monsieur Mohand Boughanem, Professeur à l'Université Paul Sabatier de Toulouse et à Monsieur Mohamed Abid, Professeur à l'Université de Sfax qui m'ont accordé leur confiance en me proposant ce sujet de thèse et pour leur aide et soutenance durant la réalisation de ce travail. Je les pris de croire à ma respectueuse estime et ma sincère reconnaissance pour leurs conseils qui m'ont été très précieux et fertiles.

Je tiens à exprimer ma très grande gratitude à Monsieur Mohamed Tmar, Maître assistant à l'Université de Sfax, pour avoir dirigé mes recherches. Ses conseils, ses critiques, sa ferme volenté de collaboration ainsi que la confiance qu'il m'a toujours témoigné m'ont été d'un grand apport tout au long de mes recherches. Qu'il soit assuré de mon très grand respect.

Je suis très honoré que Monsieur Patrick Gallinari, Professeur à l'Université Pierre et Marie Curie de Paris et Madame Rim Faiz, Maître de Conférences à l'Université 7 Novembre à Carthage, aient accepté de rapporter mon travail. Je tiens également à remercier Monsieur Claude Chrisment, Professeur à l'Université Paul Sabatier de Toulouse, pour l'honneur qu'il me fait en présidant ce jury.

C'est avec sympathie que je souhaite témoigner ma reconnaissance à Madame Karen Pinel-Sauvagnat, Maître de Conférences à l'Université Paul Sabatier de Toulouse pour la pertinence de ses remarques et ses conseils.

J'aimerais exprimer tous mes remerciments envers mes amis. Je pense particulièrement à Wadii Ben Mahmoud, Bassem Ben Hamed, Anis Jedidi, Faiza Ghozzi, Hamdi Gabsi, Mohamed Khazri, Khalil Chebil, Riadh Ben Halima, Mahdi Khemakhem, Mourad Kmimech, Moultazem Ghazal, Désiré kompaore, Asma Brini et Hamid Tebri.

J'aimerais exprimer aussi toute ma gratitude envers les membres de l'Unité de recherche Computer and Embedded Systems et l'équipe Systèmes d'Informations Général-

isés pour leur sympathie. Ils ont rendu très agréables ces quatre années.
Je voudrais exprimer aussi mon profond remerciement à tous mes enseignants de la Faculté des Sciences de Sfax et l'École Notionale d'Ingénieurs de Sfax, qui m'ont prodigué le savoir et m'ont permis d'arriver à ce stade, je pense particulièrement à Monsieur Adel Mahfoudhi et Monsieur Mohamed Jmaiel et à tous mes amis qui m'ont apporté leurs soutien pour la réalisation de ce mémoire dans d'excellentes conditions.

Table des Matières

Liste des tableaux

Table des figures

Introduction générale

Contexte du travail

La croissance intense du volume d'information disponible sur le Web a engendré de nombreux problèmes d'accès à l'information. Les outils permettant de faciliter l'accès personnalisé à l'information pertinente, et en particulier les systèmes de recherche d'information, doivent réagir en conséquence. Le problème auquel ils devraient faire face est le stockage de l'iformation, c'est-à-dire la gestion de l'index. Plusieurs techniques sont apparues pour parvenir à stocker l'information de manière à ce que son interrogation soit simple et rigoureuse. Dans ce contexte, c'est le volume de l'information qui est mis en exergue, sans aucune nécessité de faire évoluer les modèles traditionnels de recherche d'information.
Cependant, l'apparition de nouveaux formats de documents nécessite l'adaptation de ces modèles pour parvenir à retrouver l'information réellement pertinente. En effet, le contenu seul ne suffit pour décider de la pertinence du document s'il est structuré. Le document XML est au centre de cette problématique, la notion de structure est arrivée pour complèter le contenu du document et doit alors être prise en charge dans un processus de recherche d'information, désormais connue par le terme recherche d'information structurée.
Les techniques traditionnelles de recherche d'information n'ont traité la structure des documents comme caractère de pertinence que d'une manière marginale. Avec les nouveaux besoins imposées par la recherche d'information structurée, elles ne peuvent se dissocier des techniques de stockage et d'interrogation de données structurées qui font la notoriété des systèmes de gestion de bases de données. En revanche, les systèmes de gestion de bases de données ne traitent que la structure : toutes les données sont supposées être atomiques et par conséquent, aucun traitement du contenu n'est réalisé. Les communautés bases de données et recherche d'information se sont investies à cette problématique d'actualité. On retrouve alors des approches basées sur le contenu (document-centric) qui traitent essentiellement le contenu en tenant compte de la structure comme une dimension additionnelle, et des approches basées sur la structure (data-centric) qui privilégient la structure des documents et traitent le contenu par génération de requêtes sursemblables de SQL.

Selon la technique, il y a plus d'enthousiasme vers le contenu ou la structure. Le passage à l'échelle contrairement à toute attente, montre que la structure détériore la qualité de réponse. Les systèmes de recherche d'information actuels donnent de l'importance à la notion de structure, mais toujours d'une façon relativement marginale.

Problématiques

Un document XML est assimilé à un arbre où les nœuds représentent des balises XML et peuvent contenir des éléments de contenu. Contrairement à la recherche d'information classique, une requête peut être formulée en XML. L'accès aux documents XML consiste à retrouver tous les fragments de l'arbre relatif à la requête ayant un correspondant dans l'arbre relatif au document, en terme de contenu et de structure : un bon système de recherche d'information structurée doit prendre en considération ces deux aspects d'une manière flexible, autrement dit, la similarité en terme de contenu et/ou de structure doit être une mesure graduée.

Les langages d'interrogation XML classiques sont basés sur la comparaison exacte, ils ne permettent par conséquent que de fournir de résultats binaires, or en recherche d'information (classique ou structurée), on tente d'ordonner les granules documentaires selon leur pertinence potentielle.
L'appariement document/requête doit alors être réalisé d'une façon telle que les granules documentaires dont la structure présente de légères différences avec la structure de la requête reçoivent un score. Il peut également être vu comme l'inverse de l'effort nécessaire pour la construction incrémentale d'un arbre à partir d'un autre.
La comparaison d'arbres a fait l'objet de plusieurs initiatives. Elles sont toutes basées sur les opérations de mise à jour les plus basiques dont chacune est munie d'un coût. Le coût de construction est le coût cumulé de toutes les opérations nécessaires pour la construction. On part de la représentation d'un arbre et on le transforme en un autre avec le minimum d'effort possible. Pour des raisons de mise en situation expérimentale, les algorithmes de comparaison d'arbres de la littérature ne sont pas adaptés à la recherche d'information structurée.
Moins complexe que les algorithmes de comparaison d'arbres, la distance de Levenstein, initialement proposée pour mesurer la distance entre deux chaînes de caractères plutôt que les arbres, est une solution alternative, mais elle demeure coûteuse car elle se base sur la programmation dynamique qui est par essence coûteuse.

Le traitement du contenu est également problématique. En effet, le contenu textuel n'apparaît souvent que dans les nœuds feuilles. Cependant, un nœud interne peut être pertinent même s'il ne contient aucun terme d'indexation, la pertinence ne provient

pas seulement de la structure. Afin de rétablir l'ordre entre les nœuds pour que les feuilles ne soient pas prévilégiées par rapport aux nœuds internes, ceux-ci doivent avoir un score relatif au contenu. On distingue dans la littérature deux principes courants, la plupart des modèles de RI utilisent la propagation des scores : on propage le score d'un nœud pertinent à une requête (en terme de contenu) à ses ancêtres. Les autres modèles se basent sur la propagation du contenu : au lieu de la propagation du score, on propage le contenu d'un nœud à un autre via les liens de descendance et on calcule son score indépendamment.
Seule la distance entre les éléments XML est prise en considération lors de la propagation (profondeur), cependant, il existe bien d'autres critères relatifs à la structure (largeur de l'arbre, ou d'un fragment de l'arbre) qui pourront avoir un impact considérable sur la qualité de réponse du système.

Le traitement du contenu et celui de la structure sont les majeurs problèmes de la recherche d'infomation structurée. Il est indispensable d'obtenir des scores de pertinence dans une échelle hautement graduée. C'est à ce niveau que s'impose un autre problème : comment combiner les deux scores pour arriver à fournir une liste triée d'éléments XML. Autrement dit, chaque élément XML doit recevoir un score unique, qui dépend évidemment de sa pertinence selon la structure et le contenu, mais en fonction de quoi est-il exprimé et comment? Une combinaison linéaire semble être une solution, mais face à l'évaluation expérimentale, d'autres combinaisons plus complexes doivent être expérimentées, et plusieurs autres paramètres doivent être pris en compte. La problématique traditionnelle liée à l'évaluation de la pertinence d'une information vis-à-vis d'une requête reste d'actualité, mais elle se complique et implique d'autres questions dans le cadre des documents XML, notamment en ce qui concerne la structure. Les requêtes orientées contenu, qui sont de loin les plus simples pour l'utilisateur, imposent au système de recherche d'information de décider la granularité appropriée de l'information à renvoyer. Les unités d'informations devront être les plus *exhaustives* et *spécifiques* possibles par rapport à la requête. Contrairement à la recherche d'information traditionnelle, la pertinence dans le cadre de la recherche d'information structurée est en effet exprimée selon deux dimensions : l'exhaustivité et la spécificité. L'exhaustivité permet de mesurer à quel point l'unité d'information répond à la demande de l'utilisateur, et la spécificité permet de mesurer à quel point le contenu de l'unité d'information se focalise sur le besoin de l'utilisateur. Les modèles de recherche et de tri des unités d'information devraient donc prendre en compte ces deux dimensions de manière explicite, ce qui n'est pas forcément le cas des approches proposées dans la littérature, et notamment des approches orientées base de données. Dans le cadre des requêtes deux cas sont possibles. Dans le premier cas, l'utilisateur peut exprimer des conditions sur la structure des documents, mais ne pas préciser le type des unités d'information qu'il désire voir renvoyées par le système. Cette problématique, dans laquelle l'information structurelle peut être utilisée seulement comme une indication pour aider à retrouver l'information pertinente et non comme une indication de ce que souhaite l'utilisateur, n'a pas (ou peu) été abordée dans la littérature. Le deuxième

cas concerne les requêtes pour lesquelles le type de l'élément à renvoyer est spécifié par l'utilisateur. D'autres notions de pertinence entrent alors en jeu. La dimension de spécificité n'a plus réellement de sens, puisque l'utilisateur précise la granularité de l'information qu'il désire. Cependant, le contenu des éléments de structure ainsi que les expressions de chemin présentes dans la requête doivent pouvoir être traitées de manière vague. En d'autres termes, un degré de pertinence doit pouvoir être attribué aux éléments.

Contributions

Notre contribution dans le cadre de la recherche d'information structurée se situe à plusieurs niveaux et tente de répondre aux problématiques présentées dans la section précédente :

1. Notre première contribution porte sur le concept du traitement de la structure. Etant par essence une méthode d'appariement de complexité élevée, nous avons prévu de mettre en relief la technique de comparaison d'arbres très tôt dans le processus de recherche : c'est ce que nous avons baptisé *indexation de la structure*. Les algorithmes de comparaison d'arbres adoptent une recherche flexible sur les représentations exactes des arbres à comparer, nous proposons un algorithme de recherche exacte sur les représentations flexibles (ou étendues) des arbres à comparer. Ce choix est plus coûteux au niveau représentation (chaque arbre doit être étendu puis stocké dans un index) mais beaucoup moins coûteuse au niveau interrogation.

2. Nous traitons subséquemment mais indépendamment le contenu des éléments XML en faisant appel aux fondements de la recherche d'information traditionnelle par adaptation à la nature des documents semi-structurés. Nous avons proposé plusieurs techniques pour propager le texte se situant dans les nœuds feuilles vers leurs ancêtres, permettant de pondérer chaque terme dans ces derniers, en se basant sur les caractéristiques de l'arborescence d'un document XML. Les techniques de propagation actuelles se basent sur la profondeur, nous proposons de tester l'effet de la largeur d'un fragment XML sur la recherche structurée. Nous proposons par ailleurs une méthode statistique pour la classification des termes basée sur les NGrams pour la radicalisation des termes d'indexation.

3. Le traitement séparé du contenu et de la structure de chaque élément XML engendre deux scores : un score pour le contenu et un score pour la structure. Leur combinaison en un score définitif permet de les ordonner selon leur pertinence potentielle. Nous avons développé deux techniques pour la combinaison des scores : une technique basée sur une combinaison linéaire et une deuxième technique basée sur les distributions des scores.

4. Nous avons développé notre proposition dans un système qui regroupe les modules d'indexation des corpus XML et d'interrogation de la base. Nous avons optimisé notre système selon les exigences liées à la nature des corpus d'INEX et de la syntaxe des requêtes (le langage NEXI).

   Nous avons testé plusieurs formules de pondération orientées contenu et structure. Plusieurs paramètres ont été ajustés afin de retenir les valeurs qui offrent la performance idéale. Par ailleurs, plusieurs autres propositions ont été formalisées et intégrées afin d'assurer la flexibilité dans quasiment tous les niveaux de recherche. Toutes ces propositions ont été évaluées dans le cadre de la compagne d'évaluation INEX.

Organisation de la thèse

Ce mémoire est organisé en deux parties. L'objectif de la première partie **Recherche d'information et structure** est de présenter l'état de l'art de la recherche d'information traditionnelle et structurée. Dans la chapitre 1 intitulé **Recherche d'information**, nous présentons les approches dans la littérature concernant les fondements de base de la recherche d'information traditionnelle dans les documents texte (documents plats). Nous commençons tout d'abord par décrire le processus de recherche d'information, ensuite nous présentons les différents modèles d'appariement document-requête. Nous présentons ensuite les mesures utilisées pour l'évaluation des systèmes de recherche d'information. Enfin nous présentons brièvement la compagne TREC pour l'évaluation des différents systèmes de recherche d'information des documents plats.

Le chapitre 2 intitulé **Recherche d'information structurée** s'intéresse aux approches proposées dans la littérature pour interroger des corpus de documents XML. Nous commençons tout d'abord par donner une brève description de ce langage de balisage. Nous présentons ensuite les différentes techniques d'indexation de documents structurés, ainsi que les méthodes de gestion des documents XML. Afin de présenter quelques modèles adaptés à la recherche d'information structurée, nous nous focalisons sur des modèles proposés dans la littérature, qui visent à répondre à des requêtes basées sur le contenu et la structure. Enfin nous présentons les nouvelles mesures d'évaluation adoptées par la recherche d'information structurée dans le cadre de la campagne d'évaluation INEX permettant de mesurer la performance des systèmes de recherche d'information dans les documents XML.

La seconde partie de ce mémoire, intitulée **Un modèle de recherche d'information structurée**, présente nos travaux. Tout d'abord dans le chapitre 3 nommé **Modèle de recherche structurée basé sur la comparaison d'arbres étendus**, nous donnons un aperçu sur les méthodes de comparaison d'arbres et le mécanisme de relaxation des requêtes. Ensuite nous détaillons notre modèle de comparaison d'arbres

appliqué à la recherche d'information structurée. Ensuite nous définissons les fonctions d'appariement par la structure. Concernant le traitement du contenu, nous commençons par présenter une méthode de radicalisation des termes, cette méthode permet de remplacer l'algorithme de Porter qui permet de radicaliser les termes pour la langue anglaise. Nous définissons une fonction d'appariement par le contenu. Pour calculer le score final d'un nœud par rapport à une requête nous définissons deux méthodes pour agréger le score sur le contenu et la structure : en utilisant une combinaison linéaire et en se basant sur un calcul probabiliste.

Le dernier chapitre, nommé **Expérimentations et résultats**, a pour but d'évaluer nos différentes propositions décrites dans le chapitre 3. Ce chapitre consiste à évaluer notre système de recherche par le contenu pour les deux collections issues d'INEX 2004 et 2005. Nous décrivons les conditions expérimentales et les résultats obtenus pour des requêtes qui portent sur le contenu et la structure en mettant l'accent sur l'apport de notre méthode de recherche par la structure. Au cours de ce chapitre, nous dressons un bilan sur l'apport de chaque proposition du chapitre précédent.

Partie I

Recherche d'information et structure

Chapitre 1

Recherche d'information

1.1 Introduction

La Recherche d'Information (RI) s'intéresse à l'acquisition, l'organisation, la recherche et la sélection de l'information. La tâche principale d'un Système de Recherche d'Information (SRI) [100] est de sélectionner dans une collection de documents ceux qui sont susceptibles de répondre aux besoins en information de l'utilisateur. L'accès à ces documents peut s'effectuer de manière active à travers une requête, on parle alors de recherche active, ou bien de manière passive, en se basant sur le profil de l'utilisateur on parle alors d'un système de *filtrage d'information* [123] [124].
Dans ce mémoire, nous nous intéressons à la collecte active de l'information. 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.
Nous commençons tout d'abord par donner quelques définitions, puis nous décrivons les étapes d'un processus de RI. Nous passerons ensuite en revue les différents modèles de recherche. Enfin, nous présenterons les plus importantes mesures utilisées pour évaluer un SRI.

1.2 Concepts de base de la recherche d'information

Un SRI a pour rôle de sélectionner, à partir d'un besoin en information utilisateur, les documents qui peuvent l'intéresser, c'est-à-dire ceux qui peuvent être pertinents à son besoin en information. Cette définition fait apparaitre trois notions clés qu'il convient de préciser : documents, besoin en information, pertinence.

1.2.1 Du document à la base documentaire

Un document représente dans un SRI l'élément objectif [104], dans le cadre de la RI traditionnelle, c'est du texte libre, qui peut être caractérisé selon trois vues :

la vue *représentation*, c'est la mise en forme d'un document texte (entêtes, paragraphes, alignement...) ;

la vue *logique* qui présente la structure logique d'un document, elle porte des informations sur la structure ;

la vue de *contenu*, qui se focalise sur le sens ou la sémantique d'un document le plus souvent par un ensemble de mots.

Dans les SRI traditionnels, la vue contenu est l'unique intérêt, puisque les utilisateurs forment leurs requêtes en se mettant comme objectif le contenu textuel des documents [13] : c'est d'ailleurs la raison même de l'utilisation de tels systèmes.

1.2.2 Du besoin en information à la requête

Pour exprimer son besoin l'utilisateur compose une suite de mots-clés (requête), 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**...

Dans la RI traditionnelle, le besoin en information de l'utilisateur se compose d'un ensemble de mots-clés, c'est-à-dire que les informations ciblent le contenu textuel d'un document.

1.2.3 Pertinence

La pertinence est une notion fondamentale et cruciale dans le domaine de la RI. Elle est l'objet de tout système de recherche d'information. Définir cette notion complexe n'est pas simple, car elle fait intervenir plusieurs notions. Une définition simple pourrait être : *La pertinence est la correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête.* On s'est vite aperçu que la pertinence n'est pas une relation isolée entre un document et une requête. Elle fait appel aussi au contexte de jugement. Ainsi, les travaux de recherche [103],[71], [14] s'accordent sur la difficulté de la définition de la pertinence et mettent en exergue deux types de pertinence : la pertinence système et la pertinence utilisateur :

- la pertinence système [27] est déterministe, objective et elle est définie à travers les modèles de RI. Elle est souvent traduite par un score évaluant l'adéquation du contenu des documents vis-à-vis de celui de la requête ;
- la pertinence utilisateur [46], [105] [71] quant à elle, est liée à la perception de l'utilisateur sur l'information renvoyée par le système. Elle est subjective, deux utilisateurs peuvent juger différemment un même document renvoyé pour une même requête. Elle peut évoluer dans le temps d'une recherche. Une information jugée non pertinente à l'instant t pour une requête peut être jugée pertinente à $t+1$ car la connaissance de l'utilisateur sur le sujet à évolué.

1.3 Processus de RI

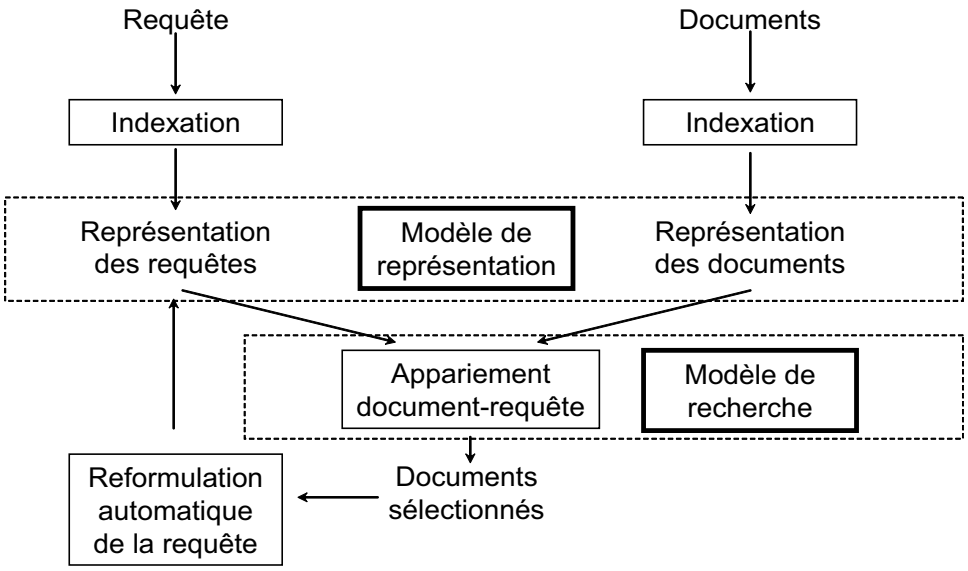
Pour répondre aux besoins en information de l'utilisateur, un SRI met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans un fonds documentaire d'une part, et les besoins en information des utilisateurs d'autre part. Un SRI est défini par ses modèles de représentation des documents, des besoins de l'utilisateur et sa fonction d'appariement.
Ce processus est composé de trois fonctions principales :

l'*indexation* des documents et des requêtes : l'indexation a pour rôle d'extraire à partir d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique.

l'*appariement requête-document* ou l'*interrogation* : la comparaison entre le document et la requête revient à calculer un score, supposé représenter la pertinence du document vis-à-vis de la requête.

la *reformulation de la requête* : qui intervient suite à une itération de recherche, et consiste à modifier les requêtes en fonction des résultats présentés et le jugement de l'utilisateur.

Le processus général d'un SRI consiste à représenter le besoin en information, et en parallèle de collecter des documents, de déterminer l'appariement entre chaque document et la requête puis de décider si le document est pertinent.
La reformulation de la requête est une procédure de base en RI. Au sens large, la reformulation de la requête peut renvoyer un dialogue interactif entre le système et l'utilisateur ce qui entraîne non seulement une demande adaptée mais aussi une meilleure compréhension par l'utilisateur de ses besoins en information comme illustre la figure 1.1 qui représente l'architecture en U d'un SRI [12] [100]. Selon la représentation in-



▶ Figure 1.1: Architecture en U d'un SRI

terne des requêtes et des documents, le SRI effectue un appariement afin de déterminer les documents potentiellement pertinents (pertinence système). Une fois l'appariement réalisé, le système sélectionne les documents les plus prometteurs[1] et les présentent à l'utilisateur. Sur la base de cette première réponse du système, l'utilisateur peut indiquer au système les documents qu'il juge réellement importants (pertinence utilisateur) et ceux qui ne lui présentent aucun intérêt. A l'aide de ces jugements, le système doit être capable de construire automatiquement une nouvelle requête (bouclage de requête ou reformulation par réinjection de pertinence) afin de présenter une nouvelle liste de référence [17] [89] [97]. Ce processus indique clairement que la RI doit toujours être

[1]Les documents les plus pertinents jugés par le SRI.

vue comme un processus itératif.
Mais le point de départ de tout système documentaire est l'organisation du fond documentaire. En effet, et dans des conditions telles que le système devra réagir assez rapidement à son utilisateur, l'organisation des documents collectés joue un rôle primordial à la robustesse du SRI.

1.3.1 L'indexation des documents et des requêtes

Dans un processus de RI le coût de recherche doit être acceptable, il convient alors de procéder à une phase primordiale pour optimiser le temps d'exécution de ce processus. Cette phase consiste à analyser chaque document de la collection, et de créer un ensemble de mots-clés, on parle alors de l'indexation. Le rôle de l'indexation est fournir une présentation synthétique du contenu du document, on distingue trois types d'indexation :

manuelle : la représentation du document se fait par un spécialiste (*documentaliste*) ;

automatique : le processus d'indexation est complètement informatisé ;

semi-automatique : le processus d'indexation se fait en premier lieu d'une manière automatique, le documentaliste intervient seulement pour ajouter des mots-clés qu'il trouve intéressants pour représenter un document.

L'indexation manuelle permet d'assurer une bonne représentation des documents, cependant elle présente un inconvénient majeur vu que la tâche du documentaliste se veut souvent influencée par son point de vue, or celui-ci est très subjectif : aucune règle universelle ne peut être appliquée à cette fin [83] [94]. L'indexation semi-automatique est un processus partiellement automatisé, le documentaliste intervient pour apporter les rectifications qu'il voit nécessaires [52]. L'indexation automatique est la plus communément utilisée. Ce type d'indexation regroupe un ensemble de traitements automatisés sur un document : l'extraction automatique des termes du document, l'élimination des mots vides, la lemmatisation ou la radicalisation des mots, la pondération et enfin la création de l'index.

1.3.1.1 Extraction des termes

L'analyse lexicale constitue la première étape du processus d'indexation. Sa fonctionnalité principale est de connaître une unité lexicale ou un radical [34]. La mission de

l'analyse lexicale est alors de transformer une suite de caractères en une suite de mots reconnaissables.

1.3.1.2 Elimination des mots vides

L'une des étapes dans le processus d'indexation permettant d'améliorer la fiabilité d'un SRI au sens de qualité logiciel (temps d'exécution) et de performance, est l'élimination des mots vides (pronoms personnels, prépositions...). Ce sont des mots ne traitent pas le sujet d'un document. On distingue deux techniques pour filtrer les mots vides :

- L'utilisation d'une liste prédéfinie de mots vides (aussi appelée *anti-dictionnaire* ou *stop-list*),
- Le comptage du nombre d'apparition d'un mot dans un document de la collection. On supprime les mots ayant une fréquence qui dépasse un certain seuil et qui deviennent alors vraisemblablement des mots vides.

L'élimination de ces mots permet de réduire l'index, on gagne alors en espace mémoire, mais aussi le non traitement des mots vides fait gagner un SRI en temps d'exécution. Par ailleurs et dans le cas où la recherche est trés ciblée, cette opération peut baisser la performance du SRI. Pour certains types de requêtes spéciales telles qu'une requête contenant le titre d'une chanson, qui bien évidemment peut contenir des mots vides, la recherche perd sa précision ; mais ces cas se présentent très rarement.

1.3.1.3 Radicalisation

Pour des raisons grammaticales, les documents utilisent différentes formes d'un mot, comme *flexible*, *flexiblement*, *flexibilité*... En outre, il y a des familles de dérivation des relations associées à la même signification. Dans bien de cas, il semble que ça serait utile pour une recherche sur l'un de ces mots de retourner les documents qui contiennent un autre mot dans la série.
L'objectif de la lemmatisation (radicalisation) est de rendre l'ensemble des formes des mots de la même famille représentées par un seul mot pour toute la famille. C'est la forme commune entre eux, qui est la forme de base (radical, par exemple *flexibl* pour *flexible*, *flexiblement*, *flexibilité*...).
Bien que le passage à la forme canonique présente un avantage, puisqu'elle permet d'indexer en un seul mot (lemme ou radical) sa famille morphologique, cette opération supprime la sémantique originale. Mais ceci ne présente aucun inconvénient puisque l'indexation passe totalement inaperçue par l'utilisateur : il s'agit d'un moyen de codage

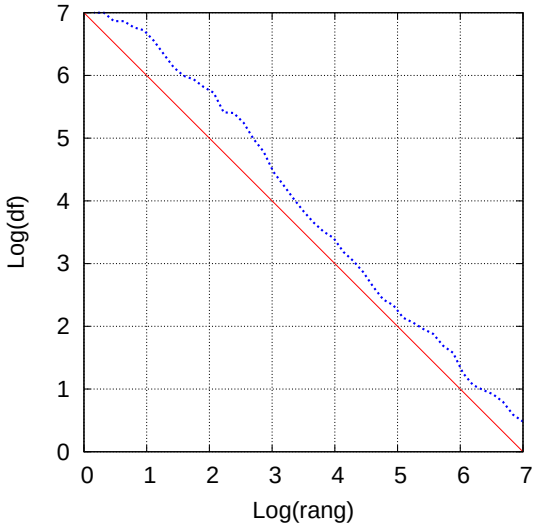
de l'information sans perte. Les plus courants algorithmes pour la radicalisation des mots de la langue anglaise est l'algorithme de Porter [79]. La méthode de radicalisation de Porter permet de construire progressivement le radical en inférant les modifications grammaticales potentielles qui se manifestent le plus souvent par des postfixes ou préfixes. Cette méthode n'est par ailleurs utilisable que pour la langue anglaise. L'indexation de documents écrits en langue française est souvent réalisée par la troncature à 7 caractères. Pour bien d'autres langages, aucune méthode de lemmatisation n'a été popularisée [2].

1.3.1.4 Pondération des termes

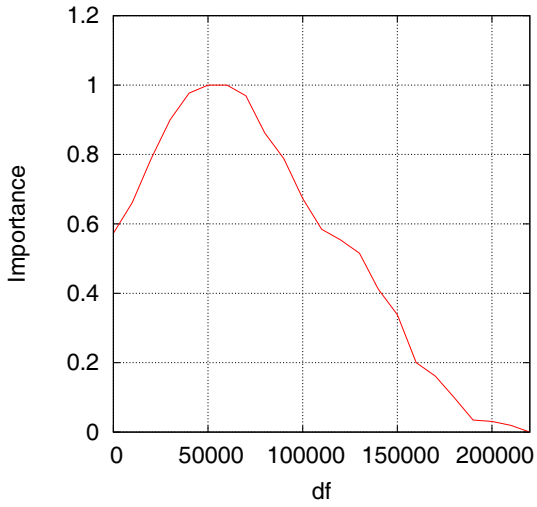
La pondération d'un terme peut être utilisée de diverses manières. Elle peut être simplement la fréquence du terme. Mais celle-ci est abusivement simpliste. De l'autre côté, la pondération est décisive de la performance du système, pour mener à bien les phases de recherche ultérieure, la communauté de RI a longtemps investi dans l'identification de la formule la plus discriminative[2] d'un terme radical. Toutes les formules proposées à cet effet reposent sur un facteur très populaire en RI : le facteur $tf \times idf$.

Approches basées sur la fréquence locale (tf) : L'objectif de ces approches est de trouver les mots qui représentent le mieux le contenu d'un document. Il est généralement admis qu'un mot qui apparaît souvent dans un texte représente un concept important. Ainsi, il convient de choisir les mots représentatifs selon leurs fréquences d'occurrence. La façon la plus simple consiste à définir un seuil de fréquence : si la fréquence d'occurrence d'un mot dépasse ce seuil, alors il est considéré comme important pour le document.
Cependant, les mots les plus fréquents sont des mots usuels et ne présentent paradoxalement aucun intérêt pour un utilisateur potentiel. Quand on fait une statistique d'occurrence, si on classe dans l'ordre décroissant des mots par leurs fréquences, et on leur donne un rang $(1, 2 \ldots)$ alors le rang serait inversement proportionnel à la fréquence $(Log(rang) \times Log(df) \approx constante)$: c'est la loi de Zipf [138].
Selon cette loi, la fréquence est inversement proportionnelle à son rang [83], la distribution des mots suit la courbe illustrée par la figure 1.2. Il devient évident qu'on ne peut pas garder tous les mots dans l'index en définissant un seuil maximal, ni un seuil minimal pour les mots qui ne peuvent pas représenter les documents. L'utilisation de ces deux seuils mesure l'informativité ou encore le sens d'un mot, la correspondance entre l'informativité et la fréquence est illustrée par la figure 1.3. Ainsi, en choisissant

[2]Celle qui attribue à un terme le poids idéal qu'il pourrait quantifier son importance réelle dans un document.



► Figure 1.2: Fréquence d'un terme en fonction de son rang



► Figure 1.3: Importance d'un terme en fonction de sa fréquence

les mots se situant entre ces deux seuils, on peut mieux représenter un document.

Approches basées sur la fréquence globale (idf) : La pondération d'un terme par discrimination reflète qu'un terme distingue bien un document des autres documents. C'est-à-dire, un terme qui a une valeur de discrimination élevée doit apparaître seulement dans un petit nombre de documents : un terme qui apparaît dans tous les documents n'est pas discriminant. La capacité de discrimination d'un terme est importante dans le choix des termes d'indexation à retenir. L'idée est de garder seulement les termes discriminants, et éliminer ceux qui ne le sont pas.
Les approches basées sur tf ainsi que ceux qui sont basées sur idf sont efficaces, par le choix des termes discriminants. La pondération est néanmoins communément basée sur la combinaison des deux mesures. Les raisons pour lesquelles ses deux paramètres ont l'intérêt de coexister est qu'ils sont très révélateurs de l'importance d'un terme, et aussi pour se livrer de l'obligation d'identifier des seuils et de migrer vers l'identification de l'importance d'un terme par une mesure plus statistique.

Approches basées sur $tf \times idf$: La mesure $tf \times idf$ en RI désigne un ensemble de schémas de pondération de termes. tf désigne une mesure en rapport avec l'importance d'un terme pour un document. En général, cette valeur est déterminée par la fréquence du terme dans le document. Par idf, on mesure si le terme est discriminant (un terme est discriminant s'il apparaît peu dans d'autres documents [101]). Cette mesure est utilisée en RI pour calculer la pertinence d'un document par rapport à une requête, ceci traduit le fait qu'un document est pertinent à une requête s'ils partagent assez de termes importants.
La mesure $tf \times idf$ est communément utilisée en RI, vu que cette mesure donne une bonne approximation de l'importance du terme dans le document. Cependant, elle ne donne aucune importance à la longueur du document. Pourtant cette caractéristique est utile dans le processus de recherche. En effet pour les documents longs, la fréquence des termes augmente, de ce fait la similarité entre ces documents et la requête augmente et les documents de petites tailles se trouvent souvent défavorisés. L'introduction de ce facteur (longueur du document) qu'on appelle facteur de *normalisation* [87] [120] s'avère indispensable.

1.3.1.5 Organisation de l'index

Au terme de toutes les différentes étapes expliquées ci-avant (analyse lexicale, élimination des mots vides, radicalisation et choix de la technique de pondération d'un terme), il devient nécessaire d'organiser et de stocker les informations sélectionnées. Le stockage de données peut se faire dans des fichiers structurés ou non structurés. Le fichier de base dans lequel sont stockées les données est appelé *fichier maître*. L'opération peut

Document	Contenu
d_1	La$_{1}$ recherche$_{4}$ d$_{14}$ ' information$_{16}$ gère$_{28}$ des$_{33}$ textes$_{37}$.
d_2	Un$_{1}$ système$_{4}$ de$_{12}$ recherche$_{15}$ d$_{25}$ ' information$_{27}$ doit$_{39}$ restituer$_{44}$ l$_{54}$ ' information$_{56}$ pertinente$_{68}$ à$_{79}$ l$_{81}$ ' utilisateur$_{83}$.
d_3	Une$_{1}$ information$_{5}$ est$_{17}$ pertinente$_{21}$ si$_{32}$ elle$_{35}$ satisfait$_{40}$ l$_{50}$ ' utilisateur$_{52}$.

▶ Tableau 1.1: Exemple de collection (fichier maître)

terme	d_1	d_2	d_3
recherche	4	15	3
information	16	27, 56	5
gère	28		
textes	37		
système		4	
restituer		44	
pertinente		68	21
utilisateur		83	52
satisfait			40

▶ Tableau 1.2: Exemple de fichier inverse

durer quelques secondes sur un fichier maître de quelques centaines d'enregistrements, cependant, elle peut se révéler très lente si la base atteint des milliers de documents. Les *fichiers inverses* sont créés autour du fichier maître [65] [72]. Ces fichiers comme leur nom l'indique, sont le résultat de l'inversion du fichier maître. Plus exactement, au lieu de donner pour chaque document les mots et les fréquences qui le constituent, on donne pour chaque mot les documents qui le contiennent et sa fréquence dans chacun. Le tableau 1.2 illustre un exemple de fichier inverse construit à partir de la collection illustrée par le tableau 1.1.

1.3.2 Appariement document-requête

L'objectif du SRI est le calcul de la pertinence d'un document par rapport à une requête, c'est une fonction d'appariement qui détermine le degré de ressemblance d'un

document par rapport à une requête, et permet éventuellement de classer les documents par ordre de pertinence. Cette fonction est notée $rsv(q, d)$ (*Retrieval Status Value*), où q est une requête et d est un document de la collection.
La fonction d'appariement est indépendante de l'indexation et de la pondération des termes, par contre elle caractérise le SRI plus que le modèle d'indexation : la plupart des approches de recherche inspirent leurs noms à partir de la façon dont ils entament l'appariement.

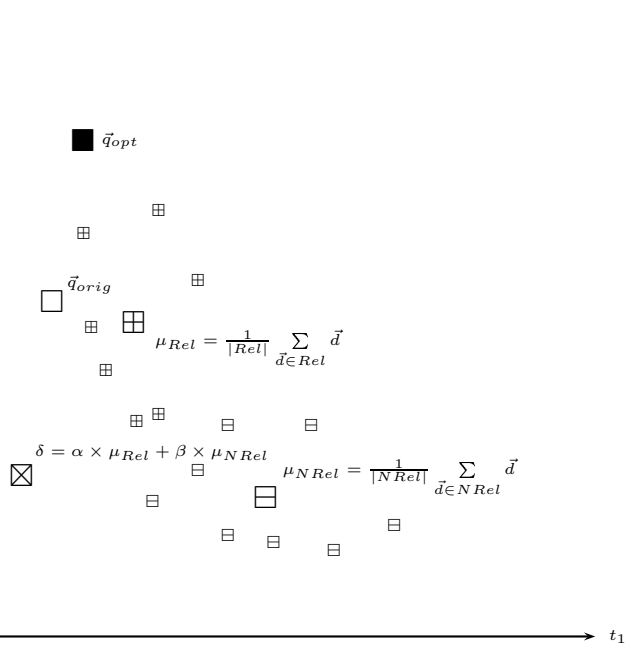
1.3.3 Reformulation de la requête

Il est souvent difficile, pour l'utilisateur, de formuler son besoin exact en information. Par conséquent, les résultats que lui fournit la RI ne lui conviennent parfois pas. Retrouver des informations pertinentes en utilisant seule la requête initiale de l'utilisateur est quasi-impossible. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence système, une étape de reformulation de la requête est souvent utilisée. La requête initiale est traitée comme une tentative pour retrouver de l'information. Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'espoir de retrouver plus de documents pertinents. La reformulation de la requête met en œuvre un algorithme de modification de la requête en termes, en poids ou les deux simultanément, moyennant des critères de choix de termes d'expansion et des règles de calcul des nouveaux poids.
En RI, savoir reformuler une idée par des termes différents est une des clefs pour l'amélioration des performances des SRI existants. L'un des moyens pour résoudre ce problème est d'utiliser des ressources sémantiques spécialisées et adaptées à la base documentaire sur laquelle les recherches sont faites [40] [80] [102].
La stratégie de reformulation de la requête est la plus populaire. On la nomme communément *réinjection de la pertinence* ou *relevance feedback* [17] [89] [97]. La reformulation par réinjection de pertinence est une forme de recherche évolutive et interactive ; elle procède à la modification de la requête initiale en termes et en poids, sur la base des jugements de pertinence de l'utilisateur sur les documents restitués par le SRI. Son principe fondamental est d'utiliser la requête initiale, puis exploiter itérativement les jugements de pertinence de l'utilisateur afin d'ajuster la requête par expansion, repondération ou combinaison des deux procédures, en direction des documents pertinents.

La méthode de Rocchio est largement utilisée en RI. Rocchio [89] a proposé une formule de reformulation de la requête permettant d'approcher la requête vers les documents pertinents et de l'éloigner des documents non pertinents en partant de la requête initiale :

$$\vec{q}_{opt} = \vec{q}_{orig} + \alpha \frac{1}{|Rel|} \sum_{\vec{d} \in Rel} \vec{d} + \beta \frac{1}{|NRel|} \sum_{\vec{d} \in NRel} \vec{d}$$

où α et β sont des constantes telles que $\alpha > 0$ et $\beta < 0$, $\vec{q}_{orig}$ et $\vec{q}_{opt}$ représentent respectivement la requête initiale et la requête reformulée, Rel (resp. $NRel$) représente l'ensemble des documents pertinents (resp. non pertinents) et $\vec{d}$ est un document. Le document et la requête sont représentés par des termes pondérés : une représentation vectorielle est alors appropriée. La méthode de reformulation de Rocchio consiste à représenter la requête reformulée par un vecteur en fonction de la requête initiale, des moyennes des documents pertinents et des documents non pertinents et des paramètres α et β. La figure 1.4 illustre un exemple de reformulation, les valeurs de α et β sont respectivement 0.3 et -0.2.



► Figure 1.4: Reformulation de la requête par réinjection de pertinence avec la méthode de Rocchio

Dans la figure 1.4, chaque document pertinent (resp. non pertinent) est représenté par le symbole ⊞ (resp. ⊟), la moyenne des documents pertinents $\mu_r = \frac{1}{|Rel|} \sum_{\vec{d} \in Rel} \vec{d}$ (resp. non pertinents $\mu_s = \frac{1}{|NRel|} \sum_{\vec{d} \in NRel} \vec{d}$) est représentée par le symbole ⊞ (resp. ⊟). La requête initiale (resp. reformulée) est représentée par le symbole □ (resp. ■).

1.4 Taxonomie des modèles de RI

Un modèle de RI modélise la fonction d'appariement qui joue un rôle central dans la RI. C'est le modèle qui détermine le comportement clé d'un SRI. On distingue trois principaux modèles pour la RI, qui sont cependant étroitement liés.

1.4.1 Le modèle booléen ou ensembliste et ses dérivés

Dans ce modèle, un document est représenté par un ensemble de termes. Une requête est une expression logique composée de termes assemblés par les opérateurs logiques **et**, **ou** et **non**.
La formulation de la requête se base sur les trois opérateurs booléens :

- La conjonction **et** ($\wedge$), exige que les termes soient présents simultanément dans la description d'un document,
- La disjonction **ou** ($\vee$), exige qu'au moins un des termes soit présent dans la description des documents à retourner,
- La négation **non** ($\neg$), utilisée pour écarter les documents qui contiennent un terme.

1.4.1.1 Le modèle booléen basique

Le modèle booléen [96] est le modèle le plus ancien dans la RI, les documents réagissent suivant la présence ou l'absence des termes utilisés. L'approche booléenne utilise le mode d'appariement exact qui consiste à ne restituer que les documents répondant exactement à la requête.
Le modèle booléen considère dans l'index qu'un terme est présent ou non dans le document, par conséquent le poids d'un terme noté $w_{i,j} \in \{0, 1\}$.
Considérons une requête contenant trois termes (t_a, t_b et t_c), et l'expression logique de la requête définie par $q = t_a \vee (t_b \wedge \neg t_c)$, la similarité entre un document et la requête est définie par :

$$rsv(q, d) = \begin{cases} 1 & \text{si } d \text{ contient } t_a \\ 1 & \text{si } d \text{ contient } t_b \text{ et ne contient pas } t_c \\ 0 & \text{si non} \end{cases}$$

Le modèle booléen considère qu'un document est soit *pertinent* soit *non pertinent*. L'inconvénient de ce modèle est qu'il rend la tâche de l'utilisateur pour formuler son

besoin en information plus complexe, à cause de sa manière d'appariement. De plus, 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 d'un utilisateur novice. Par exemple, si l'utilisateur formule une requête $t_1 \wedge t_2 \wedge t_3 \wedge t_4$, et si aucun document ne répond exactement à la requête, il convient tout de même de présenter un panorama des documents qui répondent partiellement à celle-ci en admettant qu'un document qui contient les termes t_1, t_2 et t_3 est plus pertinent qu'un autre qui ne contient que les termes t_1 et t_2, ou un autre qui ne contient aucun des termes de la liste. Tout l'art est de trouver le moyen permettant une utilisation plus flexible des opérateurs booléens. Le modèle booléen étendu est l'un des premiers modèles qui ont été proposés à cette fin.

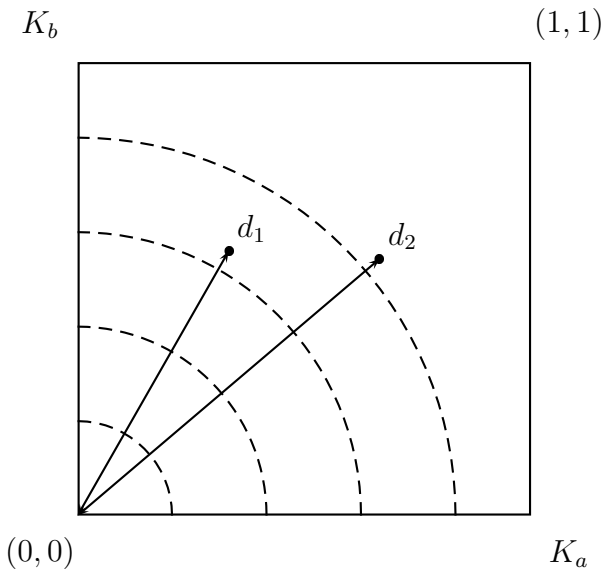
1.4.1.2 Le modèle booléen étendu

Le modèle booléen étendu est introduit en 1983 par Salton, Fox et Wu [98], l'idée est de permettre l'utilisation des opérateurs logiques tout en proposant une pertinence graduée. Ce modèle peut être vu comme une combinaison des modèles booléen et vectoriel (section 1.3.2). Au lieu d'estimer le poids d'un terme par son absence ou présence, la pondération des termes dans un document est basée sur $tf \times idf$ normalisé c'est-à-dire que le poids d'un terme dans un document se trouve entre 0 et 1. Ce modèle consiste à calculer la distance entre les coordonnées d'un document et les coordonnées d'une requête.
Le modèle booléen étendu considère que les opérations booléennes ont une influence sur la façon dont il faut entreprendre la requête. La représentation d'un document contrairement au modèle booléen basique, tient compte des poids des termes. Chaque document est représenté par un vecteur de termes pondérés. Selon l'opérateur booléen utilisé, le modèle booléen étendu calcule le score d'un document selon deux cas principaux :

- Requête de type disjonction : k_a *ou* k_b, où k_a et k_b sont deux termes. Si on considère un document d dont les poids respectifs de k_a et k_b sont w_a et w_b alors le score de d par rapport à la requête $k_a \vee k_b$ est estimé selon sa distance par rapport au point d'origine $(0, 0)$:

$$\begin{aligned} score(k_a \text{ ou } k_b, d) &= \sqrt{\frac{score^2(k_a,d)+score^2(k_b,d)}{2}} \\ &= \sqrt{\frac{w_a^2+w_b^2}{2}} \end{aligned}$$

  La figure 1.5 montre que le document d_2 est plus pertinent que le document d_1 malgré qu'ils contiennent les termes k_a et k_b (les poids de chacun sont différents). Les quarts de cercles concentriques représentent les documents ayant le même score.



► Figure 1.5: Mesure de similarité entre un document et une requête de type **ou**

- Requête de type Conjonction : k_a *et* k_b, où k_a et k_b sont deux termes. Si on considère un document d dont les poids respectifs de k_a et k_b sont w_a et w_b alors le score de d par rapport à la requête $k_a \wedge k_b$ est estimé selon sa distance par rapport au point $(1,1)$. La figure 1.6 montre que le document d_1 est plus pertinent que le document d_2 malgré qu'ils contiennent les termes k_a et k_b (les poids de chacun sont différents). Les quarts de cercles concentriques représentent les documents ayant le même score.

  Le point (1, 1) représente la situation où les deux termes k_a et k_b sont présents dans le document. La mesure de similarité de cette requête par rapport à un document est :

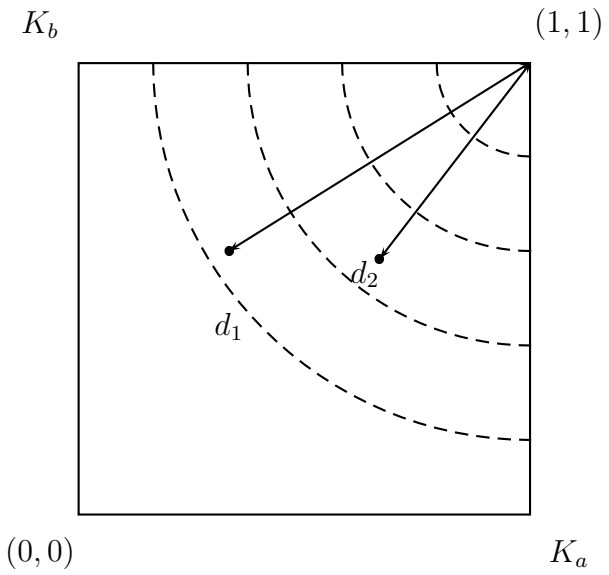
$$\begin{aligned} score(k_a \ et \ k_b, d) &= 1 - \sqrt{\frac{(1-score(k_a,d))^2+(1-score(k_b,d))^2}{2}} \\ &= 1 - \sqrt{\frac{(1-w_a)^2+(1-w_b)^2}{2}} \end{aligned}$$

- Requête de type négation : *non* k_a, le score d'un document est estimé par :

$$score(non \ k_a, d) = 1 - score(k_a, d)$$

  Les requêtes complexes sont traitées récursivement par exemple, le score d'un document d par rapport à une requête $k_a \wedge (k_b \vee k_c \wedge \neg k_d)$ est calculé comme suit :

$$\begin{aligned} score(d, k_a \wedge (k_b \vee k_c \wedge \neg k_d)) &= 1 - \sqrt{\frac{(1-score(k_a,d))^2+(1-score(k_b \vee k_c \wedge \neg k_d,d))^2}{2}} \\ score(k_b \vee k_c \wedge \neg k_d, d) &= \sqrt{\frac{score^2(k_b,d)+score^2(k_c \wedge \neg k_d,d)}{2}} \\ score(k_c \wedge \neg k_d, d) &= 1 - \sqrt{\frac{(1-score(k_c,d))^2+(1-score(\neg k_d,d))^2}{2}} \\ score(\neg k_d, d) &= 1 - score(k_d, d) \\ score(k_x, d) &= k_x \forall x \in \{a, b, c, d\} \end{aligned}$$



▶ Figure 1.6: Mesure de similarité entre un document et une requête de type **et**

1.4.1.3 Le modèle *p-norm*

Le modèle *p-norm* [121] généralise cette notion en utilisant, au lieu des distances euclidiennes, les p-distances, avec $1 \leq p$. Pour une requête de m termes, la représentation de la requête est :

$$q_{ou} = k_1 \vee^p k_2 \vee^p \ldots \vee^p k_m$$

Par analogie, la requête de conjonction est représentée par :

$$q_{et} = k_1 \wedge^p k_2 \wedge^p \ldots \wedge^p k_m$$

Le calcul du score devient alors :

$$\begin{aligned} rsv(q_{ou}, \vec{d}) &= \left(\frac{x_1^p + x_2^p \ldots x_m^p}{m}\right)^{\frac{1}{p}} \\ rsv(q_{et}, \vec{d}) &= 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p \ldots (1-x_m)^p}{m}\right)^{\frac{1}{p}} \end{aligned}$$

Si $p = 2$ on se retrouve dans l'exemple précédent. Si $p = \infty$, on peut vérifier que :

$$\begin{aligned} rsv(q_{ou}, \vec{d}) &\propto \max(x_i) \\ rsv(q_{et}, \vec{d}) &\propto \min(x_i) \end{aligned}$$

en effet :

$$\begin{aligned} rsv(q_{ou}, \vec{d}) &= \lim_{p \to \infty} \left(\frac{x_1^p + x_2^p \ldots x_m^p}{m}\right)^{\frac{1}{p}} \\ &= \lim_{p \to \infty} \left(\frac{\max(x_i^p)}{m}\right)^{\frac{1}{p}} \\ &= \frac{\max(x_i)}{m^{\frac{1}{p}}} \\ &\propto \max(x_i) \end{aligned}$$

et :

$$\begin{aligned} rsv(q_{et}, \vec{d}) &= \lim_{p\to+\infty} 1 - \left(\frac{(1-x_1)^p+(1-x_2)^p\ldots(1-x_m)^p}{m}\right)^{\frac{1}{p}} \\ &= \lim_{p\to+\infty} \left(\frac{\max(1-x_i)^p}{m}\right)^{\frac{1}{p}} \\ &= 1 - \lim_{p\to+\infty} \left(\frac{(1-\min(x_i))^p}{m}\right)^{\frac{1}{p}} \\ &= 1 - \frac{1-\min(x_i)}{m^{\frac{1}{p}}} \\ &= 1 - \frac{1}{m^{\frac{1}{p}}} + \frac{\min(x_i)}{m^{\frac{1}{p}}} \\ &\propto \min(x_i) \end{aligned}$$

On peut conclure que le modèle booléen étendu est un modèle hybride, puisqu'il inclut les modèles ensemblistes et algébriques.

1.4.1.4 Le modèle flou

La représentation des documents et des requêtes reflète partiellement les contenus sémantiques des documents et des requêtes [19] [20]. Par conséquent la correspondance d'un document par rapport à une requête est approximative ou vague.
Cette fonction d'approximation est un sous-ensemble d'un univers de discours U caractérisée par la fonction $\mu_A : U \to [0, 1]$ qui associe à chaque élément u de U la valeur $\mu_A(u) \in [0, 1]$.
Le degré d'appartenance est utilisé pour représenter l'incertitude ou l'ambiguïté [137], les trois opérations les plus couramment effectuées sur des ensembles flous sont :

$$\begin{aligned} \mu(a\ ou\ b) &= \max(\mu_a, \mu_b) \\ \mu(a\ et\ b) &= \min(\mu_a, \mu_b) \\ \mu(non\ a) &= 1 - \mu(a) \end{aligned}$$

Ces opérateurs ne sont pas efficaces dans certains cas dans un processus de RI. En effet considérons une requête $a\ et\ b$, un document appartenant à l'ensemble flou relatif à a avec $T(a) = 0.6$ et à l'ensemble flou relatif à b avec $T(b) = 0.7$, aura le même score qu'un document appartenant à l'ensemble flou relatif à a avec $T^{'}(a) = 0.6$ et $T^{'}(b) > T(b)$.
Le score d'un document est calculé comme suit :

$$rsv(d, q) = \frac{\sum_{k=1}^{n} r^{k-1} T(a_k)}{\sum_{k=1}^{n} r^{k-1}}$$

où $T(a_k)$ sont considérés dans l'ordre décroissant pour les requêtes **ou** et croissant pour les requêtes **et**, la valeur de r est déterminée expérimentalement.

1.4.2 Le modèle algébrique et ses dérivés

Les modèles algébriques proposent une représentation vectorielle pour le document et la requête. La mise en correspondance entre le document et la requête consiste à calculer la similarité entre les vecteurs représentant les documents et les requêtes.
Le modèle vectoriel est l'ancêtre de tous les modèles algébriques. Les premiers travaux de Salton [95] avaient pour finalité de concevoir la fonction d'appariement selon les propriétés et les opérations associées au concept d'espace vectoriel. Bien que simpliste, ce modèle reste un des plus utilisés et des plus efficaces.

Les modèles dérivés du modèle booléen préconisent une habilité importante à fournir des listes ordonnées de résultats. D'un point de vue statistique, les requêtes qui ne contiennent que des mots clés (sans les opérateurs ensemblistes **et**, **ou** et **non**), qui sont d'ailleurs les plus fréquemment composées, prétendent une synthèse plus approfondie. Plusieurs initiatives ont été proposées pour l'estimation de la fonction d'appariement, son comportement et son habilité à se rapprocher de l'appariement perçu par l'utilisateur, mais elles sont toutes dérivées soit de la logique vectorielle soit de la logique probabiliste. Nous détaillons dans la section suivante ces modèles et leurs modèles dérivés.

1.4.2.1 Le modèle vectoriel basique

Le modèle vectoriel (nommé aussi *VSM* pour *Vector Space Model*), a été popularisé par Salton en 1971 [95] [99]. Ce modèle propose de représenter les documents et les requêtes par des vecteurs d'indexation dans un espace engendré par les termes d'indexation. Le modèle vectoriel représente les requêtes et les documents sous forme de vecteurs dans un même espace vectoriel. La mesure de similarité entre le document représenté par un vecteur $\vec{d} = (d_1, d_2 \dots d_n)$, où d_i est le poids d'un terme i dans le document, et la requête définie par $\vec{q} = (q_1, q_2 \dots q_n)$ où q_i est le poids (souvent 0 ou 1 selon que le terme appartient ou pas à la requête) du terme i dans la requête, est estimée selon les propriétés et les mesures populaires issues de la théorie des espaces vectoriels. Ils existent plusieurs mesures pour calculer la similarité entre le document et la requête, la plus simple est le produit scalaire :

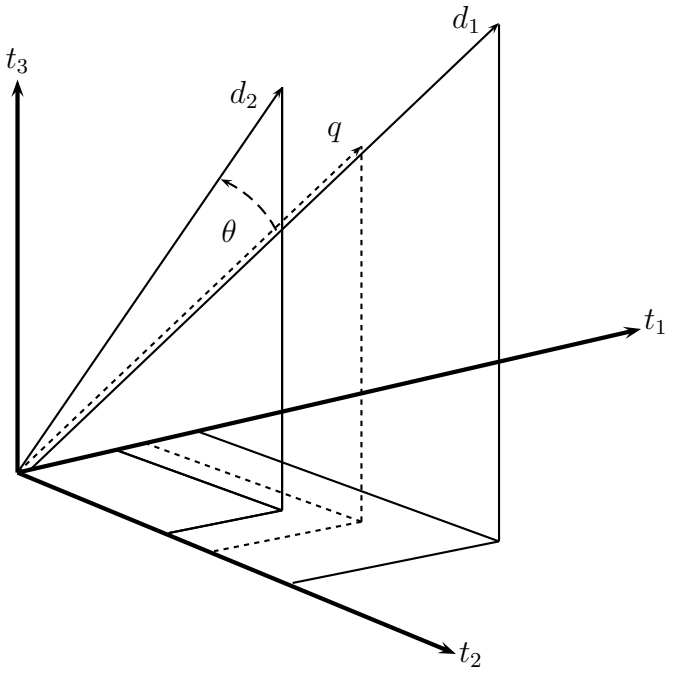
$$rsv(\vec{d}, \vec{q}) = \sum_{k=1}^{n} d_k \times q_k$$

Si les composantes des deux vecteurs sont binaires (1 si le terme existe dans le document, 0 si non) alors la mesure de similarité entre le document et la requête est égale au nombre de mots partagés entre eux.
Le produit scalaire est très sensible à la norme des vecteurs document et requête (leurs longueurs). D'autres mesures ont été proposées, elles sont tout de même basées sur

le produit scalaire. La mesure du cosinus, qui mesure le cosinus de l'angle formé par le document et la requête, est utilisé dans le modèle vectoriel : plus l'angle est petit, plus la requête est proche du document et par conséquent plus le cosinus de l'angle est élevé. La mesure cosinus est donnée par :

$$rsv(\vec{d},\vec{q}) = cos(\vec{d},\vec{q}) = \frac{\vec{d}\times\vec{q}}{|\vec{d}|\times|\vec{q}|} = \frac{\sum_{k=1}^{n} d_k \times q_k}{\sqrt{\sum_{k=1}^{n} d_k^2 . \sum_{k=1}^{n} q_k^2}}$$

La figure 1.7 illustre le cosinus de l'angle formé par une requête et deux documents d_1 et d_2. Le document d_2 est différent de la requête, le cosinus de l'angle résultant est égal à $cos(\theta) < 1$. Le document d_1 est contrairement à d_2, très proche de la requête, son rsv est égal à $1 = cos(0)$. D'autres mesures [84] sont utilisées pour percevoir le score



▶ Figure 1.7: Modèle vectoriel

d'un vecteur document par rapport à un vecteur requête :

- Jaccard : $rsv(d,q) = \frac{\sum_{k=1}^{n} d_k.q_k}{\sum_{k=1}^{n} d_k^2 + \sum_{k=1}^{n} q_k^2 - \sum_{k=1}^{n} d_k.q_k}$
- Dice : $rsv(d,q) = \frac{2\times\sum_{k=1}^{n} d_k.q_k}{\sum_{k=1}^{n} d_k^2 + \sum_{k=1}^{n} q_k^2}$

- Overlap : $rsv(d,q) = \frac{\sum_{k=1}^{n} d_k.q_k}{\min\left(\sum_{k=1}^{n} d_k^2, \sum_{k=1}^{n} q_k^2\right)}$

Toutes ces mesures ont l'avantage de profiter des propriétés de l'espace vectoriel pour la perception de l'appariement utilisateur. Le principal intérêt porté à leur application est leur habilité à retourner des listes ordonnées de documents.
Le principal inconvénient du modèle vectoriel est le fait qu'il suppose que les termes d'indexation forment une base. Or ils existent énormément de relations sémantiques qui font qu'un terme pourra s'exprimer en fonction des autres[3]. Par ailleurs il est très difficile voire impossible de traduire des relations par des combinaisons linéaires de termes, or ceci s'avère indispensable à la construction de vraie base de termes d'indexation.
La représentation vectorielle procure une autre fonctionnalité très importante : le degré de discrimination des termes d'indexation. Le modèle *LSI* a été proposé pour percevoir ce degré et d'exprimer un terme en fonction des autres.

1.4.2.2 LSI (*Latent Semantic Indexing*)

La représentation des documents et des requêtes dans le modèle vectoriel souffre de son incapacité de gérer les synonymes (voiture et automobile) les hyponymes... Or l'espace vectoriel de représentation ne parvient pas à saisir la relation entre les termes, ce qui influe sur le comportement du SRI. Prenons un document d_i contenant deux termes (automobile et voiture), et un autre document d_j qui contient uniquement un terme (voiture). Il est clair que le document d_i est plus pertinent à une requête q qui ne contient qu'un seul terme (voiture), pourtant ils possèdent le même contexte des documents potentiellement pertinents. L'intérêt du modèle LSI [39] est de remédier à ce problème. Pour ce faire l'idée consiste à transformer les vecteurs de termes de l'index dans un autre espace vectoriel de dimension associée aux concepts.
Le modèle LSI utilise la *SVD* (*Singular Value Decomposition*) pour créer un nouvel espace vectoriel :

$$A = U \times \Sigma \times V^T$$

où A est la matrice documents-termes originale $m \times n$, U est une matrice $m \times r$, Σ est une matrice diagonale $r \times r$ (seulement les éléments en diagonale sont non nuls) et V^T est une matrice $r \times n$. La valeur r est telle que $r = \inf(n, m)$ où m est le nombre de termes et n est le nombre de documents. On trie les valeurs dans Σ dans l'ordre

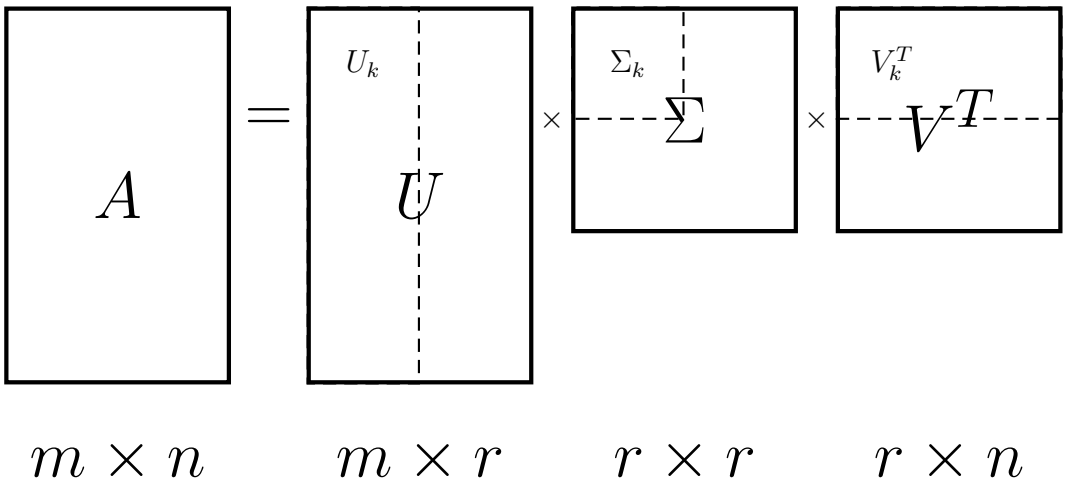
[3]Les ontologies sont des représentations très complexes de telles relations.

décroissant. Il existe une seule décomposition de cette façon :

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{pmatrix}$$

avec $\sigma_1 \geq \sigma_2 \dots \sigma_r$.

La représentation par des mots-clés contient beaucoup de bruit. Typiquement, ce bruit se traduit par les valeurs les moins élevées de Σ. Ainsi, la technique de LSI tend à supprimer ce bruit en écartant ces valeurs, ce qui ramène la dimensions de Σ à k, cette matrice réduite est notée Σ_k (voir figure 1.8 pour illustration). Quand une requête est



▶ Figure 1.8: Modèle LSI

soumise, elle est aussi traduite dans ce nouvel espace (changement de base) :

$$q = q^T \times U_k \times \Sigma_k^{-1}$$

La réduction de la dimension de l'espace vectoriel se traduit par une compression sans perte considérable. La quantité de l'information préservée par la décomposition de A en $U_k \times \Sigma_k \times V_k^T$ peut être évaluée comme suit :

$$contraste = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{r} \sigma_i}$$

1.4.2.3 Le modèle connexionniste

L'objectif du modèle connexionniste (réseaux de neurones), est d'imiter les fonctionnalités du cerveau humain. Il est maintenant bien établi que notre cerveau est composé de milliards de neurones, chaque neurone peut être vu comme une unité de traitement de base qui, lorsqu'elle est stimulée par des signaux d'entrée, peut émettre des signaux de sortie comme un réactif.
Formellement, le réseau de neurones R est composé d'un ensemble de couches L_i et un ensemble de liens C_i :

$$R = ((L_1, L_2 \dots L_n), (C_1, C_2 \dots C_{n-1}))$$

Chaque couche L_i est composée des neurones n_{ij} :

$$\forall\ 1 \leq i \leq n, L_i = \{n_{i1}, n_{i2} \dots n_{i|L_i|}\}$$

Chaque neurone n_{ij} est caractérisé par un poids w_{ij} et une fonction d'activation g_{ij}, la relation entre les différents neurones est définie comme suit :

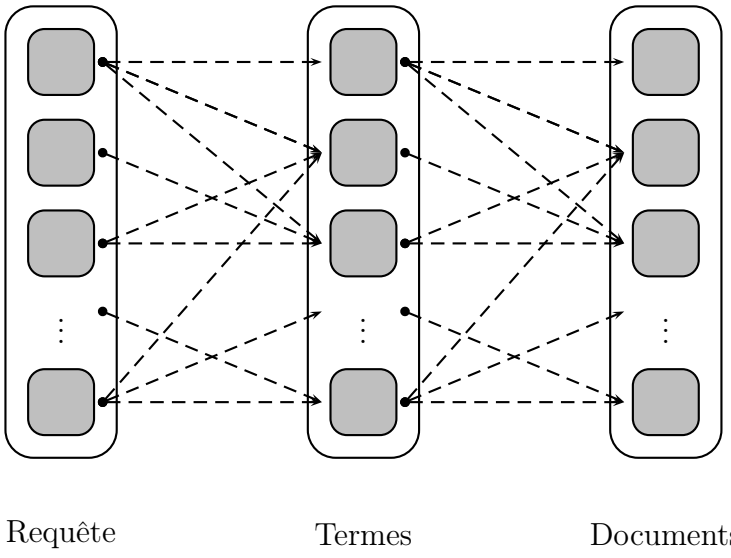
$$\begin{array}{rcl} \forall 1 \leq i \leq n, C_i : L_i \times L_{i+1} & \rightarrow & \mathbb{R} \\ (n_{ij}, n_{i+1k}) & \mapsto & C_i\,(n_{ij}, n_{i+1k}) \end{array}$$

Un réseau de neurones pour la RI peut être composé de trois couches illustrées par la figure 1.9, cette architecture a été proposée par Boughanem et al. [16], Kwok [61] a proposé une architecture similaire. La première couche représente les termes d'une requête, la deuxième représente les termes d'indexation et la troisième représente les documents de la collection. Dans le cadre de la RI il n'existe pas une seule représentation du réseau de neurones [135], c'est-à-dire le nombre de couches, le nombre de neurones par couche, la fonction de sortie, les liens entre les neurones et les poids des neurones. L'interrogation se fait par propagation de signaux de la couche d'entrée vers la couche de sortie via la couche intermédiaire, l'algorithme 1 illustre le processus de propagation des signaux dans un réseau connexionniste. Les valeurs d'activation affichées à la couche de sortie peuvent être perçues comme des scores de documents. La caractéristique la

Algorithme 1 Propagation

1: Présenter une entrée à la couche d'entrée
2: **pour** i de 2 à n **faire**
3: **pour** j de 1 à $|L_{n-1}|$ **faire**
4: $w_{ij} \leftarrow \sum_{k=1}^{|L_{i-1}|} g_{i-1k}(w_{i-1k}) C_{i-1}(n_{i-1k}, n_{ij})$
5: **fin pour**
6: **fin pour**

plus importante dans un réseau de neurone est la fonction d'apprentissage. Pour effectuer l'apprentissage, l'algorithme le plus utilisé est l'algorithme de rétro-propagation



▶ Figure 1.9: Un modèle de réseau de neurones pour la RI

Algorithme 2 Rétropropagation : neurone n_{ni}

Calculer l'erreuer $\delta_{ni} \leftarrow g'_{ni}(w_{ni}).(des_{ni} - g_{ni}(w_{ni}))$
pour j de $n-1$ à 1 **faire**
  pour k de 1 à $|L_j|$ **faire**
    $\delta_{jk} \leftarrow g'_{jk}(w_{jk}) \sum_{l=1}^{|L_{j+1}|} g_{j+1l}(w_{j+1l})$
    pour l de 1 à $|L_{j+1}|$ **faire**
      $C_{jj+1}(n_{jk}, n_{j+1l}) \leftarrow C_{jj+1}(n_{jk}, n_{j+1l}) + \alpha.\delta_{j+1l}.g_{jk}(w_{jk})$
    fin pour
  fin pour
fin pour

du gradient [91] (voir algorithme 2). L'apprentissage se fait par calcul d'erreur dans la couche documents et de le rétropropager vers les autres couches du réseau afin d'ajuster les poids des liens. La principale difficulté dans l'utilisation des réseaux de neurones pour la RI est l'estimation de cette valeur d'erreur, car les jugements de pertinence ne fournissent que des valeurs binaires pour les documents (1 pour les documents pertinents, 0 pour les autres).
Par ailleurs, il a été démontré expérimentalement que la convergence de cet algorithme dépend étroitement du nombre de couches cachées. Celles-ci augmentent la complexité de l'interrogation et de l'apprentissage, en plus, il est très difficile de concevoir des couches cachées dans un contexte où les termes et les documents sont déjà représentés dans les couches d'entrée et de sortie. Quelques alternatives à la rétro-propagation de gradient ont été proposées. La méthode de rétro-propagation de la pertinence [125] bien que plus simple fournit des résultats meilleurs.

1.4.3 Le modèle probabiliste et ses dérivés

Le modèle probabiliste est basé sur la théorie de la décision. Le but est de calculer la probabilité de la pertinence d'un document d par rapport à une requête q.

1.4.3.1 Le modèle probabiliste basique

Dans ce modèle un document et une requête sont représentés par un vecteur comme dans le modèle vectoriel, mais les poids des termes sont binaires, soient $D_1, D_2 \dots D_n$ (D_i est égale à 1 si le terme t_i est présent dans le document et 0 si non). L'idée de base dans ce modèle est de tenter de déterminer les probabilités $p(L = 1|d)$ (la probabilité que le document d soit pertinent pour la requête q) et $p(L = 0|d)$ (la probabilité que le document d ne soit pas pertinent pour la requête q).
Si le document d contient les termes $D_1, D_2 \dots D_n$, alors $p(L = 1|d)$ qui mesure la probabilité que d soit pertinent pourra se ramener à la probabilité qu'un document soit pertinent sachant qu'il contient les termes $(D_1, D_2 \dots D_n)$, cette probabilité sera notée $P(L = 1|D_1, D_2 \dots D_n)$ [69] [86].
La fonction de similarité entre un document d et une requête q est donnée par :

$$rsv(d, q) = \log \frac{P(L = 1|D_1, D_2 \dots D_n)}{P(L = 0|D_1, D_2 \dots D_n)}$$

Cependant, il y a très peu de documents, voire aucun, qui contiennent exactement le même contenu que d quel qu'il en soit, d'où $P(L = 1|D_1, D_2 \dots D_n)$ devient presque nul, et par la même $P(L = 0|D_1, D_2 \dots D_n) = 1 - P(L = 1|D_1, D_2 \dots D_n)$ devient presque égal à 1. Or, le calcul de $P(L = 0|D_1, D_2 \dots D_n)$ donne paradoxalement une

valeur presque nulle également.
On utilise la règle de transformation de Bayes :

$$\begin{aligned} rsv(d,q) &= logitP(L=1|D_1,D_2\dots D_n) \\ &= log\frac{P(L=1|D_1,D_2\dots D_n)}{P(L=0|D_1,D_2\dots D_n)} \\ &= log\frac{\frac{P(L=1)P(D_1,D_2\dots D_n|L=1)}{P(D_1,D_2\dots D_n)}}{\frac{P(L=0)P(D_1,D_2\dots D_n|L=0)}{P(D_1,D_2\dots D_n)}} \\ &= log\frac{P(D_1,D_2\dots D_n|L=1)}{P(D_1,D_2\dots D_n|L=0)} + logitP(L=1) \end{aligned}$$

L'hypothèse d'indépendance entre les termes est supposée pour simplifier le calcul $(P(t_i|t_j) = P(t_i))$, ainsi :

$$P(D_1,D_2\dots D_n|L=1) = \prod_{i=1}^{n} P(D_i|L=1)$$

et :

$$P(D_1,D_2\dots D_n|L=0) = \prod_{i=1}^{n} P(D_i|L=0)$$

ainsi :

$$\begin{aligned} rsv(d,q) &= log\frac{\prod_{i=1}^{n} P(D_i|L=1)}{\prod_{i=1}^{n} P(D_i|L=0)} + logitP(L=1) \\ &= log\prod_{i=1}^{n}\frac{P(D_i|L=1)}{P(D_i|L=0)} + logitP(L=1) \\ &= \sum_{i=1}^{n} log\frac{P(D_i|L=1)}{P(D_i|L=0)} + logitP(L=1) \end{aligned}$$

Le calcul de la pertinence d'un document par rapport à une requête nécessite une base d'apprentissage dans laquelle la pertinence de quelques documents est connue ainsi :

$$\begin{aligned} P(D_i|L=1) &= \frac{|Rel\cap doc_i|}{|Rel|} \\ P(D_i|L=0) &= \frac{|NRel\cap doc_i|}{|NRel|} \\ P(L=1) &= \frac{|Rel|}{|Rel|+|NRel|} \\ P(L=0) &= \frac{|NRel|}{|Rel|+|NRel|} \end{aligned}$$

où doc_i est l'ensemble des documents contenant le terme t_i, Rel et $NRel$ représentent respectivement les ensembles de documents pertinents et non pertinents de la base d'apprentissage.
Outre la supposition de l'indépendance entre les évènements, le modèle probabiliste ne tient pas compte de la fréquence d'un terme dans un document, or ce critère est indispensable pour quantifier le score d'un document. Cependant, la prise en compte de cette mesure peut avoir des conséquences très négatives sur le calcul des scores. En effet, si un terme apparaît 15 fois dans un document et 14 fois dans un autre, le modèle probabiliste considère en revanche que les fréquences sont différentes malgré qu'elles sont sensiblement les mêmes. Le modèle 2-Poisson a été proposé pour pallier à cette ambiguîté.

1.4.3.2 Le modèle 2-Poisson

L'objectif du modèle 2-Poisson [85] est d'intégrer dans la modélisation probabiliste la fréquence des termes dans les documents et requêtes et la longueur des documents. Cette mesure est à la base d'un des SRI les plus performants à l'heure actuelle : le système *Okapi* [88].
Le modèle 2-Poisson est une hypothèse de répartition basée sur la pertinence à une requête qui peut bien entendu, contenir de nombreux concepts liés aux documents potentiellement pertinents, plutôt que directement à la fréquence des termes. On considère dans ce modèle que la longueur du document est une constante, bien que techniquement il n'a pas besoin de cette hypothèse. Les paramètres de distribution pour ce modèle sert à estimer la fréquence des termes au sein du document, Ils supposent que cette fréquence est une combinaison de deux distributions de poisson de paramètres respectifs μ_1 et μ_2 définies comme suit :

$$P(X = tf) = \lambda \frac{e^{-\mu_1} \mu_1^{tf}}{tf!} + (1 - \lambda) \frac{e^{-\mu_2} \mu_2^{tf}}{tf!}$$

où X est la variable aléatoire de la fréquence d'apparition.
Le modèle 2-Poisson suppose que les documents sont créés à partir d'une source aléatoire d'occurrence. Pour chaque terme, la collection peut être divisée en deux sous-ensembles :

- un ensemble de documents qui traitent du sujet du terme, caractérisé par une fréquence moyenne μ_1 du terme en question ;
- un ensemble de documents qui ne traitent pas du sujet du terme, caractérisé par une fréquence moyenne μ_2 du terme en question.

λ représente la proportion des documents du premier sous-ensemble par rapport à la collection entière, les moyennes μ_1 et μ_2 ($\mu_1 > \mu_2$) peuvent être estimées à partir du

nombre moyen d'occurrence du terme dans chaque sous-ensemble.

Le modèle 2-Poisson peut être utilisé pour l'estimation de la probabilité $P(D_i|L = [1,0])$ où D_i désigne maintenant la fréquence du terme t_i au lieu de sa présence (1) ou absence (0) :

$$\begin{aligned} P(D_i|L=1) &= \frac{e^{-\mu_1}\mu_1^{D_i}}{D_i!} \\ P(D_i|L=0) &= \frac{e^{-\mu_2}\mu_2^{D_i}}{D_i!} \end{aligned}$$

où μ_1 (resp. μ_2) représentent les fréquences moyennes du terme dans les documents pertinents (resp. non pertinents).

1.4.3.3 Le modèle bayésien

Les réseaux bayésiens [74] sont des graphes orientés acycliques dans lesquels les nœuds représentent des variables aléatoires, et les liens sont des dépendances entre ces variables. On distingue deux principaux types de réseaux, les réseaux d'inférence et les réseaux de croyance.

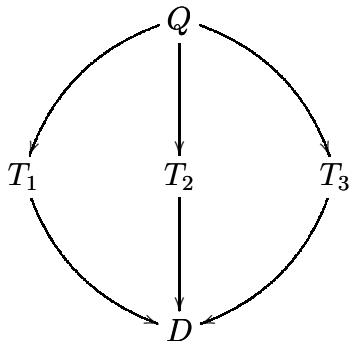
Les réseaux bayésiens d'inférence : En attachant des probabilités initiales aux racines du graphe, on calcule de proche en proche le degré de croyance associé aux autres nœuds du graphe.

Les réseaux inférentiels bayésiens associent des variables aléatoires aux termes d'indexation, aux documents et aux requêtes de l'utilisateur.

Une variable aléatoire associée à un document représente l'évènement d'observer ce document. Les arcs sont dirigés du nœud document vers les nœuds termes, l'observation d'un document est alors conditionnée par l'augmentation de la valeur des variables associées aux termes d'indexation [15] [22]. Ceux-ci conditionnent à leur tour l'observation de la requête. Ainsi, les arcs sont orientés des nœuds associées aux termes d'indexation vers le nœud de la requête. La figure 1.10 illustre un réseau inférentiel bayésien simple de pertinence d'un document à une requête de trois termes.

L'évènement *la requête est accomplie* ($Q = 1$) est réalisé si le sujet lié à un terme est vrai ($T_1 = 1, T_2 = 1$ ou $T_3 = 1$), ou une combinaison de ces évènements.

Les trois sujets sont inférés par l'évènement *le document est pertinent* ($D = 1$). Par l'enchaînement de règles de probabilités, la probabilité jointe des autres nœuds du graphe est :



▶ Figure 1.10: Modèle de réseau inférentiel pour la RI

$$P(D, T_1, T_2, T_3, Q) = P(D) \times P(T_1|D) \times P(T_2|D, T_1) \times P(T_3|D, T_1, T_2) \times P(Q|D, T_1, T_2, T_3)$$

En supposant l'indépendance entre l'apparition d'un terme et celle des autres termes dans un document comme dans le modèle probabiliste basique, $P(T_i|D, T_1, T_2 \dots T_{i-1})$ devient équivalent à $P(T_i|D)$, l'équation devient :

$$P(D, T1, T2, T3, Q) = P(D) \times P(T_1|D) \times P(T_2|D) \times P(T_3|D) \times P(Q|T_1, T_2, T_3)$$

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

$$\begin{aligned} P(Q = 1|D = 1) &= \frac{P(Q=1, D=1)}{P(D=1)} \\ &= \frac{\sum_{(t_1,t_2,t_3) \in \{0,1\}^3} P(D=1, T_1=t_1, T_2=t_2, T_3=t_3, Q=1)}{P(D=1)} \end{aligned}$$

La mise en œuvre du modèle nécessite la connaissance de $P(D = [0|1])$, $P(Ti = [0|1]|D = [0|1])$ et $P(Q = [0|1]|(T_1, T_2 \dots T_n) \in \{0, 1\}^n)$.

Le nombre d'évènements $Q = [0|1]|(T_1, T_2 \dots T_n) \in \{0, 1\}^n$ augmente exponentiellement en fonction du nombre de termes de la requête (2^n). Turtle [130] [131] a identifié quatre formes canoniques de $P(Q|T_1, T_2 \dots T_n)$: *and*, *or* , *sum* et *wsum* pour résoudre le problème d'explosion combinatoire provoqué par l'augmentation du nombre de termes dans la requête. Le problème n'est résolu que partiellement, le temps exponentiel que demande le calcul des probabilités demeure l'inconvénient principal des réseaux bayésiens.
Le système Inquery [23] est l'un des modèles qui implémentent un réseau inférentiel bayésien. Il est utilisé pour formuler des requêtes simples basées sur des mots-clés, des requêtes booléennes ou ensemble.

Il propose des opérateurs de moyenne et de moyenne pondérée, des opérateurs booléens probabilistes ou stricts, des opérateurs de proximité et de synonymie. Une procédure de prétraitement de la requête permet de générer une forme inférentielle prête à être évaluée.

Les réseaux bayésiens de croyance : Les réseaux bayésiens de croyance (*belief network*), sont introduits par Ribeiro-Neto et Muntz [82]. Ils sont fondés sur une étude epistémologique[4] pour l'interprétation des probabilités. Mais ils s'écartent de l'inférence réseau par l'adoption d'un modèle bien défini, les réseaux de croyance [82] sont une généralisation des réseaux d'inférence.

1.4.3.4 Les modèles de langage

Par modèle de langue, on désigne une fonction de probabilité P qui assigne une probabilité $P(s)$ à un mot ou à une séquence de mots $s = m_1 m_2 \dots m_n$ en une langue [18] [77].

Cette fonction permet d'estimer la probabilité de générer cette séquence de mots à partir du modèle de la langue :

$$P(s) = \prod_{i=1}^{n} P(m_i | m_1 \dots m_{i-1})$$

Lorsque le nombre de mots dans la séquence est élevé, la probabilité de génération devient très faible. On utilise alors un modèle de langue n-gramme (on ne considère que les l prédécesseurs) : $P(m_i|m_1 \dots m_{i-1})$ devient $P(m_i|m_{i-l+1} \dots m_{i-1})$. Les modèles souvent utilisés sont les modèles uni-gramme, bi-gramme et tri-gramme :

- Unigramme : $P(s) = \prod_{i=1}^{n} P(m_i)$
- Bigramme : $P(s) = \prod_{i=1}^{n} P(m_i|m_{i-1}) = \prod_{i=1}^{n} \frac{P(m_{i-1}m_i)}{P(m_{i-1})}$
- Trigramme : $P(s) = \prod_{i=1}^{n} P(m_i|m_{i-2}m_{i-1}) = \prod_{i=1}^{n} \frac{P(m_{i-2}m_{i-1}m_i)}{P(m_{i-2}m_{i-1})}$

[4] Intrèprète la probabilité comme un degré de croyance dont les spécifications viennent des expérimentations statistiques.

Ce que l'on doit estimer sont les probabilités $P(m_i|m_{i-l+1}\dots m_{i-1})$. L'estimation ne peut se faire que par rapport à un corpus de textes C. Si le corpus est suffisamment grand, on peut faire l'hypothèse que le modèle de langue peut être approximativement le modèle de langue pour ce corpus. Selon les fréquences d'occurrences d'un n-gramme α, sa probabilité $P(\alpha|C)$ peut être directement estimée comme suit :

$$P(\alpha) = \frac{|C|_\alpha}{\sum\limits_{\beta\in C,|\alpha|=|\beta|} |C|_\beta} \approx \frac{|C|_\alpha}{|C|}$$

où $|C|$ est le nombre d'occurrences de mots dans le corpus, $|C|_w$ est le nombre d'occurrences de w dans le corpus et $|w|$ désigne la longueur de w.

La pertinence d'un document à une requête est en rapport avec la probabilité que la requête puisse être générée par le modèle de langue du document. Ainsi, on considère qu'un document d incarne un sous-langage, pour lequel on tente de construire un modèle de langue MD. Le score du document d à une requête $q = t_1t_2\dots t_n$ est déterminé par la probabilité que son modèle génère la requête :

$$\begin{aligned} rsv(d,q) &= P(Q|MD) \\ &= P(t_1t_2\dots t_n|MD) \end{aligned}$$

1.5 Evaluation des SRI

L'évaluation des SRI est un problème majeur sur lequel la communauté de la RI a investi beaucoup d'efforts [28] [50]. L'intérêt est de pouvoir comparer des SRI entre eux à l'aide des critères objectifs (les réponses idéales que l'utilisateur souhaite recevoir). Plusieurs quantités mesurables ont été proposées pour l'évaluation d'un SRI : le temps de réponse, la pertinence, la qualité et la présentation des résultats...

1.5.1 Hypothèses d'évaluation

Plusieurs hypothèses sont considérées ou faites dans les différentes mesures d'évaluation dont les principales sont :

- Présentation : les documents sont ordonnés par scores décroissants,
- Ordre de parcours : l'utilisateur parcourt la liste des documents en partant du premier jusqu'au dernier présenté (ne procède jamais de façon aléatoire),

- Jugement absolu : un document reste pertinent même s'il contient exactement la même information qu'un autre document déjà présenté à l'utilisateur,
- La non additivité : deux documents non pertinents ne pourront jamais former une unité d'information pertinente,
- Pertinence binaire : un jugement de pertinence doit pouvoir se ramener au mieux à un nombre réel généralement borné. Cette hypothèse est réduite à un jugement de pertinence binaire.

1.5.2 Mesures d'évaluation

Les mesures les plus courantes pour mesurer la performance d'un SRI sont le temps et l'espace. Les plus rapides en temps de réponse et les moins gourmands en espace utilisé sont les meilleurs SRI. Dans le cadre des SRI, on s'intéresse plutôt aux résultats retournés par ce dernier, sans négliger les deux premiers critères de performance. L'évaluation d'un SRI se mesure indépendamment de la méthode d'indexation ou du modèle qu'il l'implante. Pour cela, ces techniques s'appuient essentiellement sur l'estimation de la qualité des informations retrouvées par le SRI.
Ils existent plusieurs facteurs d'évaluation, les deux principaux facteurs permettant d'évaluer un SRI sont le *rappel* et la *précision*. Le rappel mesure la capacité du système à retrouver tous les documents pertinents et la précision mesure son habilité à ne retrouver que des documents pertinents. Ces mesures peuvent être quantifiées en pourcentage ou par des valeurs entre 0 et 1.
On désigne par R l'ensemble des documents pertinents et par L l'ensemble des documents retrouvés.

1.5.2.1 Le rappel

Le rappel mesure la capacité du SRI à sélectionner tous les documents pertinents. Il donne une indication sur le nombre de documents pertinents trouvés par rapport au nombre total de documents pertinents pour la requête. La valeur de rappel est entre 0 et 1 :

$$\text{rappel} = \frac{|R \cap L|}{|R|}$$

Le rappel mesure aussi la probabilité qu'un document d soit sélectionné sachant qu'il est pertinent :

$$\text{rappel} = P(d \in L | d \in R)$$

1.5.2.2 La précision

La précision mesure la capacité du système à rejeter tous les documents non pertinents. Elle donne une indication sur la proportion des documents pertinents renvoyés par le SRI. La précision est d'une part un indicateur pour la qualité du résultat à la demande de la recherche, et d'autre part, elle sert à ne pas distribuer des documents non pertinents afin de ne pas saturer la capacité d'un système. La valeur de la précision est entre 0 et 1. La précision se calcule alors par :

$$\text{précision} = \frac{|R \cap L|}{|L|}$$

La précision mesure aussi la probabilité qu'un document d soit pertinent sachant qu'il est sélectionné :

$$\text{précision} = P(d \in R | d \in L)$$

Le SRI n'a pas à décider de la pertinence d'un document : il range juste les documents de la collection selon leurs potentiels de répondre au besoin en information de l'utilisateur. La mesure de précision et de rappel ne peuvent alors avoir un sens que si l'on considère les x premiers documents de la liste ordonnée. La précision et le rappel ne sont alors que des mesures relatives de performance.

La précision exacte ou la R-précision La précision à x documents est souvent reliée à ce que l'on appelle la précision *exacte* ou la *R-precision*. La précision exacte représente celle obtenue à l'endroit où elle vaut le rappel. Si la requête admet x documents pertinents, la précision exacte est celle calculée pour les x premiers documents de la liste ordonnée des documents restitués (top x).

La précision moyenne On peut faire bouillir les valeurs informant sur la précision d'un SRI à quelques chiffres ou même à un seul chiffre. La précision moyenne consiste à interpoler la précision mesurée à différentes positions, nous calculons alors la moyenne arithmétique des valeurs interpolées de la précision. La précision moyenne est la moyenne des valeurs de précision à chaque document pertinent de la liste ordonnée. Elle tient compte à la fois de la précision et du rappel. En effet, si un document est pertinent et apparaît loin dans la liste, la précision à ce document et par conséquent la précision moyenne, tend à devenir nulle.

$$\text{précision moyenne} = \sum_{d \in R} \frac{\text{précision}(d)}{|R|}$$

où précision(d) est la précision calculée sur la base de tous les documents classés avant le document d :

$$\text{précision}(d) = \frac{|\{d' \in R, rsv(d', q) \geq rsv(d, q)\}|}{|\{d', rsv(d', q) \geq rsv(d, q)\}|}$$

1.5.2.3 Autres mesures d'évaluation

Le rappel et la précision, sont les mesures les plus populaires en RI, ces deux mesures représentent deux aspects qui sont différents pour les documents retrouvés. La combinaison de ces deux mesures d'évaluation peut être plus appropriée, pour cela deux mesures ont étés proposées : la mesure *Harmonique* et la mesure d'évaluation E.

La Moyenne harmonique La mesure de moyenne harmonique F est une mesure unique qui combine le rappel et la précision [54]. La moyenne harmonique est une fonction dans l'intervalle $[0, 1]$, elle est calculée comme suit :

$$F(j) = \frac{2}{\frac{1}{R(j)} + \frac{1}{P(j)}}$$

où $R(j)$ (resp $P(j)$) est le rappel (resp. la précision) aux j premiers documents dans le classement et $F(j)$ est la moyenne harmonique de $R(j)$ et $P(j)$. $F(j)$ vaut 0 quand aucun document pertinent n'a été retrouvé et 1 lorsque tous les documents renvoyés sont pertinents. Le détermination de la moyenne harmonique maximale peut être interprétée comme une tentative pour trouver le meilleur compromis possible entre le rappel et la précision.

La mesure E La mesure E est une autre mesure qui peut également combiner le rappel et la précision [83]. Le but de cette mesure est de permettre à l'utilisateur de spécifier laquelle des valeurs de précision ou de rappel est la plus intéressante. La mesure d'évaluation E est définie comme suit :

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

La variable b mesure l'importance relative de la précision ou du rappel, si $b = 1$, $E(j)$ serait alors le complément de la mesure Harmonique $F(j)$. Si b est supérieur à 1, alors on privilégie la précision, dans le cas contraire ($b < 1$) on privilégie le rappel.

Mesures orientées utilisateurs Le rappel et la précision sont basées sur l'hypothèse que l'ensemble des documents pertinents, pour la même session de recherche, est indépendant de l'utilisateur. Toutefois, certains utilisateurs pourraient avoir une interprétation différente. Pour faire face à ce problème Korhage a proposé la *mesure utilisateur*, dans ce contexte plusieurs mesures ont été proposées telles que la *ratio de couverture*, le *ratio de nouveauté ratio*, le *rappel relatif* et l'*effort rappel* [60] :

- *Ratio de couverture (Coverage ratio)* : est défini par la proportion des documents connus (à l'utilisateur) qui sont pertinents, et qui ont été effectivement renvoyés.

- *Ratio de nouveauté (Novelty ratio)* : est défini par la fraction des documents pertinents renvoyés et qui étaient inconnus à l'utilisateur.
- *Rappel relatif* : proportion des documents pertinents trouvés par le SRI par rapport au nombre de documents que l'utilisateur espérait récupérer.
- *Effort de rappel* : proportion des documents que l'utilisateur espérait récupérer et le nombre de documents analysés dans l'espoir de récupérer des documents pertinents.

D'autres mesures qui pourraient être d'intérêt, ces mesures sont généralement des mesures d'évaluation complémentaires au rappel et à la précision.

Le bruit La mesure d'évaluation bruit est une notion complémentaire à la précision, elle est définie par $B = 1 - P$ où P est la précision du SRI.
Cette mesure est dans l'intervalle $[0, 1]$. Elle tend vers 0 pour les meilleurs SRI.

Le silence La mesure d'évaluation silence est une notion complémentaire au rappel, elle est définie par $S = 1 - R$ où R est le rappel du SRI. Cette mesure est dans l'intervalle $[0, 1]$. Elle tend également vers 0 pour les meilleurs SRI.

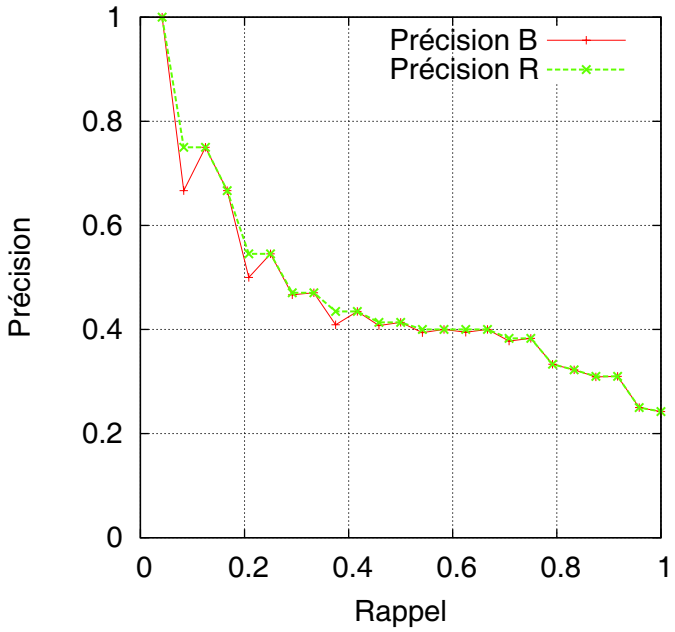
1.5.2.4 La courbe rappel-précision

Les deux facteurs rappel et précision ne sont pas indépendants, il y a une forte corrélation : quand l'un augmente, l'autre diminue. Il est nécessaire de prendre ces deux facteurs en compte car le rappel comme facteur unique qui ne prend pas en compte le nombre de documents non pertinents peut être maximisé en distribuant tous les documents de la base. Dans ce cas, la valeur de la précision serait très faible. De même, le seul regard de la précision ne mesure pas la performance des résultats. Le comportement d'un système peut varier en faveur de la précision ou en faveur du rappel, ainsi pour un système une courbe de rappel/précision qui a en général une forme exponentielle décroissante peut mieux donner un aperçu sur la résistance du système au bruit et au silence. La figure 1.12 illustre un exemple de l'allure générale de la courbe rappel/précision. Le tableau 1.3 illustre le résultat de jugement de l'utilisateur sur les 100 premiers documents retrouvés. Le symbole "$\surd$" indique que le document est jugé pertinent. Les résultats sont explorés par lots de 20, chaque lot est exploré sur une page. La courbe rappel-précision détermine la précision en fonction du rappel. Les points valeurs pour sa construction sont tout couple de valeurs (r, p) pour les i premiers documents classés et pour toutes les valeurs de i possibles (i va de 1 jusqu'au nombre de documents de la collection).

Page 1	Page 2	Page 3	Page 4	Page 5
√				
	√		√	
√	√			
√				
		√		
√				
	√	√		
			√	
	√			
√				
√			√	
				√
	√			
√	√			
√		√		
	√			
				√
	√			

► Tableau 1.3: Exemple de résultats de recherche et jugements de l'utilisateur

La construction d'une courbe exige que les abscisses soient tous différents, or les paires r ne sont pas toutes différentes, en effet, pour l'exemple illustré par le tableau 1.3, le rappel au premier et au deuxième document sont les mêmes ($\frac{1}{24}$ car un seul document pertinent parmi 24 est sélectionné pour $i = 1$ et $i = 2$). On retient pour chaque valeur de rappel la valeur de précision maximale, c'est-à-dire, la première valeur de précision observée pour chaque valeur de rappel. Il suffit de considérer les couples (r, p) pour chaque document pertinent retrouvé. La courbe **Précision B** de la figure 1.11 illustre cette construction. Cependant, elle ne correspond pas à la forme générale de la figure



▶ Figure 1.11: Représentation des points (rappel,précision)

1.12 : en effet elle n'est pas décroissante. On retient alors pour chaque valeur de rappel la valeur de précision la plus élevée parmi toutes les valeurs de rappel qui lui sont supérieures :

$$\text{Précision\_R}(r) = \max_{r' \geq r} \text{Précision\_B}(r')$$

Comme les niveaux de rappel ne sont pas unifiés pour l'ensemble des requêtes, car le nombre de documents pertinents varie d'une requête à une autre, on retient dans la littérature 11 points de rappel standards : de 0 à 1 par pas de 0.1 en admettant que $f(1) = 0$ et $f(0) = 1$. Les valeurs de précision relatives aux valeurs de rappel non identifiées peuvent être déterminées par interpolation linéaire. On considère dans ce cas les valeurs de rappel les plus proches :

$$\begin{aligned} r^+ &= \min_{r'>r} \text{Précision\_R}(r') \\ r^- &= \max_{r'<r} \text{Précision\_R}(r') \end{aligned}$$

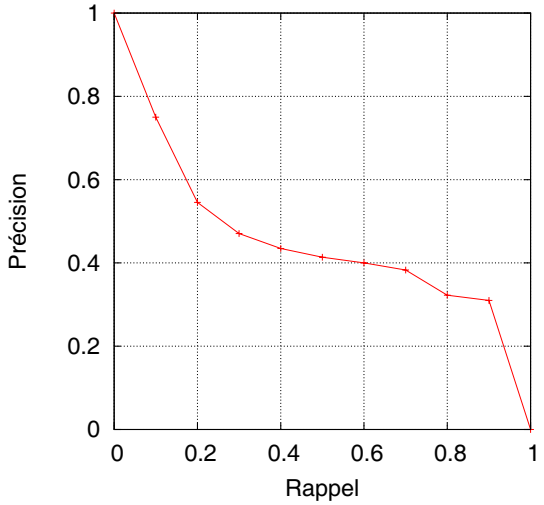
et les valeurs de précision relatives :

$$\begin{aligned} p^+ &= \text{Précision\_R}(r^+) \\ p^- &= \text{Précision\_R}(r^-) \end{aligned}$$

qui servent pour l'estimation de la précision définitive interpolée :

$$f(r) = \frac{r - r^-}{r^+ - r^-}(p^+ - p^-) + p^-$$

La figure 1.12 illustre la courbe de rappel-précision de l'exemple du tableau 1.3. Plus



▶ Figure 1.12: Courbe rappel-précision

la courbe est proche de la courbe $y = 1$ (la précision est maximale pour toutes les valeurs de rappel, ce cas est atteint lorsque tous les documents pertinents sont classés en tête de liste), plus le système est performent. La courbe de rappel-précision fournit la précision exacte. Celle-ci est obtenue par l'intersection de cette courbe avec la courbe représentative de la fonction identité ($g(x) = x$).
Pour comparer deux SRI, il faut les tester avec le même corpus de test (ou plusieurs corpus de test), les mêmes requêtes, on doit savoir aussi les résultats pour ces requêtes (jugements). Plusieurs mesures peuvent comparer les SRI. Le temps de réponse ou la présentation des résultats ne sont pas répandus à grande échelle, à cause de la difficulté de leur mise en œuvre. Les mesures basées sur les courbes de rappel-précision demeurent les plus largement utilisées par les bancs d'essai (benchmarks) les plus connus.
Ils existent de nombreux projets, dans ce cadre, on peut citer la collection **CACM**, la compagne **CLEF**[5] (Cross Language Evaluation Forum). Le plus célèbre étant **TREC**[6] (Text REtrieval Conference).

[5]http://www.clef-campaign.org/
[6]http://trec.nist.gov

1.5.3 TREC

TREC est un projet international qui a été lancé en 1992 par le **NIST** (*National Institute of Standards and Technology*) aux Etats-Unis. Il est aujourd'hui co-sponsorisé par le **NIST** et **DARPA/ITO** (*Defense Advanced Research Projects Agency - Information Technology Office*). Son objectif est de proposer un standard pour comparer les différents modèles de RI, indépendamment de la méthode de l'indexation ou bien du modèle qu'ils implémentent. Afin de mesurer l'efficacité des SRI de manière standard, il suffit de fournir une collection de test, des requêtes et les jugements à ces requêtes.

1.5.3.1 Collection de test

Dans ce cadre, il y a eu de nombreuses pistes sur une gamme d'essais différents, le plus connu est TREC *ad hoc*[7]. Les documents sont représentés par du texte intégral, ils sont faiblement structurés (titre, paragraphe...), ces documents sont sous forme électronique. La plupart de ces documents possèdent une longueur entre 300 mots (documents courts) et 3000 mots (documents longs).

1.5.3.2 Collection d'entraînement

Afin de caractériser l'habilité des systèmes à s'adapter aux goûts des utilisateurs, une gamme d'essais est également fournie pour l'entraînement. Généralement, elle est de très petite taille comparée à celle de test (de l'ordre de 10000 documents contre plus que 100000 pour le test).

1.5.3.3 Les requêtes (Topics)

Les requêtes représentent le besoin en information de l'utilisateur, mais dans TREC elle représente aussi un moyen pour évaluer un SRI. La forme des requêtes a notamment évolué. Au début la forme était très structurée, elle comportait des champs permettant la structuration des requêtes (un titre, le thème, une description et l'objet de la recherche). Pour alléger la forme des requêtes, seuls le titre et une description très brève (d'une phrase ou deux) sont conservés.

[7]La tâche ad hoc dans TREC évalue les performances des SRI sur des ensembles statiques de documents, seules les requêtes changent

1.5.3.4 Les jugements de pertinence

Parce que les collections de test sont si volumineuses, les jugements ont été recueillies uniquement pour les documents qui ont été classés parmi les premiers. La pertinence d'un document pour une requête est codée par une valeur binaire (pertinent ou non). L'affectation des valeurs de pertinence pour chaque document se fait donc par une personne, cette étape est très pénible vu le nombre important de documents dans la collection.

1.6 Conclusion

Dans ce chapitre nous avons présenté le processus de RI traditionnelle, et les modèles utilisés pour construire un SRI ; ainsi que les facteurs d'évaluation pour comparer les différents systèmes, qui n'exploitent que le contenu sémantique des documents.
L'apparition des ordinateurs a joué un rôle central dans le développement des SRI. Différentes tâches qui étaient manuelles ont été automatisées. L'utilisation massive de larges ressources d'information rend les techniques classiques incontournables pour la RI. Une des conséquences de cette évolution a été la structuration de l'information. Actuellement, Internet et le format **XML** ont favorisé l'essor des documents, par structurer les documents par des liens entre eux et par la structure interne des documents. Les SRI actuels s'intéressent à cette évolution. Nous décrivons dans le chapitre suivant les influences de la communauté des bases de données et les approches issues de la communauté de la RI pour permettre une utilisation effective de cette dimension supplémentaire qui est la structure.

Chapitre 2

Recherche d'information structurée

2.1 Introduction

L'objectif principal d'un SRI classique est de retrouver les documents dont le contenu est conforme à une requête donnée. Dans cette optique, les documents sont représentés par un ensemble de mots-clés décrivant leur contenu. La structure du document n'est pas prise en considération ni au niveau de la requête, ni au niveau de la réponse pour retourner les parties pertinentes : la réponse à une requête reste le document tout entier. Aujourd'hui, l'utilisation de l'information apportée par la structure devient une nécessité dans le domaine d'accès à l'information. Deux communautés se sont engagées à proposer des solutions pour la recherche d'information dans des documents structurés : la communauté de la RI qui s'intéresse à la structure présentée dans les documents pour apporter des réponses plus précises à une requête d'un utilisateur, et la communauté des bases de données qui s'intéresse à la structure comme un moyen plus souple pour stocker des données.

L'objectif principal sur lequel se focalisent ces deux communautés est un type de document qui est de plus en plus répandu sur Internet : le langage XML (*eXtensible Markup Langage*) est utilisé aujourd'hui comme format de données structurées sur le Web [21] [132], il impose au SRI de retrouver des unités d'information qui ne sont pas nécessairement le document entier.

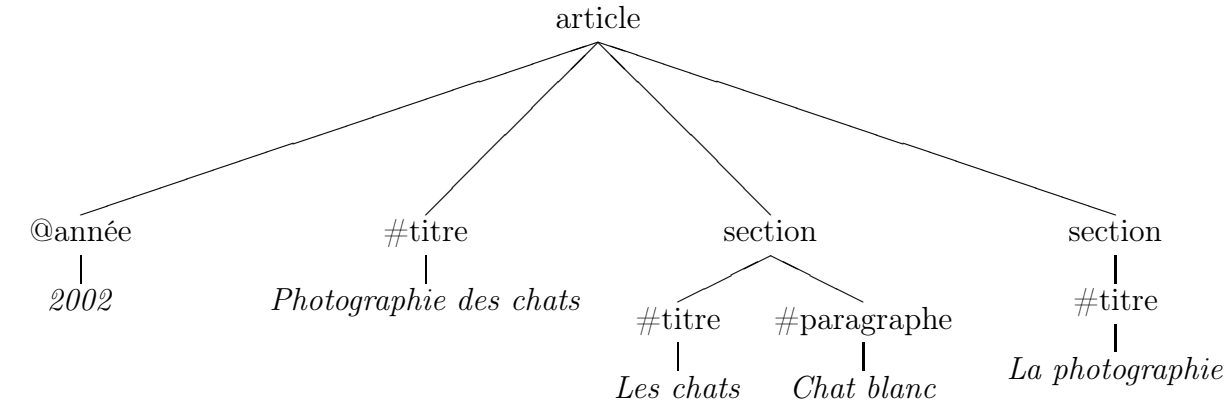
Nous présentons tout d'abord dans ce chapitre la définition des documents structurés, et plus particulièrement nous nous focalisons sur les documents XML [21] [81] [132], ensuite nous présentons les différentes problématiques soulevées par la RI dans ce type de documents, afin de donner les principales approches proposées dans la littérature. Enfin nous donnons un aperçu des méthodes d'évaluation pour la RI dans les

documents structurés.

2.2 Documents structurés : le langage XML

La notion de document a évolué pour passer du concept de document plat à un concept où le document est devenu un objet pouvant contenir plusieurs sources d'informations (vidéo, texte, images...). Pour exploiter ces sources d'une manière rigoureuse, le format de document est défini par une structure logique. Cette structure est décrite par plusieurs normes internationales (SGML, XML). Nous nous intéressons aux documents décrits par la norme XML. Ce dernier permet la représentation du contenu et de la structure du document d'une manière indépendante. La figure 2.1 illustre un exemple de document structuré au format XML.

```
<article annee="2002">
  <titre>
    Photographie des chats
    <section>
      <titre> Les chats </titre>
      <paragraphe> Chat blanc </paragraphe>
    </section>
  </titre>
  <section>
    <titre> La photographie </titre>
  </section>
</article>
```



▶ Figure 2.1: Exemple de document XML

2.2.1 La notion de structure

Le terme structure consiste à identifier et à décrire les différentes composantes textuelles et non textuelles qui constituent le document. En effet, nous distinguons en général deux types de structure : la structure physique qui traite les propriétés typographiques (police, taille, couleur...) associée à chaque élément du texte, et la structure logique qui décrit la nature des éléments et les relations hiérarchiques qui les relient. C'est à ce second type que se rapporte l'utilisation du terme structure dans les documents XML. Un document structuré au format XML est composé d'un ensemble d'éléments structurels. Ces éléments sont représentés par des balises qui servent à repérer les différentes parties ou sous parties du document (figure 2.1).

2.2.2 La notion de contenu

Le contenu d'un document structuré désigne le contenu textuel représenté sous la forme d'un ensemble de parties non structurées, comme par exemple des paragraphes ou des sections. Il existe deux grands types d'éléments répondant au contenu dans un document XML :

- Les attributs : les attributs sont préfixés par le symbole "@". Ils peuvent être associés seulement aux éléments structurels. Dans la figure 2.1 un seul attribut est présenté, l'attribut *année* dont la valeur est 2002 est associée à l'élément *article*.
- Les données : les éléments qui peuvent contenir des données sont marquées par le symbole "#".

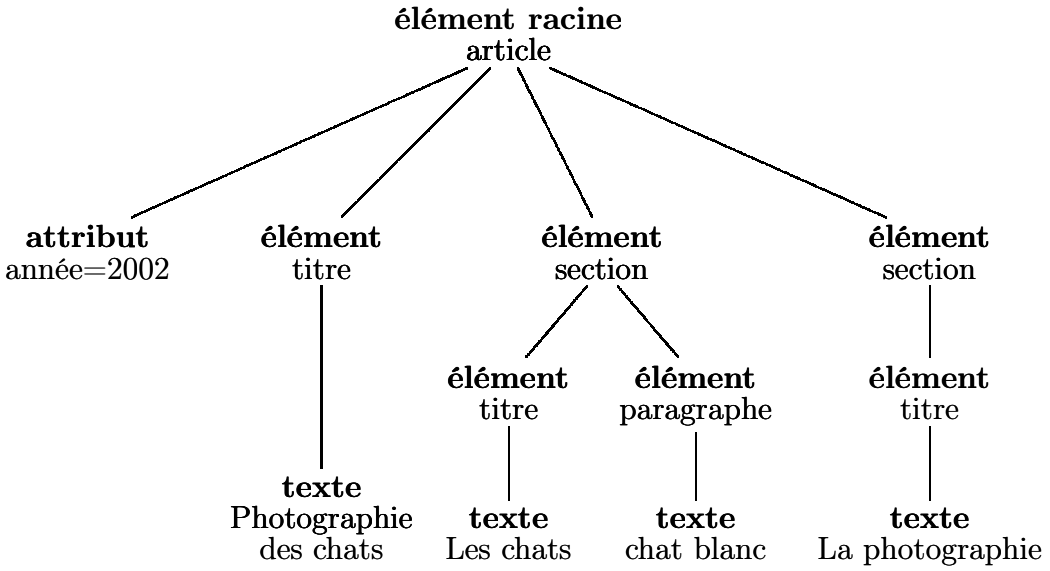
2.3 Les concepts fondamentaux de XML

La structure logique d'un document XML est un arbre. Chaque nœud de l'arbre est un élément XML et est décrit par une balise ouvrante et une balise fermante. Un élément peut avoir un ou plusieurs attributs XML dans l'arbre du document XML. XML est simplement un format de structuration de documents. Les dispositifs d'extraction de l'information diffèrent, plutôt que des fonctions le plus souvent de type SQL, XML adopte des analyseurs qui, en partant des structures des documents XML, peuvent renvoyer toute information ou élément d'information qui pourra intéresser l'utilisateur : ces analyseurs sont appelés *parseurs*.

Un parseur est une interface de programmation qui offre un grand nombre de fonctions d'accès aux données XML. Ils favorisent la navigation dans un document XML par des moyens de repérage aussi simples d'utilisation que puissants.

Il existe deux types d'analyseurs de documents XML, le parseur s'appuyant sur des flux d'évènements SAX (*Simple API for XML*) et le parseur DOM (*Document Object Model*) qui produit un graphe d'objets.

Le DOM[1] représente les éléments, les attributs et le texte des éléments au sein des nœuds d'un arbre comme illustre la figure 2.2. DOM est alors une interface de programmation qui représente les documents XML en mémoire sous forme d'arbre. Grâce à ses fonctions, DOM permet de consulter, modifier le contenu et la structure d'un document chargé en mémoire.



▶ Figure 2.2: Représentation du document XML comme un objet DOM

Il est facile avec l'API DOM de commencer à partir de l'élément racine c'est-à-dire des parents aux enfants. Il est recommandé pour se repérer efficacement dans un document XML, relativement à un élément de l'arbre XML. Si le besoin en information est exprimé selon un chemin XML absolu, il devient ardu d'utiliser DOM et d'avoir recours à d'autres standards.

XPath [26] est un standard pour l'énumération des chemins d'une collection de

[1] http://www.w3.org/DOM

documents XML. L'expression *XPath node* sélectionne tous les nœuds de ce nom. Les éléments successifs d'un chemin sont séparés par des barres obliques, *section/titre* sélectionne tous les éléments *titre* dont le père est *section*. Les doubles barres obliques indiquent qu'un nombre arbitraire d'éléments peuvent intervenir sur la voie : *article//titre* sélectionne tous les éléments *titre* subordonnés de l'élément *article*. Dans la figure 2.2 cet ensemble se compose de trois éléments *titre*, accessibles depuis l'élément *article*.

XPath est une spécification conçue pour parcourir une collection de documents XML, et de sélectionner un ensemble de nœuds en exploitant notamment les relations existantes entre ces derniers. Ces nœuds devront répondre à certaines contraintes structurelles ou sémantiques (contenu) pour être sélectionnés. Les contraintes sont sous la forme d'un chemin. L'utilisateur doit décrire des expressions de chemin dans l'arbre d'un document XML pour retourner des fragments de document.

2.4 Enjeux et problématiques de la recherche d'information structurée

Les documents XML ont réactualisé la problématique de la recherche d'information. En RI traditionnelle ou non structurée l'unité documentaire retournée à l'utilisateur est le document tout entier. Le premier défi de la recherche dans des documents XML concerne précisément cette notion d'unité documentaire, en effet dans le cas des documents XML, tout élément peut être renvoyé comme réponse à la requête. Il n'existe par conséquent pas d'unité normale (document entier). Si notre requête pour le document illustré par la figure 2.1 est *photographie*, nous pouvons renvoyer l'élément *titre*, l'élément *section* ou l'élément *article* : le résultat sera un granule de document (une partie du document).

2.4.1 Information recherchée

Un SRI structurée (SRIS) est censé définir un critère de décision pour choisir la partie la plus appropriée d'un document XML [109] : un tel système devrait toujours rechercher la partie la plus spécifique d'un document répondant à une requête. Ce principe motive une stratégie de récupération qui renvoie la plus petite unité contenant l'information recherchée, mais pas au-dessous de ce niveau. Cependant, il peut être difficile de mettre en application ce principe algorithmiquement. Considérons une requête qui cherche l'unité d'information contenant le mot *photographie*, dans l'exemple de la figure 2.1 nous avons deux éléments *titre* contenant ce mot. Mais dans ce cas, le titre qui contient seulement le texte *la photographie* est très spécifique [1] [118].

Cependant, les éléments retournés ne doivent pas manquer d'exhaustivité : si plusieurs éléments sont très spécifiques mais faiblement exhaustifs, ils peuvent être regroupés pour former une unité moins spécifique mais très exhaustive.

Décider quel niveau de l'arbre répond potentiellement à une requête est difficile. On doit grouper des nœuds, mais la problématique de ceci est que les parties du document peuvent ne pas avoir du sens pour l'utilisateur parce qu'elles ne sont pas des unités logiques.

En raison de la redondance provoquée par les éléments de petite taille, il est commun de limiter l'ensemble d'éléments qui sont éligibles pour être retournés. Pour ce faire les stratégies de restriction doivent :

- écarter tous les petits éléments,
- écarter tous les types d'éléments que les utilisateurs ne regardent pas,
- écarter tous les types d'éléments que les utilisateurs ne jugent généralement pas appropriés (si les évaluations de pertinence sont disponibles),
- garder seulement les types d'éléments qu'un concepteur ou un bibliothécaire de système a considéré utiles.

Dans la plupart de ces approches, les ensembles de résultats contiendront les éléments de petite taille. Ainsi, nous pouvons enlever quelques éléments dans une étape de post-traitement pour réduire la redondance. Alternativement, nous pouvons grouper plusieurs éléments de petite taille ensemble. Si des conditions sont accentuées dans le besoin en information, le balayage moyen des éléments (par exemple une section) prend un peu plus de temps que le balayage des éléments de plus petite taille (par exemple un paragraphe). Ainsi, si la section et le paragraphe se produisent dans la liste de résultats, il sont suffisants pour montrer la section. L'avantage de cette approche est que le paragraphe est présenté ainsi que son contexte (c'est-à-dire la section). Ce contexte peut être utile dans l'interprétation, le paragraphe (par exemple, la source d'information rapportée) est préservé si le niveau paragraphe seul satisfait la requête. Si l'utilisateur connaît le schéma de la collection et peut indiquer le type d'élément désiré, alors le problème de la redondance est toléré pourvu que quelques éléments nichés aient le même type.

2.4.2 Expression du besoin en information

Dans la RI structurée (RIS) les utilisateurs doivent indiquer les éléments qu'ils interrogent. Par conséquent l'interface de formulation est plus complexe qu'une boîte de

recherche de besoins en information composés de simples listes de mots-clés comme en RI classique. Nous pouvons également assister l'utilisateur en lui fournissant des suggestions en se basant sur les structures les plus fréquentes dans les documents de la collection.
Les requêtes contiennent d'une part l'information sur le contenu (comme dans la RI traditionnelle), mais peuvent aussi contenir l'information structurelle utilisée dans la RIS. Les langages de requête [33] [107] utilisés pour des requêtes structurées sont généralement basés sur XPath [26]. Les utilisateurs préfèrent néanmoins une interprétation détendue des contraintes structurelles, pour mieux cibler les unités restituées.

2.5 Approches de RI dans des documents XML

L'accès aux documents XML a été appréhendé selon deux principaux angles [38] : l'approche orientée données qui utilise les techniques issues de la communauté des bases de données (*data-centric*), et l'approche orientée document (*document-centric*) qui adapte les techniques développées par la communauté RI. Le tableau 2.1 illustre les principes de chaque communauté pour le traitement des documents structurés.

	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: SRI vs. SGBD

2.5.1 Les approches orientées données

Les approches orientées données réalisent en surcouche l'intégration de XML en une base objet relationnelle. La surcouche permet de transformer les documents XML en tables et inversement. Le but de ces approches est d'utiliser la richesse de la RI textuelle par mots-clés dans l'interrogation de la base de données. A ce stade, de nombreux langages de requêtes ont été développés pour interroger les documents XML comme Xpath et Xquery. Tous ces langages permettent d'intégrer des prédicats pour les termes de contenu et d'information structurelle des documents XML. L'avantage de ces approches est de traiter la structure des documents XML d'une manière efficace, mais elles sont limitées car elles traitent la partie textuelle de façon binaire (présent/absent), or il a été démontré en RI textuelle que la pondération des mots-clés est nécessaire pour

graduer la pertinence d'un document et par conséquent de renvoyer une liste ordonnée de résultats.

2.5.2 Approches orientées documents

Les approches orientées RI considèrent les documents XML comme une collection de documents textes possédant des balises et des relations entres ces balises. L'objectif principal pour cette communauté se focalise sur l'utilisation de l'information portée par la structure du document afin d'améliorer les résultats de recherche et changer la granularité de ces derniers (des éléments structurels au lieu du document tout entier). Les modèles de RI classiques (modèle booléen, vectoriel et probabiliste) ont été étendus pour mieux prendre en compte la coexistence du contenu et de la structure. Ils s'intéressent à un traitement efficace du contenu textuel porté par les éléments structurels dans les documents XML. La pertinence d'un document vis-à-vis d'une requête est une agrégation de l'ensemble des valeurs de pertinence assignées aux éléments pertinents pour une requête donnée.
La dimension structurelle est prise en compte conjointement avec la dimension textuelle. Dans plusieurs approches [127], le contenu textuel , qu'il soit pondéré ou non, subit une propagation des éléments feuilles vers les ascendants. Cette technique est mise en relief dans la perspective de faire apparaître dans chaque élément du contenu textuel en fonction de ses éléments descendants, et de valuer par la suite le score de chacun en fonction de ce contenu. Il existe une approche similaire qui inversement consiste à calculer un score dans les éléments feuilles et le propager vers les éléments ascendants [110].

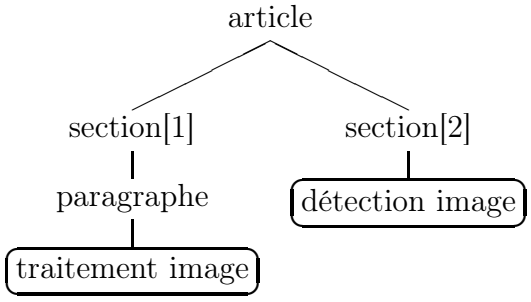
2.6 Techniques d'indexation des documents structurés

L'indexation des documents structurés semble plus complexe par rapport aux documents plats. En effet l'indexation des documents plats consiste seulement à extraire les mots représentatifs de chaque document (chapitre 1). L'indexation des documents semi-strcuturés quant à elle est plus complexe vue la co-existence de l'information textuelle et de l'information structurelle, donc indexer un document structuré revient à trouver un moyen représentant ces deux types d'information. L'un des principaux défis dans la RIS est de trouver une manière pour représenter l'information structurelle dans un document XML, afin de pouvoir exploiter cet index dans le calcul de pertinence entre un document XML et une requête. Subséquemment, l'information textuelle doit être représentée en fonction de l'information structurelle. Dans cette section nous présentons les différentes approches proposées dans la littérature pour répondre à la problématique de l'indexation des documents structurés.

2.6.1 Indexation de l'information textuelle

L'information textuelle dans les documents XML se localise dans les nœuds feuilles (de type #PCDATA), pour les approches bases de données ils considèrent que ces nœuds possèdent du texte, contrairement aux approches orientées document, où le terme est pondéré afin de refléter son importance.
Le problème d'indexation de l'unité textuelle est de trouver le lien de cette information avec celle de l'information structurelle, autrement dit de quelle manière doit-on organiser l'information structurelle et l'information textuelle pour pouvoir identifier l'élément pertinent répondant à la requête. on distingue deux visions pour indexer l'information textuelle dans les documents XML selon les approches RI, l'information textuelle est traitée simultanément ou indépendamment de l'information structurelle.

Propagation des termes Les approches de ce courant concrétisent cette relation en propageant le texte des nœuds feuilles à leurs ancêtres [1] [55] [118]. Généralement, on calcule un poids pour chaque terme dans le nœud qui le contient, puis ce terme est propagé vers les nœuds ancêtres en diminuant son poids selon la distance entre le nœud qui le contient et celui vers lequel le terme est propagé [106].



▶ Figure 2.3: Exemple de document XML

Unités disjointes Les approches issues de ce courant ignorent à l'étape d'indexation la relation structurelle entre les différents nœuds du document XML. Elles considèrent que les nœuds sont des unités disjointes [4] [36] [41] [59] [73] [90].

2.6.2 Indexation de l'information structurelle

On peut distinguer selon le mode de représentation des éléments des documents [68] les méthodes suivantes :

L'indexation par les champs Pour chaque terme du document, on indique les nœuds qui le contiennent [45]. Donc pour chaque champ ou balise on localise l'information textuelle. Avec cette méthode on filtre, au moment de la recherche, les balises contenant le texte en question. Le tableau 2.2 illustre le résultat d'indexation du document illustré par la figure 2.3.

terme	**champs**
traitement	paragraphe
image	paragraphe
détection	section2
image	section2

▶ Tableau 2.2: Indexation basée sur des champs

L'indexation par les chemins Au lieu d'indiquer le nom de balise pour localiser l'information textuelle, l'indexation par les chemins remplace le nom de la balise par le chemin de l'élément basé sur XPath [49] [57] comme illustre le tableau 2.3. Ceci accélère le processus de recherche pour une information se situant dans plusieurs balises qui portent le même nom. L'inconvénient majeur de cette méthode est qu'elle ne permet pas de décrire la relation de descendance des différents éléments du document XML.

terme	**chemins**
traitement	/article/section1[1]/paragraphe[1]
image	/article/section1[1]/paragraphe[1]
détection	/article/section2[1]
image	/article/section2[1]

▶ Tableau 2.3: Indexation basée sur les chemins

L'indexation par les arbres Cette technique d'indexation ressemble à l'indexation par les chemins [64], mais contrairement à cette dernière elle permet d'assigner pour chaque nœud des valeurs de pré-ordre et post-ordre pour distinguer la relation ancêtre-descendant (hiérarchiques) [117]. Le tableau 2.4 illustre un exemple d'indexation basée sur les arbres. La structure d'index de XPath Accelerator [44] permet de représenter l'arborescence du document XML, en assignant pour chaque nœud des valeurs croissantes de pré-ordre ou post-ordre comme illustre la figure 2.3.

terme	arbres
traitement	/article/section1[1]/paragraphe[1] **(3)**
image	/article/section1[1]/paragraphe[1] **(3)**
détection	/article/section2[1] **(4)**
image	/article/section2[1] **(4)**

▶ Tableau 2.4: Exemple d'indexation basée sur les arbres

2.7 Technique de stockage et d'interrogation des documents XML

La manipulation des documents XML fait actuellement l'objet d'effort soutenu. Trois types de solutions sont possibles pour stocker et interroger des collections de documents XML [126] : utilisation des fichiers textes, utilisation des SGBD relationnels et utilisation d'un SGBD XML natif.

2.7.1 Modèle de fichiers textes

Les fichiers textes constituent le moyen le plus simple de stocker les documents XML. Ils présentent l'avantage de pouvoir être lus et édités par un utilisateur. Ce format constitue de plus le moyen d'échange le plus simple des données XML sur un réseau. Pour l'interrogation, XQuery [33] permet d'interroger ces documents après une traduction préalable sous forme d'un arbre d'objets en mémoire selon le standard DOM.

2.7.2 Modèle de SGBD relationnels

Les principaux SGBD relationnels (Oracle, SQL server...) ont été étendus pour les données XML. Deux méthodes de stockage existent :

- définir un nouveau type de données adapté à XML et stocker les documents XML comme des objets dans une colonne.
- réaliser une correspondance entre un document XML et un ensemble de tables en s'appuyant sur le DTD du document (destruction du document XML afin de stocker les éléments et les attributs en colonnes de tables).

Les documents stockés peuvent êtres manipulés en SQL par un jeu de fonctions prédéfinies, par exemple l'extraction des objets par une expression XPath.

2.7.3 Modèle de SGBD XML natifs

Les SGBD natifs sont développés spécifiquement pour XML [10]. Ils stockent et manipulent directement des arbres XML au lieu de passer par une structure intermédiaire (table relationnelle). Ils possèdent des index spécialisés permettant d'accéder aux composants d'un arbre de documents XML : éléments, attributs et texte. Les langages d'interrogation pour ce modèle sont les langages de requête XPath et XQuery.

2.8 Modèles d'interrogation de documents XML

Les modèles classiques de RI ont été adaptés à la RIS en tenant compte de la dimension structurelle. Indépendamment des modèles de recherche, l'appariement est effectué selon deux approches différentes :

- soit au niveau des éléments restitués grâce à une propagation de termes qu'ils soient pondérés ou non.
- soit au niveau de la plus petite unité d'indexation. Dans ce cas les éléments sont restitués grâce à une propagation de pertinence.

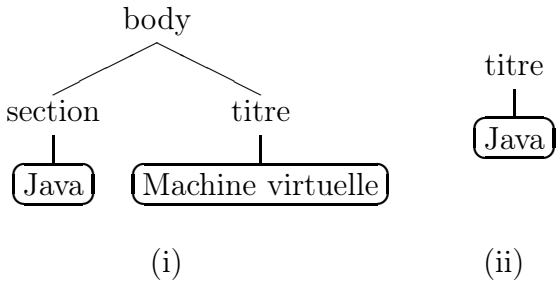
Nous présentons dans cette section un extension de chaque modèle présenté en 1.4 pour répondre aux besoins et problématiques de le RIS.

2.8.1 Le modèle vectoriel étendu

Le modèle vectoriel étendu [35] est une extension du modèle vectoriel qui sépare l'information apportée par la structure de l'information de contenu. La prise en compte de l'information structurelle est mise en relief par l'identification d'un espace vectoriel dans lequel les documents (ou les éléments de chaque document) et les requêtes peuvent être représentés par des vecteurs, ceci revient à identifier une base dans laquelle chaque vecteur élément ou requête XML reçoit des coordonnées scalaires. Le fondement du modèle vectoriel étendu repose sur la représentation de chaque dimension par un sous-arbre lexical. Un sous-arbre lexical est un chemin XML où la feuille est un terme d'indexation, et les nœuds internes sont des noms de balises XML. Si nous créons une dimension distincte pour chaque sous-arbre lexical qui apparaît dans la collection, la dimension de l'espace devient très grande mais de nombreuses dimensions auront peu d'utilité car ils apparaissent généralement dans un document unique.
Nous pouvons traiter le plan structurel par les sous-arbres lexicaux restants, tout

comme les termes d'indexation dans le modèle vectoriel classique.

Cela signifie que nous pouvons utiliser le formalisme de l'espace vectoriel pour la recherche XML. La différence est que les dimensions de l'espace vectoriel dans la recherche non structurée sont les termes alors qu'ils sont des sous-arbres dans la recherche structurée. Si on restreint la dimension structurelle, on se retrouve avec le modèle vectoriel traditionnel.
Nous pouvons maintenant représenter les chemins XML, qu'ils soient dérivés des requêtes ou des documents, en tant que vecteurs dans cet espace vectoriel et calculer des similarités entre eux. Une mesure simple de la similitude d'un chemin c_d dans un



► Figure 2.4: La représentation (i) représente un document tandis que la représentation (ii) représente une requête

document et d'un chemin c_q dans une requête est calculée par la fonction c_r suivante :

$$c_r(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{si } c_d \text{ peut être produit à partir de } c_q \text{ en insérant quelques éléments} \\ 0 & \text{si non} \end{cases}$$

où $|c_q|$ et $|c_d|$ sont respectivement le nombre de chemins dans la requête et dans le document. La valeur de c_r du document et de la requête qui sont représentés respectivement par les représentations (i) et (ii) illustré par la figure 2.4 est égale à $\frac{3}{4} = 0.75$. Ainsi le score final du document en utilisant la mesure du cosinus est donné par :

$$sim\_c_r(q,d) = \sum_{c_k \in C} \sum_{c_l \in C} c_r(c_k, c_l) \sum_{t \in V} poids(q,t,c_k) \frac{poids(d,t,c_l)}{\sqrt{\sum_{c \in C, t \in V} poids^2(d,t,c)}}$$

où V est le vocabulaire des dimensions non structurelles, C est l'ensemble de tous les chemins de l'arbre XML et $poids(q,t,c)$ et $poids(d,t,c)$ sont respectivement les poids d'un terme t dans un composant XML c dans la requête q et le document d. On peut utiliser les mesures de $tf \times idf$ pour calculer le poids d'un terme t.

A cause de la structure des documents, les différents types de modèles ont été étendus de diverses façons. Ainsi il faut prendre en compte les paramètres supplémentaires

qui sont apparus avec la RIS. Pour ce faire il faut donc ajuster les formules classiques et injecter des paramètres de structure [92] tels que la profondeur du document, le nombre d'enfants d'un élément...

2.8.2 Le modèle booléen étendu

Pour permettre l'expression des requêtes plus puissantes permettant de spécifier les relations entre les termes de l'index, le modèle booléen a été étendu avec un nouvel opérateur binaire non commutatif **contains**. Le premier opérande est de type XPath et le second est une expression booléenne. Ce modèle permet aux requêtes d'être complètement spécifiées en termes de contenu et d'information structurelle basée sur le langage d'interrogation XPath.
La recherche consiste à extraire le titre et le convertir en requête booléenne, les éléments considérés comme pertinents sont par la suite classés selon la somme OkapiBM25 [93]. Larson et al. [63] utilisent une combinaison de méthodes probabilistes utilisant une réegression logistique avec une approche basée sur le modèle booléen, pour évaluer la pertinence des documents et des éléments. La valeur de probabilité de pertinence R d'un composant C (élément) est calculée comme étant le produit des probabilités de la pertinence de C vis-à-vis la requête $Qbool$ présentée par un modèle booléen et de la pertinence de C vis-à-vis la requête $Qprob$ présentée par un modèle probabiliste :

$$p(R|Q,C) = P(R|Qbool,C) \times P(R|Qprob,C)$$

Cette combinaison permet de restreindre l'ensemble des documents pertinents aux documents ayant une valeur booléenne égale à 1 tout en leur attribuant un rang.

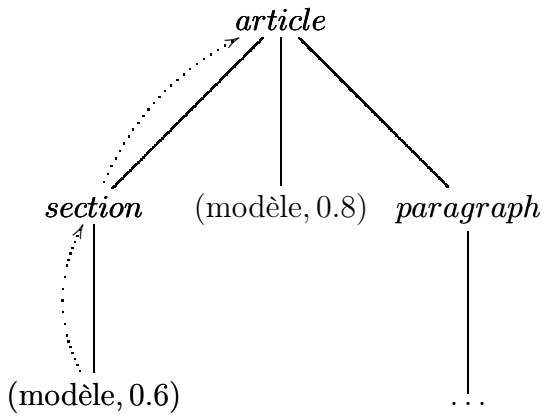
2.8.3 Modèle probabiliste

Dans le modèle probabiliste [83], le classement des documents est basé sur la probabilité que le document retrouvé d implique la requête donnée q. Pour étendre le modèle probabiliste aux documents XML, les probabilités doivent tenir compte de l'information structurelle. Deux approches ont été développées.
La première approche permet d'utiliser des probabilités conditionnelles de jointure, par exemple $P(d|t)$ devient $P(d|p \ contains \ t)$ où d représente un document ou une partie du document, t est un terme et p est un chemin dans l'arbre XML, la deuxième permet d'étendre la logique propositionnelle à la logique prédicationnelle et prend en compte les problèmes liés à la structure : cette approche est basée sur la définition des relations entre les tuples des tables d'une base de données, et la modélisation des prédicats avec des formules logiques. Comme pour les modèles ensemblistes, cette formalisation ne

permet pas d'ordonner une liste de documents et n'accorde aucune place à l'imprécision de la formulation de la requête. Ce problème est résolu par Fuhr [38] qui s'est intéressé à la RI dans des bases de données. Il a proposé de combiner les approches de RI et les bases de données. Il propose une algèbre relationnelle probabiliste qui est une généralisation de l'algèbre relationnelle. Cette algèbre a pour but d'assigner des poids probabilistes aux tuples d'une relation. Ces poids donnent la probabilité qu'un tuple appartient à une relation. Cette approche présente deux avantages : elle permet de représenter les valeurs de données avec imprécision et de classer les documents par leurs poids probabilistes.
Cette méthode est basée sur le langage de requête XIRQL [37], et a été implémentée au sein du moteur de recherche HyRex. Les termes sont propagés jusqu'au nœud indexable le plus proche. Le poids de pertinence des nœuds est calculé grâce à la propagation des poids des termes dans l'arbre du document. Le poids de chaque terme diminue par multiplication par un facteur nommé *facteur d'augmentation*.
Considérons la structure du document illustré par la figure 2.5, nous attribuons à chaque terme un poids selon sa probabilité d'apparition dans un nœud. Nous voulons calculer le poids du terme *modèle* dans l'élément racine du document présenté par la figure 2.5 (article) :

$$P\left([article] \wedge [0.6, modèle] \vee [modèle, 0.8]\right)$$

Par la suite en introduisant le facteur d'augmentation (0.7) on aura :



▶ Figure 2.5: Modèle d'augmentation

$$\begin{aligned} P\left([article] \wedge [0.6, modele] \vee [modele, 0.8]\right) &= P([article, modele]) + P([section, modele]) \\ &\quad \times P([section]) - P([article, modele]) \\ &\quad \times P([section]) \times P([section, modele]) \\ &= 0.6 + 0.7 \times 0.8 + 0.6 \times 0.7 \times 0.8 \\ &= 0.824 \end{aligned}$$

Pour des requêtes contenant des conditions structurelles, des sommes pondérés de ces probabilités sont cumulés.

2.8.4 Le modèle inférentiel

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

Piworwarski et al. [75] ont proposé un modèle probabiliste basé sur les réseaux bayésiens où les dépendances de hiérarchisation sont exprimées par des probabilités conditionnelles. La probabilité de pertinence d'un élément e sachant son parent p pour une requête q est $P(e|p,q)$ estdonnée par :

$$P(e = a|p = b, q)\frac{1}{1 + e^{F}_{e,a,b(q)}}$$

où $F_{e,a,b(q)}$ est la pertinence de l'élément e selon le modèle Okapi. Une requête q structurée est décomposée en un ensemble de n sous-requêtes élémentaires q_i. Chacune de ces sous-requêtes reflèete une entité structurelle et un besoin en information.

Le score final est donné par :

$$rsv(e_i, q) = rsv_{q_1}(e_i, q) \times rsv_{q_n}(e_i, q)$$

De Compos, Fernandez et Huete [24] ont également proposé un modèle de recherche basé sur les réseaux bayésiens où le diagramme d'inférence est basé sur la probabilité conditionnelle. Deux types de diagrammes sont proposés : SID (*Simple Inference Diagram*) et CID (*Context based Inference Diagram*). Un diagramme se compose d'un composant qualitatif (représentation des variables et des inférences) et un composant quantitatif (probabilités des nœuds).

2.8.5 Modèles de langage

Sigurbjornsson et al. [118] ont proposé un modèle combinant des modèles de langage de l'élément, du document et de la collection. Pour estimer les modèles de langage, les auteurs ont utilisé deux types d'index : un index pour les éléments du document XML qui assure la même fonction qu'un fichier inverse en RI classique et un autre (index article) pour tout le document utilisé pour des calculs statistiques.

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

$$P(e|q) \propto P(e) \times P(q|e) \tag{2.1}$$

Les auteurs considèrent l'indépendance entre les termes de la requête, l'équation 2.1 devient alors :

$$P(e|q) \propto P(e) \times \sum_{i=1}^{k} P(t_i|e)$$

où t_i est un terme de la requête.

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

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

où $P_{mle}(t-i|e)$ est la probabilité de t_i dans le modèle de langage de l'élément estimée par les statistiques à partir de l'index des éléments, $Pmle(t_i|d)$ est la probabilité de t_i dans le modèle de langage du document estimée par les statistiques à partir de l'index article et $Pmle(t_i)$ est la probabilité de t_i dans le modèle de langage de la collection. λ_e et λ_d sont des constantes.

2.8.6 Le modèle XFIRM

L'un des modèles de RIS qui se base sur le modèle vectoriel est le modèle XFIRM [108] [110], il est basé sur un modèle de données générique permettant l'implémentation de nombreux modèles de RI et le traitement de collections hétérogènes (c'est-à-dire contenant des documents ne suivant pas la même DTD).

Le traitement des requêtes est effectué en deux temps : une première étape consiste à évaluer la similarité des nœuds feuilles de l'index à la requête (on parle alors de calcul des poids des nœuds feuilles) et une seconde étape consiste à rechercher les sous-arbres pertinents. La pertinence des sous-arbres est évaluée en effectuant la propagation des poids des feuilles dans l'arbre du document.

Les requêtes composées de conditions de structure sont décomposées en requêtes élémentaires de type *nom_ element[condition contenu]* et chacune de ces requêtes est

traitée de manière indépendante : on évalue la similarité des nœuds feuilles à la condition de contenu et une première propagation est effectuée pour répondre à la contrainte de structure.

Calcul du score des nœuds feuilles Les scores des nœuds feuilles identifiés dans l'arbre du document sont calculés grâce à la fonction de similarité $rsv(q, nf)$.

Si la requête est composée de termes et des poids associés, on a :

$$rsv(q, nf) = \sum_{i=1}^{T} w_i^q \cdot w_i^{nf},\ avec\ w_i^q = tf_i^q \quad et \quad w_i^{nf} = tf_i^{nf} \times ief_i \times idf_i \tag{2.2}$$

où w_i^q et w_i^{nf} sont respectivement le poids du terme i dans la requête q et le nœud feuille nf, avec tf_i^q et tf_i^{nf} sont respectivement la fréquence du terme i dans la requête q et dans le nœud feuille nf, $idf_i = log(\frac{|D|}{|d_i|})$ permet d'évaluer l'importance du terme i dans la collection de documents, où $|D|$ est le nombre total de documents de la collection et $|d_i|$ est le nombre de documents contenant i, et $ief_i = log(\frac{|F_c|}{|nf_i|})$ permet d'évaluer l'importance du terme i dans la collection de nœuds feuilles, où $|F_c|$ est le nombre total de nœuds feuilles de la collection, et $|nf_i|$ est le nombre de nœuds feuilles contenant i.

Propagation de la pertinence des nœuds feuilles Une valeur de pertinence est ensuite calculée pour chaque nœud de l'arbre de document, en utilisant les poids des nœuds feuilles qu'il contient [111]. Les termes apparaissant près de la racine d'un sous-arbre paraissent plus porteurs d'information pour le nœud associé que ceux situés plus bas dans le sous-arbre. Il semble ainsi 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 $dist(n, nf_k)$, qui représente la distance entre le nœud n et un de ses nœuds feuille nf_k dans l'arbre du document, c'est-à-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. Le paramètre $|F_n^p|$, qui est le nombre de nœuds feuilles descendants de n ayant un score non nul est alors introduit dans la fonction de propagation. Une première évaluation de la pertinence p_n d'un nœud peut être calculée selon la formule 2.3 :

$$p_n = |F_n^p| \times \sum_{nf_k \in F_n} \alpha^{dist(n,nf_k)-1} \times (rsv_m(q, nf_k)) \tag{2.3}$$

où F_n est l'ensemble des nœuds feuilles nf_k descendants de n, et $\alpha \in]0 \dots 1]$ est un paramètre permettant de quantifier l'importance de la distance séparant les nœuds dans la formule de propagation. Deux propositions sont également faites pour évaluer de manière plus juste l'informativité des nœuds résultats:

- l'utilisation du paramètre β pour augmenter l'importance des nœuds de petite taille durant la propagation (nœuds supposés être utilisés pour faire ressortir des informations importantes, comme par exemple les nœuds *titres* qui permettent de situer avec précision le sujet de leur nœud ancêtre):

$$\beta(nf_k) = \begin{cases} \frac{l_k}{\Delta l} si\ dist(n, nf_k) = 1\ et\ l_k < \Delta l \\ log(\frac{\Delta l}{l_k})\ si\ dist(n, nf_k) > 1\ et\ l_k < \Delta l \\ 1\ sinon \end{cases} \quad (2.4)$$

  avec Δl la taille moyenne d'un nœud feuille de la collection;

- l'utilisation du contexte des éléments (c'est-à-dire de la pertinence du documents qui les contient) : un nœud appartenant à un document fortement pertinent doit être mieux classé qu'un nœud se trouvant dans un document de pertinence moindre.

La pertinence p_n d'un nœud n est alors définie de la façon suivante:

$$\begin{aligned} p_n &= \rho \times |F_n^p| \times \sum_{nf_k \in F_n} \alpha^{dist(n,nf_k)-1} \times \beta(nf_k) \times rsv(q, nf_k) \\ &\quad +(1-\rho) \times p_{racine} \end{aligned} \quad (2.5)$$

avec F_n et F respectivement l'ensemble des nœuds feuilles nf_k descendants de n et l'ensemble des nœuds feuilles nf_k du document, $|F_n^p|$ et $|F^p|$ respectivement le nombre de nœuds feuilles descendant de n ou du document et ayant un score non nul, $rsv(q, nf_k)$ calculé d'après 2.2, $\beta(nf_k)$ calculé d'après 2.4 et $\rho \in [0 \dots 1]$ est un paramètre servant de pivot et permettant d'ajuster l'importance de la pertinence du nœud racine lors de la rétropropagation. Les nœuds sont ensuite renvoyés à l'utilisateur par ordre décroissant de pertinence à la requête [112] [48].

2.9 Evaluation des SRIS

L'évaluation de la performance d'un SRIS, se base généralement sur les mesures de rappel et de précision (décrites dans la section 1.5.2). Pour l'évaluation de la RI dans les documents XML, il existe à l'heure actuelle une seule compagne d'évaluation : INEX (*INitiative for the Evaluation of XML retrieval*).

2.9.1 Compagne d'évaluation INEX

INEX est un programme qui produit des collections, des ensembles de requêtes, et des jugements de pertinence. Le colloque annuel d'INEX est tenu pour présenter et

discuter les résultats de recherche. Cette compagne a commencé en 2002, le tableau 2.5 illustre l'évolution des données dans la compagne d'évaluation INEX. Il est à noter que la tâche CO (*Content Only*) est toujours présente dans le porgramme INEX.

année	collection	Tâche orientée contenu et structure
2002	12107 (articles)	VCAS, SCAS
2003	12107 (articles)	VCAS, SCAS
2004	12107 (articles)	VCAS
2005	16819 (articles)	CO+S, VVCAS, SSCAS, VSCAS et SVCAS
2006	659388 (articles)	CO+S
2007	659388 (articles)	CO+S

▶ Tableau 2.5: Evolution des collections et des tâches de la compagne d'évaluation INEX pour les requêtes orientées contenu et structure

2.9.1.1 Collection de test

La compagne d'évaluation INEX fournit une collection de documents balisés au format XML. La collection contient 12107 articles, d'environ 500 Mo, publiés de 1995 à 2002, cette collection a été utilisée lors des évaluations des années 2002, 2003 et 2004. En 2005 la collection a été enrichie, elle comporte 16819 articles couvrant la période de 1995 à 2004 et totalisant environ 750 Mo. La figure 2.6 ilustre un extrait d'un document de la collection issue d'INEX'2005.
La DTD comprend 192 balises différentes et un article est composé en moyenne d'environ 1500 nœuds, et est de profondeur moyenne de 6.9.

2.9.1.2 Les requêtes (Topics)

En 2003, pour exprimer les besoins en information, XPath [26] est le formalisme qui a été utilisé. Ce formalisme était complexe ; le taux d'erreurs dans les requêtes atteint 63%. En 2004 le langage NEXI (*Narrowed Extended XPath I*) [129] a été proposé comme formalisme pour l'expression du besoin, le taux d'erreurs a diminué à 12%. En 2005 le langage NEXI a été de nouveau adopté pour l'expression du besoin en information [119].
Deux types de besoins ou de matières de l'information sont proposés dans INEX'2004 : contenu seulement (requêtes de type CO, *Content Only*), la figure 2.7 illustre un exemple de ce type de requêtes, des requêtes comportant des informations sur le contenu et la structure (type CAS, *Content And Structure*), la figure 2.8 illustre un extrait de ce type de requêtes.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE xmlarticle (View Source for full doctype...)>
 <article>
  <fno>A4006</fno>
  <doi>10.1041/A4006s-1995</doi>
<fm>
<hdr>
<hdr1>
  <ti>IEEE Annals of the History of Computing</ti>
<crt>
  <issn>1058-6180</issn>
  /95/4.00
<cci>
  <onm©> 1995 IEEE</onm>
  </cci>
  </crt>
  </hdr1>
<hdr2>
<obi>
  <volno>Vol. 17</volno>
  <issno>No. 4</issno>
  </obi>
<pdt>
  <mo>Winter</mo>
  <yr>1995</yr>
  </pdt>
  <pp>pp. 6</pp>
  </hdr2>
  </hdr>
  ...
```

► Figure 2.6: Exemple de document XML de la compagne d'évaluation INEX'2005

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE inex_topic (View Source for full doctype...)>
<inex_topic topic_id="190" query_type="CO" ct_no="136">
  <title>"e-commerce" "e-business" "data warehouse"</title>
  <description>We want information about creating a data
     warehouse for the e-business and
 the e-commerce section.</description>
  <narrative>Development of a data warehouse includes
     development of systems to extract data from operating
systems plus installation of a warehouse database system
   that provides managers flexible access to the data.
   Supply cha
in applications and the integration of data provided
   through the internet are critical to competitive success
   .
Companies probably will have to deploy advanced technology
   efficiently, such as data warehousing techniques,
 and use its capabilities to the fullest extent. We are
    looking for basic information plus already existing
    implementations
 to work out several possible applications of data
    warehousing in a medium-sized company. A relevant
    document
should contain information about how to strengthen the e-
   business value chain by applying data warehousing
 technologies.</narrative>
<keywords>data warehouse, OLAP, e-business</keywords>
  </inex_topic>
```

► Figure 2.7: Exemple de requête de type CO issue de la compagne d'évaluation INEX'2004

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE inex_topic (View Source for full doctype...)>
<inex_topic topic_id="127" query_type="CAS" ct_no="13">
  <title>//sec//(p| fgc)[about( ., Godel Lukasiewicz and
     other fuzzy implication definitions)]</title>
  <description>Find paragraphs or figure-captions
     containing the definition of Godel, Lukasiewicz or
     other
     fuzzy-logic implications</description>
  <narrative>Any relevant element of a section must contain
      the definition of a fuzzy-logic implication operator
     or
 a pointer to the element
 of the same article where the definition can be found.
    Elements containing criteria for identifying or
    comparing fuzzy
 implications are also
 of interest. Elements which discuss or introduce non
    implication fuzzy operators are not relevant.</
    narrative>
<keywords>Godel implication, Lukasiewicz implication, fuzzy
    implications, fuzzy-logic implication</keywords>
</inex_topic>
```

► Figure 2.8: Exemple de requête de type CAS issue de la compagne d'évaluation INEX'2005

- 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 champ *title* : indique le besoin en information sur le contenu ou sur le contenu et la structure.
- Le champ *description* : donne une courte description du besoin en information.
- Le champ *narrative* : donne une description plus détaillée du besoin en information.
- Le champ *Keywords* : est un ensemble de mots-clés de la requête en question.

En 2005, un nouveau type de requête CO+S (*Content Only + Structure*) a été introduit. La figure 2.9 illustre un exemple de requête de type CO+S, ces requêtes possèdent le même contenu que les requêtes CO, mais on rajoute des conditions structurelles.
Pour les requêtes de type CAS ou CO+S, un autre champ nommé *castitle* a été rajouté, indiquant le besoin en information sur la structure.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="206"  query_type="CO+S" ct_no="13" >
<InitialTopicStatement>What kind of new problems raises the
    miniaturization
of microprocessors ?</InitialTopicStatement>
<title>problems physical limits miniaturization
   microprocessor</title>
<description>What are the problems and physical limits that
    are encountered by the microprocessor
 miniaturization process ?</description>
<narrative>I'm writing a report about future of
   microprocessors, and I wish to understand why
the process of reduction of their size is soon expected to
   reach its limits.</narrative>
</inex_topic>
```

▶ Figure 2.9: Exemple de requête de type CO+S issue de la compagne d'évaluation INEX'2005

2.9.1.3 Les jugements de pertinence

L'évaluation de la pertinence des SRIS passe par une phase de validation des documents renvoyés par les SRI. Chaque élément du document ou le document en totalité est jugé

à la main (par les participants) pour chaque requête. Cette phase permet d'obtenir des jugements de pertinence. Une échelle à deux dimensions a été proposée lors de la compagne d'évaluation INEX 2004 : la première dimension mesure l'exhaustivité et la seconde dimension mesure la spécificité d'un élément pour une requête donnée [76]. Ces deux dimensions sont multivaluées, ce qui permet une grande finesse dans les jugements.

Exhaustivité l'exhaustivité ne tient compte que de la présence ou de l'absence de l'information recherchée dans un élément, même si cette information n'apparaît que dans une petite partie de l'élément. Par exemple l'élément représentant le document entier sera considéré comme fortement exhaustif même si un seul paragraphe dans tout le document est très pertinent pour la requête et que le reste du document ne l'est pas. On distingue pour ce critère d'évaluation quatre niveaux d'exhaustivité :

- Non exhaustif (0) : l'élément ne traite pas du sujet de la requête ;
- Faiblement exhaustif (1) : l'élément traite marginalement le sujet de la requête ;
- Moyennement exhaustif (2) : le sujet de la requête est en grande partie traité dans l'élément ;
- Fortement exhaustif (3) : le sujet de la requête est traité exhaustivement dans l'élément.

Spécificité la spécificité est totalement liée à l'évaluation de documents structurés. Cette mesure s'intéresse au degré avec lequel l'élément renvoyé par le système traite de toute l'information recherchée. On distingue pour ce critère d'évaluation quatre niveaux de spécificité :

- Non spécifique (0) : l'élément renvoyé ne contient pas de passages pertinents ;
- Faiblement spécifique (1) : une petite partie seulement de l'information contenue dans l'élément est de l'information pertinente ;
- Moyennement spécifique (3) : la plus grande partie contenue dans l'élément est de l'information pertinente ;
- Fortement spécifique (4) : l'élément renvoyé ne contient que de l'information pertinente.

Ces jugements ne sont pas tout à fait orthogonaux, car lorsqu'un élément n'est pas exhaustif, il n'est pas possible de définir sa spécificité et inversement. Il n'y a donc que

10 valeurs possibles et non 16. La spécificité est depuis 2005 considérée comme une dimension continue représentée par une valeur réelle comprise entre 0 et 1 [58] [62].

2.9.2 Les mesures d'évaluation

Les mesures qui ont été proposées par la compagne d'évaluation INEX étendent les mesures traditionnelles utilisées dans la RI. Le but est de prendre en considération les besoins supplémentaires apportées par la structure. Plusieurs mesures ont étés proposées pour évaluer les différents critères d'un système de recherche.

2.9.2.1 Les métriques proposées dans INEX'2004

Plusieurs mesures d'évaluation dans INEX'2004 ont été proposées. Deux fonctions principales permettent de mesurer la performance du SRI, ces fonctions s'appuient sur deux critères : l'exhaustivité (e) et la spécificité (s). Ces deux dimensions sont agrégées en une seule valeur pour mesurer la performance du système, ces fonctions sont définies comme suit :

- une quantification *stricte* pour évaluer si une approche donnée peut trouver les composants qui sont fortement spécifiques suivant une valeur de s, et fortement exhaustifs suivant une valeur de e :

$$f_{strict}(s,e) = \begin{cases} 1 & si\ (e,s) = (3,3) \\ 0 & si\ non \end{cases} \tag{2.6}$$

- une quantification *généralisée* pour évaluer le système selon son degré se pertinence :

$$f_{sog}(s,e) = \begin{cases} 1 & si\ (e,s) = (3,3) \\ 0.9 & si\ (e,s) = (2,3) \\ 0.75 & si\ (e,s) \in \{(1,3),(3,2)\} \\ 0.5 & si\ (e,s) = (2,2) \\ 0.25 & si\ (e,s) \in \{(1,2),(3,1)\} \\ 0.1 & si\ (e,s) \in \{(2,1),(1,1)\} \\ 0 & si\ non \end{cases} \tag{2.7}$$

En 2004, d'autres fonctions d'agrégation ont été introduites. Ils accordent une préférence à la notion d'exhaustivité, en attribuant des scores élevés à des éléments exhaustifs mais pas forcément spécifiques. On parle de fonctions orientées spécificité (les équations 2.8 et 2.9) et de fonctions orientées exhaustivité (les équations 2.10 et 2.11) :

- Les fonctions orientées exhaustivité ne considèrent que les éléments ayant le plus haut degré d'exhaustivité :

$$f_{e3\_s321}(s,e) = \begin{cases} 1 & si\ e = 3\ et\ s \in \{1,2,3\} \\ 0 & si\ non \end{cases} \tag{2.8}$$

$$f_{e3\_s32}(s,e) = \begin{cases} 1 & si\ e = 3\ et\ s \in \{2,3\} \\ 0 & si\ non \end{cases} \tag{2.9}$$

- Les fonctions orientées spécificité ne considèrent que les éléments ayant le plus haut degré de spécificité :

$$f_{s3\_e321}(s,e) = \begin{cases} 1 & if\ s = 3\ et\ e \in \{1,2,3\} \\ 0 & si\ non \end{cases} \tag{2.10}$$

$$f_{s3_e32}(s,e) = \begin{cases} 1 & si\ s = 3\ et\ e \in \{2,3\} \\ 0 & si\ non \end{cases} \tag{2.11}$$

Dans INEX'2004, on ne considère que les 1500 premiers nœuds. L'étape d'évaluation se pose sur les deux mesures populaires de la RI : le rappel et la précision. Un autre critère est considéré dans les jugements dans la recherche des documents XML, c'est le pourcentage d'imbrication ou chevauchement (*overlap*). Cette mesure est supplémentaire, qui favorise 2 nœuds non imbriqués à 2 nœuds pertinents imbriqués. Cette mesure n'est pas prise en considération dans INEX'2002 et INEX'2003 [37] [42].

2.9.2.2 Les métriques proposées dans INEX'2005

Un élément peut être jugé comme :

- 0 : non exhaustif

- 1 : peu exhaustif (voire moyennement exhaustif)
- 2 : exhaustif (voire trop exhaustif)

La spécificité est considérée depuis 2005 comme une dimension continue représentée par une valeur réelle comprise entre 0 et 1. Deux fonctions de quantification principales ont été proposées dans INEX'2005 : les quantifications *stricte* et *généralisée*, ces deux fonctions combinent les deux mesures d'exhaustivité et de spécificité pour juger la pertinence d'un nœud. Ces deux fonctions sont définies comme suit :

$$f_{strict}(s,e) = \begin{cases} 1 & si\,(e,s) = (2,1) \\ 0 & si\ non \end{cases} \tag{2.12}$$

$$f_{gen}(s,e) = e \times s \tag{2.13}$$

Une autre quantification a été ajoutée en 2005 : la quantification *généralisée étendue* (*generalizedLifted*), elle a pour rôle de prendre en considération les nœuds de faible longueur. Cette quantification est définie par l'équation suivante :

$$f_{genLifted}(s,e) = (e+1) \times s$$

La mesure effort-precision/gain-recall (ep/gr) La mesure **ep/gr** est une mesure inspirée de la mesure MAep communément utilisée en RI. La mesure **MAep** (Mean Average Effort Precision) est la moyenne des valeurs MAep d'effort-précision pour chaque rang auquel un nouvel élément pertinent est renvoyé par le système. Cette mesure est considérée comme la mesure la plus importante dans le cadre d'INEX'2005.

Gain cumulé étendu (XCG) La métrique de **xCG** est une famille des métriques des gains cumulés, qui sont une extension du gain cumulé (**CG**) basés sur la métrique de [53]. Cette mesure prend en considération la dépendance des éléments XML (overlap décrit dans la section 2.9.2.1). La mesure du gain cumulé est illustrée par l'équation suivante :

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

où $xG(j)$ est le score obtenu pour l'élément classé à la position j par le système.

La métrique XCG inclut les mesures de gain cumulé étendu normalisé ($nxCG$) données par :

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

où $xCI(i)$ est le gain cumulé idéal.

2.10 Conclusion

Nous avons présenté dans ce chapitre un aperçu sur la RI dans des documents structurés et en particulier sur les documents XML. Les documents XML permettent aux utilisateurs de formuler des requêtes non seulement sur leur contenu, mais aussi d'utiliser leur structure. Les documents semi-structurés, ont réacualisé la problématique de la RI classique. En effet, les SRIS doivent traiter l'information avec une granularité plus fine. Le but des SRIS est alors d'identifier des parties des documents les plus pertinentes à une requête donnée. Ce n'est cependant pas la seule problématique, car la notion de structure doit être prise en compte comme un besoin réel de l'utilisateur tout comme les termes de sa requête.
Nous avons ainsi présenté les principales approches d'indexation et d'appariement développées en RIS. Nous avons également donné un aperçu sur les nouveaux concepts d'évaluation des SRIS. Nous constatons qu'avec la structure la RI dans ses documents peut être plus spécifique et précise. Nous proposons dans le chapitre suivant une approche statistique de RIS basée sur la comparaison d'arbres.

Partie II

Un Modèle de recherche d'information structurée

Chapitre 3

XIVIR : un modèle de recherche structurée basé sur la comparaison d'arbres étendus

3.1 Introduction

Nous avons présenté au cours du deuxième chapitre les différentes problématiques posées dans le cadre de la RIS. La principale conclusion que nous avons pu tirer de cette étude est que le contenu de la requête (les mots-clés) est le principal élément décisif de la sélection des éléments XML à présenter, la structure n'est traitée que d'une façon marginale.

Nous admettons que la structure de la requête est aussi importante que le contenu : son existence est bien une raison de la traiter. Par ailleurs, les conditions structurelles sont souvent accablantes, il existe en général si peu d'éléments XML qui répondent entièrement au besoin en information structurelle de l'utilisateur qu'il devient ardu de rejeter ceux qui en répondent partiellement. Il est indispensable de pouvoir les identifier, quantifier leurs scores orientés structure et les combiner à ceux orientés contenu.

Dans ce chapitre nous proposons un modèle de recherche dans les documents XML (XIVIR : ***X****ML* ***I****nformation retrieval based on* ***VIR****tual links*), permettant une exploitation efficace et flexible de la structure.

Nous proposons dans ce chapitre une approche statistique permettant de combiner à la fois la RI sur la structure et le contenu d'un document XML.

Nous commençons tout d'abord par présenter quelques algorithmes de comparaison d'arbres. Ensuite nous présentons notre modèle pour la RIS basée sur la comparaison d'arbres, en utilisant le principe de relaxation de requêtes. Nous proposons par la suite une méthode d'indexation de la structure. Enfin nous détaillons les techniques que nous avons adoptées pour l'estimation des scores du contenu ainsi que leur combinaison avec les scores de la structure.

3.2 Motivations

L'appariement document-requête consiste à retrouver tous les fragments de l'arbre relatif à la requête ayant un correspondant dans l'arbre relatif au document. La similarité en termes de structure doit être une mesure graduée. Les langages d'interrogation XML classiques sont basés sur la comparaison exacte, ils ne permettent par conséquent que de fournir de résultats binaires, or en RI classique ou structurée, on tente d'ordonner les granules documentaires selon leur pertinence potentielle vis-à-vis de la requête.
L'appariement document-requête doit alors être réalisé d'une façon telle que les granules documentaires dont la structure présente de légères différences avec la structure de la requête reçoivent également un score. Il peut également être vu comme l'inverse de l'effort nécessaire pour la construction incrémentale d'un arbre à partir d'un autre. La comparaison d'arbres a fait l'objet de plusieurs travaux. Ils sont tous basés sur les opérations de mise à jour les plus basiques dont chacune un coût. Le coût de mise à jour est le coût cumulé de toutes les opérations basiques nécessaires. On part de la représentation d'un arbre et on le transforme en un autre avec le minimum de coût possible. Le coût de construction d'un arbre document à partir d'un arbre requête (ou le contraire) peut alors être vu comme l'inverse du score du document par rapport à la requête. En particulier, si le coût est nul, ceci signifie qu'aucune transformation n'est nécessaire pour construire un arbre à partir de l'autre et les arbres sont alors identiques, le score sera dans ce cas précis très élevé, car le coût de construction est très faible.
Pour des raisons de mise en situation expérimentale, les algorithmes de comparaison d'arbres de la littérature ne sont pas toujours adaptés à la RIS. Nous proposons un modèle de comparaison dual. En effet, contrairement aux algorithmes de comparaison d'arbres classiques, qui procèdent à la recherche flexible dans les représentations arborescentes exactes, nous proposons un algorithme de recherche exacte sur des représentations flexibles. Les techniques de stockage et de comparaison sont différentes mais les résultats sont similaires.

3.3 Comparaison d'arbres : travaux antérieurs

Plusieurs techniques ont été mises en place pour la comparaison d'arbres. Ces techniques se basent sur la construction de plans de reconstitution d'un arbre à partir d'un autre, en effectuant des opérations de base sur les nœuds (suppression, ajout, permutation...) et d'estimer le coût de reconstitution.

3.3.1 Algorithme de Selkow

L'algorithme de selkow [115] est basé sur les opérations élémentaires traditionnelles : insérer nœuds, supprimer nœuds et modifier nœuds, il ne permet la suppression et l'insertion que sur les feuilles de l'arbre de départ. De ce fait, un nœud interne ne peut être supprimé ou inséré que si tout son sous arbre subordonné a été supprimé afin qu'il devienne une feuille de l'arbre. Pour calculer la distance entre deux arbres donnés A et B, l'algorithme commence par calculer le coût de transformation de la racine de A en la racine de B auquel il ajoute le coût de transformer par le même algorithme les sous arbres $A_1, A_2 \dots A_n$ de A en les sous arbres $B_1, B_2 \dots B_n$ de B. L'algorithme est donc invoqué récursivement jusqu'à arriver aux feuilles des arbres. La complexité de cet algorithme est de l'ordre de $O(n.m.d)$ où n et m sont les nombres de nœuds respectifs de A et B et d est la profondeur la plus élevée parmi celles des deux arbres.

3.3.2 Algorithme de Tai

Tai [122] utilise une approche de programmation dynamique et pas récursive basée sur les opérations : changer nœud par un autre, insérer nœud et supprimer nœud. Cet algorithme parcourt l'arbre en pré-ordre. Le premier nœud traité est la racine puis, récursivement, tous les sous arbres sont traités en pré-ordre de gauche à droite jusqu'à atteindre les feuilles. Durant ce parcours, chaque nœud est énuméré : le numéro 1 est attribué à la racine et le nœud i est celui qui suit le nœud $i-1$. Pour comparer les arbres, Tai utilise des associations entre les nœuds. La paire de nœuds dans une association peut signifier une opération de changement du premier nœud par le deuxième ou une simple connexion si les nœuds ont la même étiquette. Si pour un nœud n du premier arbre il n'existe aucune association avec un nœud du deuxième arbre, ceci est équivalent au fait que n va être supprimé. De même si pour un nœud m du deuxième arbre il n'existe aucune association, ceci est équivalent au fait que m va être inséré. L'important dans l'algorithme de Tai est qu'il faut prendre en considération la structure (la hiérarchie) de l'arbre. En effet, chaque nœud de l'arbre est en relation avec les autres nœuds du même arbre.

Les résultats de cet algorithme au niveau de la liste des opérations et du coût sont exacts. La complexité de cet algorithme est de l'ordre de $O(n.m.d^2.d'^2)$ où n et m sont respectivement le nombre de nœuds du premier et du deuxième arbre et d et d' leurs profondeurs respectives.

3.3.3 Algorithme de Zhang et Shasha

Cet algorithme est similaire à celui de Tai au niveau des définitions, des associations et des opérations élémentaires [116]. La différence se situe au niveau du parcours et de l'énumération des nœuds. Le calcul des différents coûts est basé sur l'utilisation de nœuds spécifiques dans les arbres appelés *keyroots*, c'est l'ensemble des nœuds qui ont un frère gauche en plus de la racine. Cette approche n'a pas besoin de vérifier les associations non autorisées puisque le coût de la transformation d'un nœud n'est calculé qu'après avoir déterminé les différents coûts des opérations sur tous ses descendants. La complexité de cet algorithme est de l'ordre de $O(n.m.d^2.d')$ où n et m sont respectivement le nombre de nœuds du premier et du deuxième arbre, et d et d' leurs profondeurs respectives.

3.3.4 Discussion

Les algorithmes de comparaison d'arbres représentent une excellente illustration de la recherche d'information structurée. Ils fournissent des moyens de comparaison basés sur des opérations élémentaires simples (ajouter, supprimer, permuter, renommer un nœud...). Le coût de comparaison est favorable à la recherche structurée, en effet, ces algorithmes calculent ce coût dans une échelle hautement graduée, ce qui permet de classer les éléments XML en fonction de celui-ci (le coût de construction pourra être vu comme l'inverse de la valeur de similarité entre un document et une requête). L'avantage majeur de ces approches et la grande flexibilité assurée au moment de la comparaison, elles permettent de comparer un arbre à un autre dès que la possibilité se présente, et avec le minimum de coût possible. Cependant, ces algorithmes ont une complexité très élevée. Comme leur mise en marche est réalisée au moment de la recherche, le passage à l'échelle les rend inexploitables.

Popovici [78] a proposé une alternative aux algorithmes de comparaison d'arbre pour la RIS qui part d'une simplification qui consiste à remplacer un arbre XML par un ensemble de chemins. L'idée consiste à mesurer la similarité entre deux arborescences représentant respectivement le document et la requête en se basant sur la distance de Levenshtein [66] utilisée pour mesurer la distance entre deux chaînes de caractères. La distance de Levenshtein se base sur les opérations suivantes : insertion, suppression et

substitution de caractère, il s'agit donc de donner un coût en se basant sur les opération de transformation des arbres (insertion nœud, suppression nœud et substitution nœud) :

- La substitution : un nœud n^q d'un chemin p_q de la requête peut être remplacé par un nœud n d'un chemin p_d du document avec un coût minimal $C(n_q, n)$.
- La suppression : un nœud n peut être éliminé d'un chemin p_d avec un coût $C(n, \epsilon)$.
- L'insertion : un nœud n sera insérer dans un chemin p_d avec un coût $C(\epsilon, n)$.

Etant données deux chaines de caractères s_1 et s_2, la distance d'édition est le nombre minimum d'opérations nécessaires pour transformer s_1 en s_2. Les opérations sont (i) insérer un caractère; (ii) supprimer un caractère et (iii) remplacer un caractère par un autre; La distance de Levenshtein est l'une des plus utilisées dans ce domaine. La notion de distance d'édition peut être généralisée par l'attribution de poids à chaque opération, par exemple nous pouvons attribuer un poids plus élévé au remplacement du caratère s par p que par le caractère a (Parcequ'il est plus proche de a que de p dans un clavier).

La complexité du calcul de la distance d'édition entre s_1 et s_2 est $O(|s_1| \times |s_2|)$ où $|s|$ est la longueur de la chaine de caractères s L'idée est d'utiliser l'algorithme de programmation dynamique 3, où les chaines s_1 et s_2 sont données sous forme de tableaux de caractères.

Algorithme 3 Distance de Levenstein entre s_1 et s_2

$M[0,0] = 0$
pour i de 1 à $|s_1|$ **faire**
  $M[i,0] = i$
fin pour
pour j de 1 à $|s_2|$ **faire**
  $M[0,j] = j$
fin pour
pour i de 1 à $|s_1|$ **faire**
  pour j de 1 à $|s_2|$ **faire**
    si $s_1[i] = s_2[j]$ **alors**
      $M[i,j] = min\{M[i-1,j-1], M[i,j-1]+1, M[i-1,j]+1\}$
    si non
      $M[i,j] = min\{M[i-1,j-1]+1, M[i,j-1]+1, M[i-1,j]+1\}$
    fin si
  fin pour
fin pour

L'algorithme remplit les entrées de la matrice d'entiers M où les deux dimensions sont égales aux longueurs respectives de s_1 et s_2. $M[i,j]$ contient, après l'exécution de l'algorithme, la distance d'édition des i premiers caractères de s_1 et des j premiers caractères de s_2. $M[i,j]$ est le minimum entre $M[i-1,j-1]$ qui correspond à remplacer un substituer un caractère de s_1 par un autre, $M[i-1,j]+1$ qui correspond à ajouter un caractère dans s_1 et $M[i,j-1]+1$ qui correspond à ajouter un caractère dans s_2. La figure 3.3.4 illustre un exemple de l'exécution de l'algorithme 3 pour les chaînes de caractères $fast$ et $cats$.

		f	a	s	t
	$\begin{pmatrix} . & . \\ . & 0 \end{pmatrix}$	$\begin{pmatrix} . & . \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} . & . \\ 2 & 2 \end{pmatrix}$	$\begin{pmatrix} . & . \\ 3 & 3 \end{pmatrix}$	$\begin{pmatrix} . & . \\ 4 & 4 \end{pmatrix}$
c	$\begin{pmatrix} . & 1 \\ . & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 \\ 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 3 & 4 \\ 3 & 3 \end{pmatrix}$	$\begin{pmatrix} 4 & 5 \\ 5 & 5 \end{pmatrix}$
a	$\begin{pmatrix} . & 2 \\ . & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 2 \\ 3 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 3 \\ 3 & 1 \end{pmatrix}$	$\begin{pmatrix} 3 & 4 \\ 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 4 & 5 \\ 3 & 3 \end{pmatrix}$
t	$\begin{pmatrix} . & 3 \\ . & 3 \end{pmatrix}$	$\begin{pmatrix} 3 & 3 \\ 4 & 3 \end{pmatrix}$	$\begin{pmatrix} 3 & 2 \\ 4 & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 4 \\ 3 & 2 \end{pmatrix}$
s	$\begin{pmatrix} . & 4 \\ . & 4 \end{pmatrix}$	$\begin{pmatrix} 4 & 4 \\ 5 & 4 \end{pmatrix}$	$\begin{pmatrix} 4 & 3 \\ 5 & 3 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 \\ 4 & 2 \end{pmatrix}$	$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}$

▶ Figure 3.1: Distance de Levenshtein calculée dynamiquement entre *fast* et *cats*

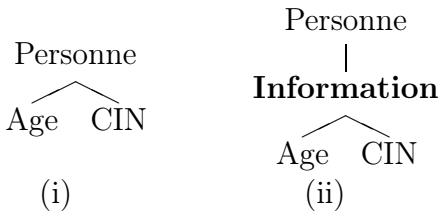
Une entrée est typiquement une matrice $E2 \times 2$ où $E[2,2]$ correspond au minimum des autres valeurs $E[1,1]$, $E[1,2]$ et $E[2,1]$. C'est la distance de Levenstein calculée dynamiquement pour les i premiers caractères de $s1$ et les j premiers caractères de s_2. Les autres entrées sont $M[i-1,j-1]+0$ ou 1 selon que $s_1[i] = s_2[j]$ ou non, $M[i-1,j]+1$ et $M[i,j-1]+1$.

Chaque coût de transformation varie entre 0 (correspondance parfaite) et 1 (aucune correspondance). Le degré de similarité entre les deux chemins de la requête et du document est inversement proportionnel à la somme de ses coûts de transformation. Plus le coût diminue, plus les deux chemins se ressemblent. Le coût tend vers 0 si les des chemins d'un document XML sont très similaires à un chemin de la requête et tend vers 1 dans le cas contraire. Des heuristiques complexes sont utilisées pour estimer les coûts de transformation en tenant compte des relations sémantiques entre les différents identificateurs des éléments XML.

3.4 Relaxation de requêtes

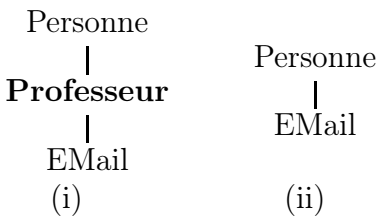
La relaxation d'une requête [133] [134] consiste à tolérer des résultats qui présentent des différences légères en terme de structure, et de les retourner avec un score qui illustre ces différences. Les résultats d'une requête seront ceux de la requête exacte auxquels on rajoute des fragments candidats qui lui sont isomorphes au moyen de quatre relaxations possibles.

Insertion d'un nœud interne : L'ajout d'un nœud interne entre deux nœuds transforme la relation entre eux de (père-fils) en (ancêtre-descendant). La figure 3.2 illustre la transformation de la relation entre le nœud **Personne** et ses deux enfants **Age** et **CIN** en introduisant le nœud **Information**. Malgré cette différence, le deuxième arbre contient tous les champs du premier et respecte ses niveaux de hiérarchie. En supposant que le premier arbre est une requête, le deuxième, bien que partiellement différent au niveau de la structure, peut être retenu comme réponse.



▶ Figure 3.2: Insertion d'un nœud interne

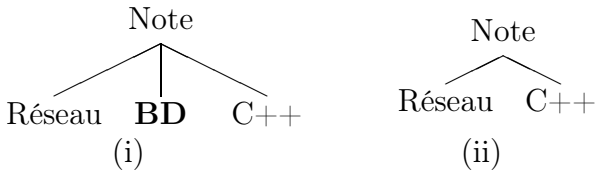
Suppression d'un nœud interne : La suppression d'un nœud interne entre deux nœuds transforme la relation entre eux de (ancêtre-descendant) en (père-fils). Dans la figure 3.3, la suppression du nœud **Professeur** fait que le **EMail** appartient pour une personne inconnue. Il est vrai que l'information est manquante, mais la réponse peut être retenue avec moins d'importance.



▶ Figure 3.3: Suppression d'un nœud interne

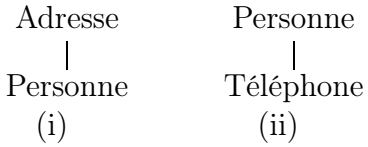
Suppression d'un nœud feuille : Prenons l'exemple d'une requête qui cherche les différentes notes d'un étudiant (figure 3.4) et qui trouve une réponse qui ne contient pas la note de **bases de données**, il est vrai qu'on ne pourra pas calculer

la moyenne générale de cet étudiant mais on pourrait utiliser les autres notes de cet étudiant pour calculer son rang et celui de ses camarades dans les différentes matières autres que les bases de données. Nous pouvons alors tolérer les réponses pour lesquelles il manque une ou plusieurs feuilles tout en indiquant à l'utilisateur, dans le classement des résultats, que cette réponse est incomplète.



▶ Figure 3.4: Suppression d'un nœud feuille

Insertion d'un nœud feuille : L'ajout d'un nœud terminal correspond à une information ou une valeur supplémentaire dans la réponse à une requête. Dans la figure 3.5, l'ajout du nœud feuille **Téléphone** fait que l'information associée est supplémentaire pour une personne.



▶ Figure 3.5: Insertion d'un nœud feuille

Nous pouvons tolérer la réponse à une requête pour laquelle un ou plusieurs nœuds feuilles peuvent être ajoutés.
Le principe de relaxation de requêtes semble être un concept pertinemment applicable à la RIS. En effet, l'appariement document-requête pourra être vu comme la relaxation de la requête suivant les 4 opérations de mise à jour précédemment mentionnées qui, partant de la requête, tente de rejoindre la structure du document (ou une partie de celui-ci) ou inversement. Cependant, le passage à l'échelle rend la relaxation de la requête une tâche très complexe et à forte composante combinatoire.
Nous nous inspirons tout de même de ce principe pour l'appariement document-requête. Notre modèle se base la migration des liens XML de la relation père-fils qualitatives aux relations ancêtre-descendant valuées et leur utilisation en RI vague.

Cette transformation peut être vue comme une extension d'un arbre XML en une représentation plus flexible [7]. Un résultat est retenu si il est à la fois présent dans l'arbre tendu du document et celui de la requête.

Notre approche consiste alors à rechercher tous les sous-arbres de l'arbre document qui ont un équivalent en terme de structure dans celui de la requête [9]. Un algorithme

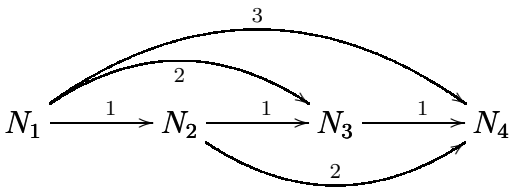
de recherche exacte semble faire l'affaire, qui se ramène en effet à une recherche flexible sur les représentations exactes du document et de la requête. Les deux approches sont duales, elles offrent des résultats similaires mais notre approche, par opposition aux algorithmes de Selkow, de Tai et de Zhang et Shasha, n'est coûteuse que pendant la transformation de l'arbre original.

3.5 Le modèle XIVIR

Nous nous sommes basés sur les modélisations arborescentes des documents XML. Cette modélisation permet de façon naturelle de traiter la structure et le contenu d'un document XML d'une manière indépendante. L'objectif principal de notre méthode est de combiner l'information portée par la structure et celle portée par le contenu pour la RIS. Nous nous focalisons sur le traitement de la structure auquel nous donnons une importance capitale, tout comme le contenu. Nous décrivons dans cette section le modèle XIVIR. Nous allons tout d'abord présenter l'étape d'indexation de la structure. Nous allons ensuite décrire l'utilisation de l'index résultant dans l'interrogation orientée structure. Nous allons par la suite présenter le traitement du contenu, et enfin la combinaison des scores résultants de l'interrogation orientée contenu et celle orientée structure.

3.6 Indexation de la structure d'un arbre XML

La structure peut influencer la pertinence d'un fragment XML. Elle doit alors être indexée à l'instar du contenu des documents en RI classique. Le principe de notre méthode est purement statistique, il consiste à ajouter des relations entre un nœud et ses descendants. Ces relations sont exprimées à l'aide d'arcs virtuels au niveau de la structure d'un document XML comme illustre la figure 3.6.



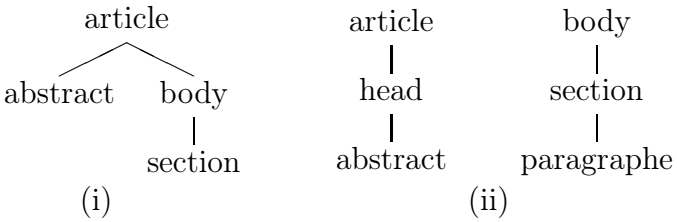
▶ Figure 3.6: Ajout des arcs virtuels

Formellement, un arbre XML est un ensemble de chemins entre deux nœuds $A \longrightarrow B$ où le nœud A est le parent du nœud B. La racine de l'arbre XML est la seule qui

n'a aucun parent. Ainsi un arbre XML T devrait avoir la propriété suivante :

$$\{N, \forall N^{'}, N^{'} \longrightarrow N \notin T\} = \{racine\} \tag{3.1}$$

La figure 3.7 montre que la représentation (i) vérifie la propriété d'un arbre XML. En effet, $T = \{article \longrightarrow abstract, article \longrightarrow body, body \longrightarrow section\}$ et $\{N, \forall N', N' \longrightarrow N \notin T\} = \{article\}$, tandis que la représentation (ii) ne l'est pas puisque $T = \{article \longrightarrow head, head \longrightarrow abstract, body \longrightarrow section, section \longrightarrow paragraphe\}$ et $\{N, \forall N', N' \longrightarrow N \notin T\} = \{article, body\}$. Un fragment peut être défini comme le



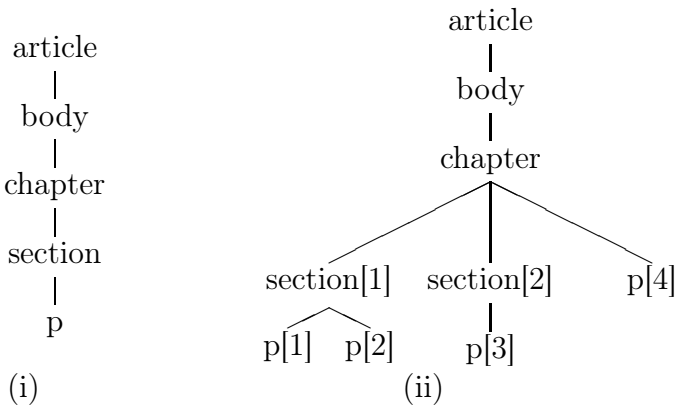
▶ Figure 3.7: Les propriétés d'arbre XML, la représentation (i) vérifie les propriétés tandis que la représentation (ii) ne l'est pas

résultat de jointure multiple des chemins. Par exemple, si un arbre XML contient des chemins $A \longrightarrow B$ et $B \longrightarrow C$, nous définissons un autre chemin $A \longrightarrow B \longrightarrow C$. Un chemin est alors une liste ordonnée de nœuds $N_1 \longrightarrow N_2 \longrightarrow \ldots \longrightarrow N_n$.
Aucun cycle ne doit apparaître dans un chemin de l'arbre, ainsi si $N_1 \longrightarrow N_2 \longrightarrow \ldots \longrightarrow N_n$ est un chemin de l'arbre T, alors tous les $N_.$ doivent être différents. Ceci mène à une autre propriété qui exige que chaque nœud sauf la racine doit possèder un seul parent :

$$\forall N, |\{N' \longrightarrow N \in T\}| = \begin{cases} 0 & si\ N = racine \\ 1 & si\ non \end{cases} \tag{3.2}$$

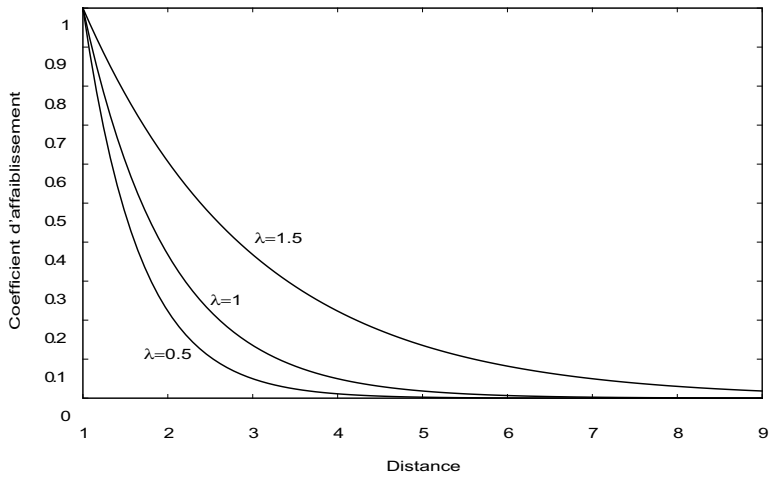
3.7 Appariement de structure

La recherche de fragments ayant la même structure de l'arbre requête peut être effectuée en les recherchant dans l'arbre document. La figure 3.8 montre que le document (ii) contient 3 fragments qui partagent la même structure que la requête (i). Ces fragments sont $(article \longrightarrow body \longrightarrow chapter \longrightarrow section[1] \longrightarrow p[1], article \longrightarrow body \longrightarrow chapter \longrightarrow section[1] \longrightarrow p[2]$ et $article \longrightarrow body \longrightarrow chapter \longrightarrow section[2] \longrightarrow p[3])$. Pour ordonner des fragments selon leurs scores [25] [56], certaines conditions doivent être prises en considération. Notre hypothèse est que les fragments qui remplissent exactement la structure de la requête doivent apparaître en tête de liste. Les fragments des documents ayant de légères différences par rapport à l'arbre requête ne devraient pas être totalement rejetés, par exemple, la figure 3.8 montre que dans le



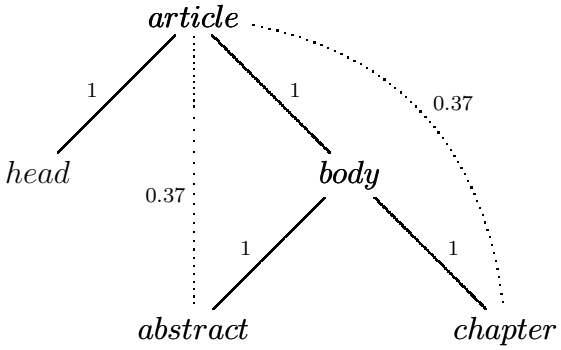
▶ Figure 3.8: Un exemple d'une requête et d'un document, plusieurs des fragments remplissent les conditions de structure

fragment *article* $\longrightarrow$ *body* $\longrightarrow$ *chapter* $\longrightarrow$ $p[4]$, un seul nœud manque (*section*), toutefois le SRIS devrait le retenir.
Le SRIS devrait rechercher les fragments les plus profonds et les plus larges possibles partagés par les deux représentations. Pour ce faire, nous ajoutons à chaque chemin $A \longrightarrow B$ un poids qui reflète l'importance de la relation entre les nœuds A et B. La relation parent-fils directe a un poids égal à 1. Plus A est éloigné de B dans le chemin original, plus le poids est faible. Par exemple, le chemin *chapter* $\longrightarrow$ *section* $\longrightarrow p$ engendre le chemin *chapter* $\longrightarrow p$ auquel on associe un poids légèrement inférieur à 1. Naturellement, le poids dépend de la distance entre les deux nœuds d'un chemin. Nous employons une fonction de poids f définie par $f(A \rightarrow B) = exp(\lambda \times (1 - d(A, B)))$ où $d(A, B)$ est la distance qui sépare le nœud A du nœud B, la figure 3.9 illustre l'effet de λ sur les poids des liens virtuels. Nous dénotons ce chemin par $A \xrightarrow{w} B$ où $w = exp(\lambda \times (1 - d(A, B)))$ est le poids du chemin $A \longrightarrow B$. Pour permettre la



▶ Figure 3.9: Le rôle du paramètre λ sur la convexité de la fonction $g(x) = \exp(\lambda \times (1 - x))$

recherche flexible de structures pertinentes, nous commençons par étendre les chemins des arbres requête et document, puis nous recherchons le fragment le plus profond et large partagé par les deux ensembles de chemins pondérés qui remplissent les propriétés d'arbre décrites par les équations 3.1 et 3.2, c'est-à-dire, en identifiant tous les ensembles $E \subset Q \cap D$[1] qui remplissent les propriétés d'arbre.
Q (resp. D) est un sous ensemble de chemins de l'arbre requête (resp. document). La figure 3.11 montre que l'ensemble $\{article \xrightarrow{1} body, body \xrightarrow{1} chapter$ et $chapter \xrightarrow{0.37} p\}$ extrait de l'arbre de la requête et l'ensemble $\{article \xrightarrow{1} body, body \xrightarrow{1} chapter$ et $chapter \xrightarrow{1} p[4]\}$ extrait de l'arbre document sont quasi-semblables. Si nous ignorons le poids de chaque chemin, ces ensembles sont identiques et ils remplissent les propriétés de l'arbre (équations 3.1 et 3.2). Nous supposons que cet ensemble de chemins est potentiellement un sous-arbre approprié de structure, son score dépend des poids inscrits sur chaque arc : plus ces poids sont élevés, plus les chemins sont réels et par conséquent plus le score est élevé. Etant donnés une requête illustrée par la figure 3.10 et un document XML illustré par la figure 3.11, les réponses avec relaxation à cette requête peuvent exister dans un ou plusieurs sous arbres du document [136]. De ce fait, la première opération que nous devons faire est de comparer l'arbre de la requête à celui du document et d'en extraire les sous arbres similaires. L'algorithme que nous



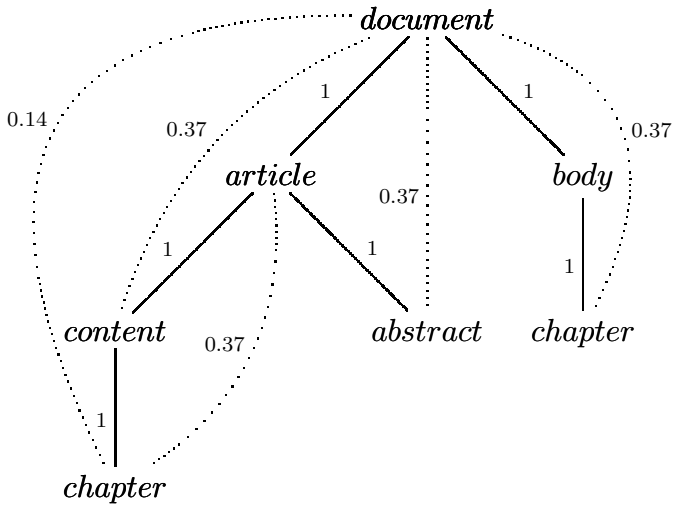
▶ Figure 3.10: L'arbre requête étendu

proposons pour la comparaison d'arbres permet de localiser les sous arbres similaires à l'arbre représentant la requête. Cet algorithme parcourt l'arbre en pré-ordre. Le premier nœud traité est la racine puis, récursivement, tous les sous arbres sont traités en pré-ordre. Pour comparer ces arbres nous avons utilisé une hypothèse sur leurs structures arborescentes qui réside dans l'association entre les nœuds (au moins deux nœuds pour faire la comparaison : dans notre approche nous prenons l'association père-fils) : si un nœud du premier arbre est similaire à un nœud du deuxième arbre (comparaison selon les noms de balise), il suffit de trouver une intersection entre les fils du nœud

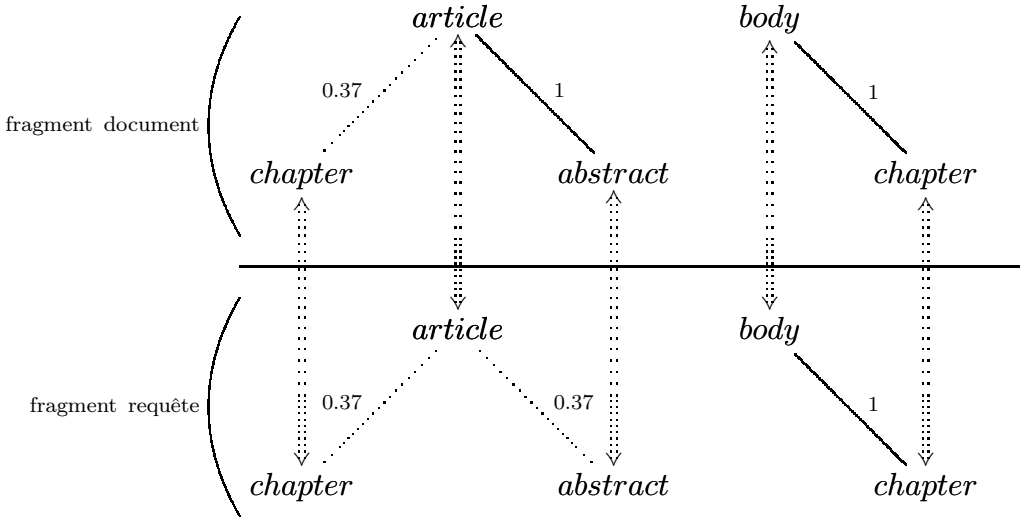
[1]Nous devons ignorer les poids du chemin pour calculer $Q \cap D$.

Algorithme 4 Recherche par la structure

E_q {requête étendue}
E_d {document étendu}
$F \leftarrow \emptyset$ {l'ensemble des fragments retournés est vide}
pour chaque $n_q \xrightarrow{w_q} n'_q \in E_q \{n'_q$ est un ancêtre de $n_q\}$ **faire**
  pour chaque $n_d \xrightarrow{w_d} n'_d \in E_d, n_q \equiv n_d$ et $n'_q \equiv n'_d \{n'_d$ est un ancêtre de n_d, $\alpha \equiv \beta$ signifie que les nœuds α et β ont le même nom de balise$\}$ **faire**
    si $\exists (f_q, f_d, C, r) \in F, (n_q \xrightarrow{\cdot} ., n_d \xrightarrow{\cdot} .) \in C$ ou $(. \xrightarrow{\cdot} n_q, . \xrightarrow{\cdot} n_d) \in C$ { les fragments f_q et f_d peuvent être enrichis par tous les chemins communs $n'_q \xrightarrow{\cdot} .$ et $n'_d \xrightarrow{\cdot} .$, C est un ensemble de couples où chaque couple $(p, p') \in C$ signifie que le chemin p dans la requête est aligné au chemin p' dans l'arbre document, r est la valeur de pertinence calculée au fur et à mesure de la recherche par la structure$\}$ **alors**
      pour chaque $n'_q \xrightarrow{w'_q} n''_q \in E_q, n'_d \xrightarrow{w'_d} n''_d \in E_d, n''_q \equiv n''_d \{n'_q \xrightarrow{w'_q} n''_q$ et $n'_d \xrightarrow{w'_d} n''_d$ sont employés pour enrichir les fragments f_q et $f_d\}$ **faire**
        $(f^*_q, f^*_d, C^*, r^*) \longleftarrow (f_q, f_d, C, r)$ {nous copions (f_q, f_d, C, r) vers un nouveau fragment (f^*_q, f^*_d, C^*, r^*) et l'enrichissons par des chemins $n'_q \xrightarrow{w'_q} n''_q$ et $n'_d \xrightarrow{w'_d} n''_d\}$
        $f^*_q \longleftarrow f^*_q \cup \{n'_q \xrightarrow{w'_q} n''_q\}$
        $f^*_d \longleftarrow f^*_d \cup \{n'_d \xrightarrow{w'_d} n''_d\}$
        $C^* \longleftarrow C^* \cup \{(n'_q \xrightarrow{w'_q} n''_q, n'_d \xrightarrow{w'_d} n''_d)\}$
        $r^* \longleftarrow r^* + w'_q \times w'_d \{$mise à jour du score$\}$
        $F \longleftarrow F \cup \{(f^*_q, f^*_d, C^*, r^*)\} \{$mise à jour de l'ensemble $F\}$
      fin pour
    si non
      $F \longleftarrow F \cup \{(\{n_q \xrightarrow{w_q} n'_q\}, \{n_d \xrightarrow{w_d} n'_d\}, \{(n_q \xrightarrow{w_q} n'_q,\ n_d \xrightarrow{w_d} n'_d)\}, w_q \times w_d)\}$ {le chemin n'existe pas encore parmi les fragments déjà retournés, il peut être ajouté comme nouveau fragment avec un chemin unique$\}$
    fin si
  fin pour
fin pour



► Figure 3.11: L'arbre document étendu

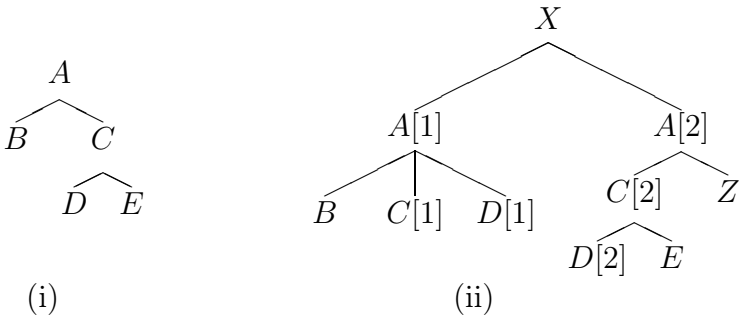


► Figure 3.12: Deux sous-arbres communs entre l'arbre requête et l'arbre document

du premier arbre et ceux du nœud du deuxième arbre. Si cette hypothèse est vérifiée, on retient le nœud du deuxième arbre avec les fils communs, les arcs virtuels facilitent cette opération de comparaison. En effet grâce aux arcs virtuels un seul niveau sépare le père et ses descendants indirects (comme ses enfants). La comparaison se fait d'une manière flexible, la figure 3.12 montre que notre processus de comparaison localise deux fragments communs entre la requête et le document représentés respectivement par les figure 3.10 et 3.11.

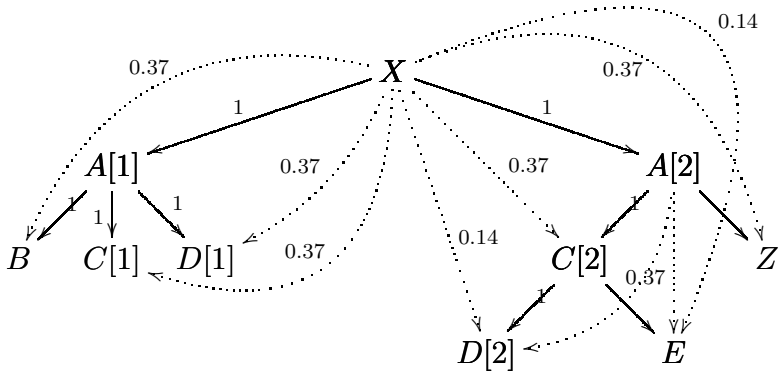
3.7.1 Illustration par l'exemple

La mise en route de l'algorithme 4 nécessite un arbre document et un arbre requête. Nous allons considérer pour cette section l'exemple de la requête et du document illustrés par la figure 3.13. La représentation (i) est celle de la requête et (ii) est celle du document. L'algorithme débute par l'extension de chaque arbre, il en résulte l'arbre

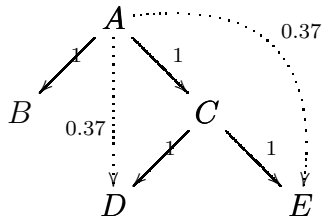


► Figure 3.13: Arbres document et requête

document étendu illustré par la figure 3.14 et l'arbre étendu requête illustré par la figure 3.15. Les chemins pondérés de l'arbre document (éléments de l'ensemble E_d)



► Figure 3.14: Arbre étendu document



► Figure 3.15: Arbre étendu requête

sont : $X \xrightarrow{1} A[1]$, $X \xrightarrow{1} A[2]$, $X \xrightarrow{0.37} B$, $X \xrightarrow{0.37} C[1]$, $X \xrightarrow{0.37} D[1]$, $X \xrightarrow{0.37} C[2]$, $X \xrightarrow{0.37} Z$, $X \xrightarrow{0.14} D[2]$, $X \xrightarrow{0.14} E$, $A[2] \xrightarrow{0.37} D[2]$ et $A[2] \xrightarrow{0.37} E$.

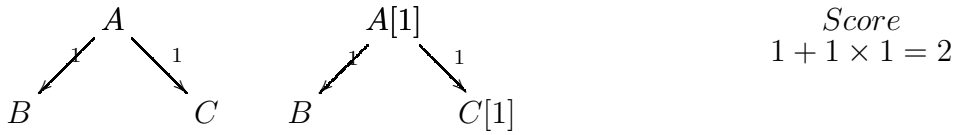
Les chemins pondérés de l'arbre requête (éléments de l'ensemble E_q) sont : $A \xrightarrow{1} B$, $A \xrightarrow{1} C$, $C \xrightarrow{1} D$, $C \xrightarrow{1} E$, $A \xrightarrow{0.37} D$ et $A \xrightarrow{0.37} E$.

La recherche de fragments potentiellement pertinents est réalisée par construction évolutive d'arbres : on part d'une branche de l'arbre requête (et d'une branche de l'arbre document ayant des extrémités de même nom), puis on tente de la développer par enrichissement itératif de sa structure jusqu'à ce que la structure soit celle d'un arbre, l'opération est réalisée tant que possible et tous les cas d'enrichissement possibles doivent être envisagés.
Le point de départ est alors l'arbre requête. L'instruction 4 de l'algorithme 4 montre que la construction d'un arbre cinstruction la sélection d'un chemin pondéré de l'arbre requête (sélection d'un élément E_q).
Le premier chemin sélectionné dans notre exemple est $A \xrightarrow{1} B$, l'instruction 5 indique que pour continuer la construction de l'arbre, il faudra trouver un chemin pondéré dans l'arbre document ayant la même description que $A \xrightarrow{1} B$. Seules les extrémités de l'arc (A et B) sont considérés pour la sélection de branches de l'arbre document candidates au développement de fragments pertinents. Le poids de l'arc est en revanche utilisé pour l'estimation du score.
La seule branche $A \xrightarrow{\cdot} B$ de l'arbre document est $A[1] \xrightarrow{1} B$, elle sera alors utilisée pour enrichir tous les fragments contenant l'extrémité supérieure de la branche (A), c'est-à-dire un fragment contenant une branche telle que le nom de l'une de ses extrémités est A. Cette hypothèse est illustrée par l'instruction 6 de l'algorithme 4. Il est à noter que si l'extrémité suinstruction la branche est celle qui porte le même nom que A, alors le fragment tend à devenir profond, autrement, il tend à devenir large. L'enchaînement de l'algorithme veut que les fragments résultats soient aussi larges que profonds, ceci dépend de la largeur et de la profondeur de la requête et l'existence de fragments de l'arbre document ayant une structure semblable à celle de la requête ou un fragment de la requête.
Or initialement, aucun fragment n'est encore construit, d'où il n'y a rien à enrichir :

ceci est le résultat de l'instruction de contrôle 6 de l'algorithme 4. Ce contrôle est alors utilinstructionrichir des fragments avec la branche courante, mais seulement lorsque ceci est possible, autrement, la branche sera admise comme un nouveau fragment potentiellement pertinent. S'il y a possibilité de l'enrichir, il le sera lors des itérations suivantes. La prise en charge de la branche $A \dot{\longrightarrow} B$ parmi les structures potentiellement pertinentes est illustrée par l'instruction 15 de l'algorithme 4. Cette première construction est illustrée pinstructione 3.16. Le score du fragment est calculé en fonction des poids des arcs dans chacune des branches $A \dot{\longrightarrow} B$ dans le document et dans la requête ($w_q \times w_d = 1 \times 1 = 1$). Le chemin suivant sélectionné dans notre

▶ Figure 3.16: Résultat 1

exemple est $A \xrightarrow{1} C$. Dans l'arbre document, deux branches peuvent être considérées : $A[1] \xrightarrow{1} C[1]$ et $A[2] \xrightarrow{1} C[2]$, la première branche est utilisée pour enrichir l'arbre résultat illustré par la figure 3.16 car il est l'unique fragment qui contient une branche dont une des extrémités est $A[1]$, puisque l'élément est l'extrémité supérieure de la branche, l'enrichissement du fragment entraîne son développement en largeur, la figure 3.17 illustre le résultat de cet enrichissement. La mise à jour du fragment est réalisée par les instructions 8 ... 13 de l'algorithme 4. La mise à jour d'un fragment concerne également instructions score par l'ajout du produit du poids de la branche document par son équivalent dans la requête. Plus les branches sont réelles, plus les poids sont élevés et par conséquent, plus le score est élevé. La branche $A[2] \xrightarrow{1} C[2]$

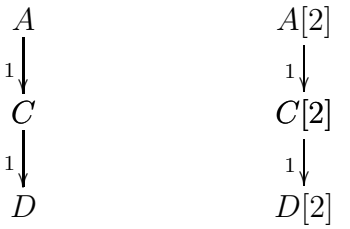


▶ Figure 3.17: Résultat 2

sera traitée différemment, car l'extrémité supérieure $A[2]$ ne figure dans aucun fragment pertinent (l'instruction 6 envoie vers l'instruction 15), elle aura alors le même sort que la preminstruction branche traitéinstructiongorithme 4 qui lui instanciera un fragment parmi l'ensemble des résultats. La figure 3.18 illustre ce fragment. Le déroulement de l'algorithme continue comme suit :

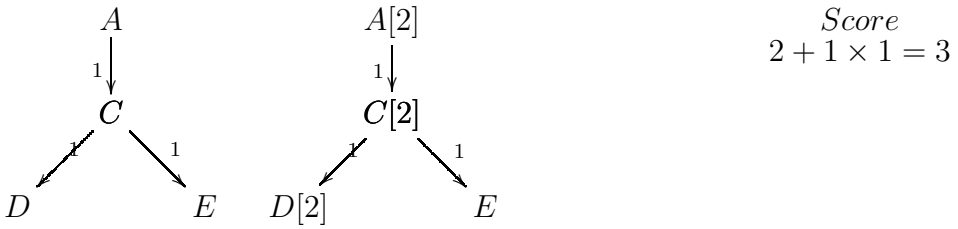
1. $C \xrightarrow{1} D$: le fragment illustré par la figure 3.19 est le résultat de la mise à jour du fragment illustré par la figure 3.18.

▶ Figure 3.18: Résultat 3



▶ Figure 3.19: Résultat 4

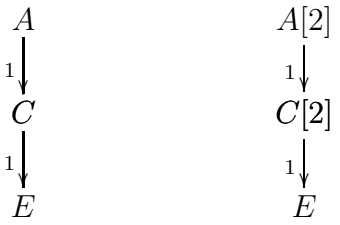
2. $C \xrightarrow{1} E$: le fragment illustré par la figure 3.20 (resp. figure 3.21) est le résultat de la mise à jour du fragment illustré par la figure 3.19 (resp. figure 3.18).



▶ Figure 3.20: Résultat 5

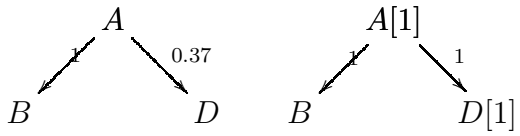
3. $A \xrightarrow{0.37} D$: le fragment illustré par la figure 3.22 (resp. figure 3.23, figure 3.24, figure 3.25) est le résultat de la mise à jour du fragment illustré par la figure 3.16 (resp. figure 3.17, figure 3.18, figure 3.21).

4. $A \xrightarrow{0.37} E$: le fragment illustré par la figure 3.26 (resp. figure 3.27, figure 3.28) est le résultat de la mise à jour du fragment illustré par la figure 3.18 (resp. figure 3.19, figure 3.24). Le parcours des éléments de la requête et celui des documents doit être organisé telle que la restitution de tous les fragments potentiellement pertinents soit garantie. En effet, le parcours de l'arbre en commençant par les feuilles jusqu'à la racine ne fournit aucun résultat. C'est justement le parcours inverse qui doit être considéré. La priorité est réservée à la racine de l'arbre, puis les éléments fils sont traités abstraction faite de leur ordre (l'ordre dont lequel les éléments subordonnés figurent n'a aucune importance, ils jouent tous



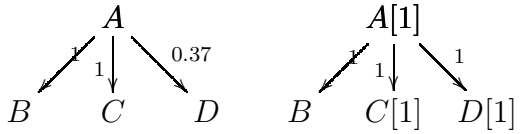
$$Score \\ 1 + 1 \times 1 = 2$$

▶ Figure 3.21: Résultat 6



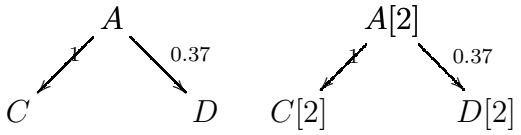
$$Score \\ 1 + 0.37 \times 1 = 1.37$$

▶ Figure 3.22: Résultat 7



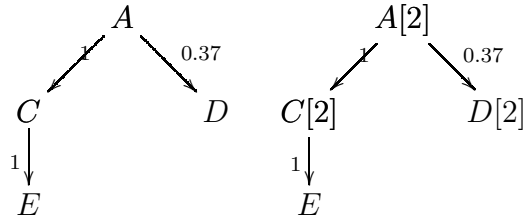
$$Score \\ 2 + 0.37 \times 1 = 2.37$$

▶ Figure 3.23: Résultat 8



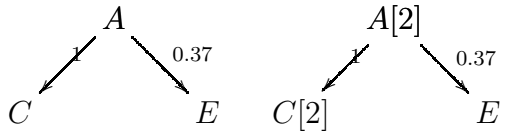
$$Score \\ 1 + 0.37 \times 0.37 = 1.14$$

▶ Figure 3.24: Résultat 9



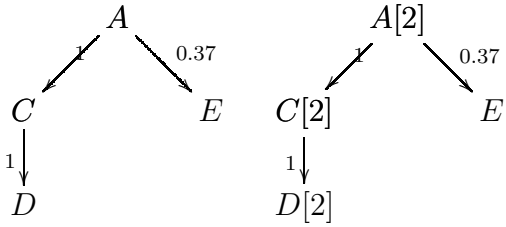
$$Score \\ 1 + 0.37 \times 0.37 = 1.14$$

▶ Figure 3.25: Résultat 10



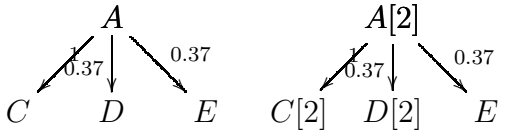
$$Score$$
$$1 + 0.37 \times 0.37 = 1.14$$

▶ Figure 3.26: Résultat 11



$$Score$$
$$2 + 0.37 \times 0.37 = 2.14$$

▶ Figure 3.27: Résultat 12



$$Score$$
$$1.14 + 0.37 \times 0.37 = 2.28$$

▶ Figure 3.28: Résultat 13

des rôles symétriques). Pour cette fin, il suffit de parcourir l'arbre dans un sens tel qu'aucun élément requête (resp. document) ne doit être traité dans l'instruction 4 (resp. 5) de l'algorithme 4 avant d'avoir traité ses ancêtres.
Leinstructionrcours que nous utilisons dans notre prototype est *Racine* → *Gauche* → *Droite*. L'exemple que nous venons de détailler utilise un type de parcours similaire.

3.7.2 Scores de structure

Nous employons le produit cumulatif de chaque poids de chemin selon la requête par son correspondant dans le document :

$$rsv_s(E_q, E_d) = \sum_{A_q \xrightarrow{w_q} B_q \in E_q \equiv A_d \xrightarrow{w_d} B_d \in E_d} w_q \times w_d$$

où rsv_s dénote le score de structure d'un sous-arbre du document qui satisfit les conditions structurelles de la requête. E_q (rep. E_d) est l'ensemble de tous les chemins pondérés de la requête (resp. document) et $A_q \equiv A_d$ signifie que A_q est l'homologue de A_d (par exemple pour la figure 3.13, $chapter \xrightarrow{0.37} p \equiv chapter \xrightarrow{1} p[4]$). Chaque nœud reçoit alors le score du fragment retrouvé auquel il appartient.

En résumé, le problème de récupération de structures pertinentes recherche un ensemble dénoté R_s de tous les sous-ensembles de l'arbre étendu du document qui remplissent les propriétés d'arbre tel qu'il lui correspond un sous-ensemble similaire dans l'arbre étendu de la requête :

$$R_s = \{(E_d, s), E_d \subset D, \exists E_q \subset Q, E_d = E_q \; et \; arbre(E_d)\}$$

où s est le score de structure et $arbre(E_d)$ signifie que E_d remplit les propriétés d'arbre.

3.8 Indexation du contenu

L'indexation d'un document correspond dans la littérature à épurer le document (enlever les termes n'ayant aucun caractère discriminatoire : les caractères spéciaux, les chiffres, les mots vides), et à radicaliser chaque terme afin d'obtenir une forme canonique représentative de plusieurs termes appartenant à la même catégorie lexicale.
La radicalisation est une phase essentielle pour l'indexation, non seulement c'est essentiel pour l'optimisation des fonctions d'accès à l'information, mais l'expérience a montré que la performance (en terme de précision et de rappel) est relativement dépendante

de la fiabilité de l'opération d'indexation : la recherche sur des documents anglais enregistre la meilleure performance grâce à l'algorithme de Porter [79]. Les approches de radicalisation (encore appelées lemmatisation ou stemming) sont orientées langage, l'algorithme de Porter est sans doute le plus connu dans ce domaine, il se base sur un ensemble d'étapes analysant les suffixes pouvant affecter un mot, et tente de transformer de proche en proche le terme en son radical, la limite de cette approche est qu'elle n'est applicable que pour des documents de la langue anglaise.

3.8.1 Radicalisation basée sur les NGrams

L'utilisation des NGrams a été proposée essentiellement pour résoudre deux types de problèmes : la manipulation de requêtes avancées et la correction orthographique (voir annexe C).

3.8.1.1 Radicalisation de termes

Beaucoup de mots ont des formes légèrement différentes, mais leur sens reste le même ou très similaire. C'est notamment le cas des mots conjugués. Par exemple, les mots suivants ont des sens très similaires : *transformer*, *transforme*, *transforment*, *transformation*, *transformateur*...
La différence de forme entre ces mots n'est pas à considérer en RI. Au contraire, on voudrait trouver des documents sur "transformation" à partir d'une requête "transformer". Ainsi, il faut éliminer ces différences non significatives, c'est-à-dire ramener ces mots à une forme identique. Il est à noter que ces mots ont la même racine (lemme). Ainsi, si on arrive à éliminer les terminaisons de mots, et garder seulement la racine, on aura une forme identique. C'est l'idée qui conduit à utiliser la lemmatisation. Il y a plusieurs façons de lemmatiser des mots. Une première façon consiste à examiner seulement la forme du mot, et selon la forme, on essaie de déduire la racine [79]. L'algorithme de Porter élimine les terminaisons des mots en anglais en 5 grandes étapes : la première étape essaie de transformer le pluriel en singulier. Les étapes subséquentes essaient d'éliminer au fur et à mesure les dérivations (exemple : *-ness* qu'on ajoute derrière certains adjectifs (*happiness*), *-able* ajouté derrière un verbe (*adjustable*)...). Cet algorithme transforme parfois deux mots différents en une même forme. Par exemple, *derivate*/*derive*, *activate*/*active*. Cependant, pour la plupart, la transformation semble raisonnable. On peut aussi utiliser un dictionnaire dans la lemmatisation. Pour savoir si une séquence de lettres à la fin correspond à une terminaison, il suffit de faire une élimination ou une transformation tentative, et de voir si la forme obtenue existe dans le dictionnaire, si non la terminaison est incorrecte, et d'autres possibilités seront ensuite envisagées. Par exemple, on peut accepter la règle qui remplace *-ation* par *-er* pour les mots, *transform-ation*, *élimin-ation*, etc. Cependant, pour *vocation*, si on

applique cette règle on obtiendra *vocer* qui n'a pas d'existence dans le dictionnaire.
L'utilisation d'un dictionnaire est avantageuse, mais elle est au prix de disposer d'un dictionnaire. La plupart des systèmes de RI n'en disposent pas, et un tel dictionnaire électronique demeure peu accessible même s'il existe.
Une lemmatisation correcte requiert souvent une reconnaissance correcte de la catégorie grammaticale. Ainsi, on peut penser à utiliser un taggeur automatique (ou un analyseur de catégorie) dans un processus de lemmatisation. Une des approches possibles est de déterminer la catégorie d'un mot de façon probabiliste. Pour cela, il faut d'abord qu'on entraîne un modèle probabiliste en utilisant un ensemble de textes catégorisés manuellement (le corpus d'entraînement). Ce modèle détermine la probabilité qu'un mot appartient à une catégorie selon sa forme, et selon les mots qui l'entourent.
Avec ce mécanisme de reconnaissance de catégorie [11], on peut se permettre de transformer une forme de mot en une forme standard au lieu de transformer simplement la terminaison.
Durant l'indexation, on doit transformer les mots (lemmatisation), sélectionner un ensemble de termes d'index et les pondérer. Le résultat d'une indexation est donc un ensemble de termes qui peuvent être soit des mots, soit des racines de mots (soit des termes composés si on possède un mécanisme pour reconnaître des termes composés (exemple. *Base donner* pour *base(s) de données*)).

3.8.1.2 Relation d'équivalence lexicale

L'objectif principal de l'indexation est de considérer que deux termes sont lexicalement équivalents si leurs radicaux sont identiques. Nous définissons une relation d'équivalence R dans l'ensemble des termes d'une même langue par l'expression suivante :

$$\forall(t_i, t_j) \in L^2, t_i \ R \ t_j \Leftrightarrow radical(t_i) = radical(t_j) \tag{3.3}$$

où *radical* est une fonction de radicalisation et L est la langue (ensemble de mots). Cette relation est une relation d'équivalence (réflexive, symétrique et transitive). Nous pouvons alors construire des classes d'équivalence selon cette relation.
Si on considère que l'objectif d'indexation est de construire des classes d'équivalence lexicales de tous les termes, il ne serait pas nécessaire de transformer chaque terme en son radical, cette transformation est néanmoins préférable car elle permet d'optimiser largement les espaces nécessaires à la représentation des documents pour la recherche. Il est par ailleurs nécessaire de définir une autre relation d'équivalence sans passer par le calcul du radical de chaque terme. Nous définissons une relation selon le degré de ressemblance de chaque paire de termes par l'équation suivante :

$$\forall(t_i, t_j) \in L^2, t_i \ R \ t_j \Leftrightarrow sim(t_i, t_j) \geq T_{sim} \tag{3.4}$$

où *sim* est une fonction de ressemblance et T_{ress} est un seuil de ressemblance. Il n'est pas possible d'affirmer que cette relation est une relation d'équivalence car elle n'est pas nécessairement symétrique ou transitive.

	Document	**Terme**
	Termes	$NGrams$
$tf \times idf$	Fréquence d'un terme dans un document	Fréquence du $NGram$ dans un terme
Collection	L'ensemble de documents (un corpus)	L'ensemble des $NGrams$ (un très grand document)

▶ Tableau 3.1: $tf \times idf$ utilisée pour les termes et les documents

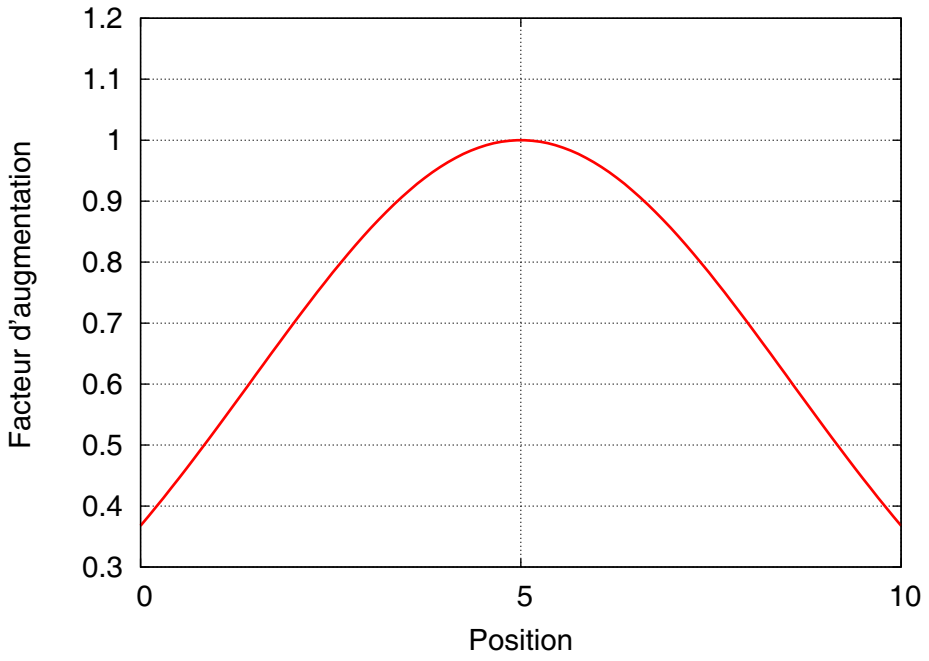
Fonction de ressemblance La mesure $tf \times idf$ utilisée en RI traduit le fait qu'un terme est important pour un document s'il apparaît souvent dans le document (tf) et peu souvent dans les autres documents (idf). Cette mesure est utilisée en RI pour calculer la pertinence d'un document par rapport à une requête, ceci traduit le fait qu'un document est pertinent à une requête s'ils partagent assez de termes importants. Nous nous sommes inspirés de cette hypothèse afin de définir une mesure de similitude entre les termes en se basant sur la même hypothèse : deux termes sont similaires s'ils partagent assez d'éléments importants. Nous avons choisi comme éléments d'un terme ses $NGrams$: un $NGram$ est défini par un facteur de longueur N d'un terme. Par exemple, les $3Grams$ du terme *recherche* sont *rec*, *ech*, *che*, *her*, *erc* et *rch*.
Un $NGram$ est important pour un terme s'il apparaît au moins une fois dans ce terme et peu dans les autres termes. Ceci implique que les $NGrams$ les moins importants pour un terme sont ceux qui apparaissent souvent dans les autres termes, or les $NGams$ les plus souvent rencontrés sont ceux qui représentent les éléments à ajouter à un terme afin d'obtenir une forme fléchée à partir de sa forme radicale, par exemple, le terme *formellement* est obtenu en ajoutant *llement* au radical *forme*.

Le tableau 3.1 présente une analogie entre l'emploi de $tf \times idf$ pour la pondération des $NGrams$ dans un terme et la pondération d'un terme dans un document. La valeur de N utilisée dans nos expérimentations est 2.

Poids d'un NGram dans un terme Nous attribuons à chaque $NGram_i$ un poids dans un terme t_j en s'inspirant du facteur $tf \times idf$, ce poids est donné par l'équation suivante :

$$p_{ij} = \frac{f_{ij}}{f_i} \tag{3.5}$$

où f_i représente le nombre de termes distincts de la collection contenant $NGram_i$ et f_{ij} représente la fréquence d'apparition de $NGram_i$ dans le terme t_j.



▶ Figure 3.29: Evolution du facteur d'augmentation de poids en fonction de la position du $NGram$

Poids de NGram en fonction de sa position La position du $NGram$ joue un rôle important pour la pondération des $NGrams$. En effet puisque nous considérons que les préfixes et les suffixes ne sont pas importants, les $NGrams$ qui apparaissent au milieu d'un terme sont probablement les plus représentatifs de son sens. Nous employons un facteur d'augmentation de poids α_{ij} défini comme suit :

$$\alpha_{ij} = exp\left(-\left(\frac{pos_{ij} - \mu_j}{\sigma_j}\right)\right)$$

où pos_{ij} est la position moyenne de $NGram_i$ dans un terme t_j, μ_j est la moyenne des positions de $NGram_i$ dans un terme t_j et σ_j est l'écart-type des positions dans un terme t_j. μ_j et σ_j valent tous les deux $\frac{l_j}{2}$ où l_j est la longueur du terme j. L'évolution du facteur d'augmentation est illustrée par la figure 3.29 dans laquelle nous considérons que la longueur du terme est 10. Le facteur d'augmentation du poids est un paramètre multiplicatif, un terme sera donc pondéré comme suit :

$$p_{ij} = \alpha_{ij} \times \frac{f_{ij}}{f_i} \tag{3.6}$$

3.8.1.3 Similitude entre deux termes

La mesure de similitude entre deux termes est donnée par l'équation suivante :

$$sim(t_i, t_j) = \sum_{g_k \in t_i \cap t_j} p_{ki} \times p_{kj} - \sum_{g_k \in t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j} p_{ki} \times p_{kj} \tag{3.7}$$

où g_i est le $NGram$, $t_i \cap t_j$ est l'ensemble des $NGrams$ appartenant à t_i et t_j, $t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j$ est l'ensemble de $NGrams$ appartenant exclusivement à t_i ou à t_j.
Selon l'équation 3.7, un terme est un ensemble de $NGrams$, deux termes sont équivalents s'ils partagent assez de $NGrams$ ($\sum_{g_k \in t_i \cap t_j} p_{ki} \times p_{kj}$) et si les poids des $NGrams$ non communs sont faibles ($\sum_{g_k \in t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j} p_{ki} \times p_{kj}$).
Afin d'en tenir compte, le seuil de ressemblance est fonction de la ressemblance de chaque terme avec lui même moyennant un paramètre multiplicatif qu'on définit expérimentalement.

La relation R est alors définie formellement par :

$$\forall (t_i, t_j) \in L^2, t_i R t_j \Leftrightarrow sim(t_i, t_j) \geq S \times \max(sim(t_i, t_i), sim(t_j, t_j)) \tag{3.8}$$

où S est un paramètre multiplicatif inférieur à 1. Nous choisissons typiquement la valeur approchant aussi bien que possible la relation R à une relation d'équivalence : la valeur laissant construire des classes de termes les plus homogènes et exhaustifs possibles[2].

3.8.2 Pondération d'un terme dans un élément XML

En RIS, le calcul de l'importance d'un terme t_i (poids) peut se calculer de deux manières :

- Comme en RI traditionnelle, le poids d'un terme se base sur la fréquence d'un terme dans le document (tf_d) et le nombre de documents contenant le terme (idf).
- En tenant compte d'autres critères liés aux propriétés des documents XML, permettant de mesurer l'importance d'un terme dans un nœud, il suffit de calculer la fréquence de ce terme dans l'élément qui le contient (tf_e) et le nombre d'éléments contenant le terme (ief).

[2]Nous pourrons utiliser des mesures de rappel et de précision relatives à la classification.

Nos expérimentations sont réalisées en utilisant ces deux mesures.
Les mesures $tf \times idf$ et $tf \times ief$ ont pour principal objectif d'écarter les termes non significatifs [6]. Or le nom d'une balise est constitué d'un ou de plusieurs termes qui ne sont pas toujours significatifs. Nous admettons que si le nom d'une balise n'est pas significatif alors la balise n'est pas significative : ceci est le cas des balises de mise en forme[3]. Chaque balise est alors pondérée selon son nombre moyen d'enfants.

3.8.3 Poids des balises

Le système devra écarter toutes les balises de mise en forme. Celles-ci ont une propriété très spécifique : elles ne possèdent généralement pas d'éléments enfants. Si une balise apparaît souvent comme une balise feuille, elle aura tendance à devenir une balise de mise en forme.
On considère par *poids de balise* son importance dans la structure du document ou de la requête, plus le poids diminue, moins elle est révélatrice de la structure du document. On utilise l'équation suivante pour évaluer le poids d'une balise B :

$$w(B) = 1 - \frac{N_F(B)}{N(B)}$$

où $N_F(B)$ est le nombre de fois où la balise B apparaît comme un élément feuille et $N(B)$ est le nombre total d'apparitions de la balise B. Le poids de la balise est donc inversement proportionnel à sa probabilité d'apparition comme un élément feuille d'un document tiré aléatoirement de la collection ($\frac{N_F(B)}{N(B)}$).
Les poids des balises sont dans l'intervalle $[0, 1]$, la valeur 0 est le poids de toutes les balises qui apparaissent toujours comme des éléments feuilles.
Pour mettre en relief le poids des balises pour la gestion de la structure, on pourra l'intégrer dans l'estimation des distances entre les éléments d'un même arbre XML.

Lors de la pondération des chemins, le poids d'un chemin $N_1 \longrightarrow N_2 \dots N_n$ est exprimé en fonction de la distance $d(N_1, N_n) = exp\,(1 - d(N_1, N_n))$, qui ne tient pas compte des poids des balises $N_2 N_3 \dots N_{n-1}$. Pour être conforme à cette hypothèse, plus les poids des balises augmentent, plus la distance entre N_1 et N_n augmente et plus le poids w du chemin $N_1 \xrightarrow{w} N_n$ augmente en conséquence.

[3] Les documents de la campagne INEX contiennent des balises qui n'ont généralement pas d'éléments enfants, nous soupçonnons que ces balises sont utilisées pour la mise en forme des documents quand ils sont associés à des feuilles de style de mise en forme (CSS (*Cascading Style Sheets*), XSL...)

La nouvelle distance d^* entre deux balises doit alors vérifier les conditions suivantes :

$$d^*(N_1, N_n) \begin{cases} = 1 & \text{si } w(N_i) = 0 \ \forall \ 2 \leq i \leq n-1 \\ = n-1 & \text{si } w(N_i) = 1 \ \forall \ 2 \leq i \leq n-1 \\ \in]0, n-1[& \text{si non} \end{cases}$$

La valeur de α est déterminée en fonction des poids des balises $N_2, N_3 \ldots N_{n-1}$:

$$\alpha = \sum_{i=2}^{n-1} w(N_i)$$

En d'autres termes, la distance d est la limite de d^* lorsque le poids des balises incrustés entre les extrémités du chemin que l'on cherche à pondérer tendent tous vers le poids idéal 1.
On pourra alors concevoir les arcs virtuels en changeant notre conception de la distance entre les balises. Dans la suite de ce rapport, on utilise le terme d pour désigner la distance entre balises. Les deux distances ont été expérimentées.

3.8.4 Propagation de texte

Le contenu textuel apparaît généralement dans les nœuds feuilles. Cependant, un nœud interne peut être pertinent même s'il ne contient aucun terme d'indexation, la pertinence ne provient pas seulement de la structure. Afin de rétablir l'ordre entre les nœuds pour que les feuilles ne soient pas favorisées par rapport aux nœuds internes, ceux-ci doivent avoir un score relatif au contenu.
On distingue dans la littérature deux principaux courants, la propagation des scores : on propage le score d'un nœud pertinent (en terme de contenu) à une requête à ses ancêtres [36] [41] [108]. Les autres modèles se basent sur la propagation du contenu [70] [114] au lieu de la propagation du score, on propage le contenu d'un nœud vers un autre via les liens de descendance et on calcule son score indépendamment. Nous adoptons 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.

3.8.4.1 Propagation basée sur la profondeur

Nous traduisons le contenu de chaque nœud feuille par un ensemble de termes pondérés, ceux-ci seront propagés vers les ancêtres de ce nœud (Ceux qui propagent le texte et ceux qui propagent les scores) tout en diminuant leurs poids en fonction de la distance parcourue au moment de la propagation.

Par ailleurs, un nœud interne peut recevoir du contenu à partir de plusieurs nœuds feuilles. L'élément particulier racine reçoit le contenu de toutes les feuilles de l'arbre XML. De ce fait le même terme peut appartenir à plusieurs nœuds, le poids final associé à un nœud interne sera alors la somme de tous les poids reçus par les nœuds feuilles descendants de celui-ci :

$$w(t,n) = ief_t \times \sum_{n \to c_1 \to c_2 \ldots c_k} \frac{tf(t,c_k)}{d(n,c_k)+1} \tag{3.9}$$

où $w(t,n)$ est le poids du terme t dans le nœud n, $tf(p,c_k)$ est la fréquence du terme t dans le nœud c_k (un descendant direct ou indirect de n) et ief_p est inversement proportionnel au nombre de nœuds de la collection contenant le terme t :

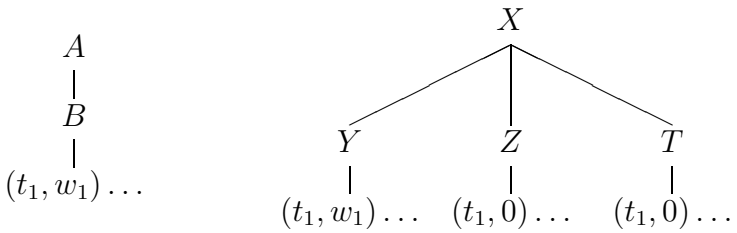
$$ief_p = \frac{N}{N_t} \tag{3.10}$$

où N est le nombre d'éléments dans la collection et N_t est le nombre d'éléments contenant le terme t. $n \to c_1 \to c_2 \ldots c_k$ est une branche de l'arbre en question ($n_i \to n_j$ signifie que n_i est le père de n_j).
L'équation 3.9 montre que le poids d'un terme propagé diminue par propagation, en effet celui-ci est divisé par la distance $d(n,c_k)$ entre le nœud vers lequel le terme sera propagé (n) et un nœud feuille descendant c_k qui le contient. Les descendants et la requête sont aussi sujets à la propagation du contenu.

L'équation 3.9 montre que le contenu de chaque nœud est propagé à ses nœuds ancêtres. Par exemple, si le nœud A est le parent du nœud B qui contient le terme p , nous supposons que ce nœud A contient le terme p et nous diminuons son score dans le nœud A en le divisant par $2 = d(A,B) + 1$ [106]. Si le nœud C est le parent du nœud A , le terme p est propagé du nœud B vers le nœud C et on divise son poids par $3 = d(C,B) + 1$.

3.8.4.2 Propagation basée sur la profondeur et la largeur

La propagation de texte est souvent réalisée en fonction de la distance qui sépare l'élément feuille qui contient du texte et l'élément nœud interne qui est censé recevoir le texte. Le facteur de propagation est calculé en fonction de cette distance. La profondeur du sous arbre dans lequel ces éléments sont imbriqués n'intervient pas dans l'estimation de ce facteur.
Nous admettons que la profondeur est une propriété très intéressante, voire indispensable à prendre en considération. La figure 3.30 illustre un exemple de deux fragments de profondeur 1. La propagation du terme t_1 de l'élément B vers l'élément A est réalisée avec le même facteur de propagation que celui utilisé pour la propagation du même terme de l'élément Y vers l'élément X. Dans le cas où le poids w_1 de t_1 dans les



▶ Figure 3.30: Effet du nombre de fils

éléments B et Y est le même, et si le terme n'apparait dans aucun autre élément, ceci entraîne des poids identiques dans les éléments A et X. Or intuitivement nous pensons que ce terme est plus important pour l'élément A que pour l'élément X, car il lui est très spécifique. Par ailleurs, si le terme apparaît dans les éléments Y, Z et T avec le même poids, le terme pourra avoir un poids plus élevé dans le nœud X que dans le nœud Y : la propagation du texte dans ce cas aura des effets contradictoires bien qu'on utilise un facteur de diminution de poids, on peut se retrouver avec des poids propagés plus élevés.
Si le nœud A est le parent du nœud B qui contient le terme t de poids w, ce poids devrait être diminué dans le nœud A, si le nombre de ses fils augmente. Nous proposons une fonction qui tient compte du nombre de fils lors de la propagation par ce qui suit :

$$w(p,n) = ief_p \times \sum_{n \to n'} \frac{tf(p,n')}{|Children(n)|} \tag{3.11}$$

où $|Children(n)|$ est le nombre total des nœuds fils de l'élément n. L'équation 3.11 montre que le contenu est seulement propagé d'un nœud vers son parent, par transitivité, il atteint tous ses ancêtres, par conséquent :

$$w(p,n) = ief_p \times \sum_{n \to c_1 \to c_2 \ldots c_k} \frac{tf(p,c_k)}{\prod_{e \in \{n, c_1, c_2 \ldots c_{k-1}\}} |Children(e)|} \tag{3.12}$$

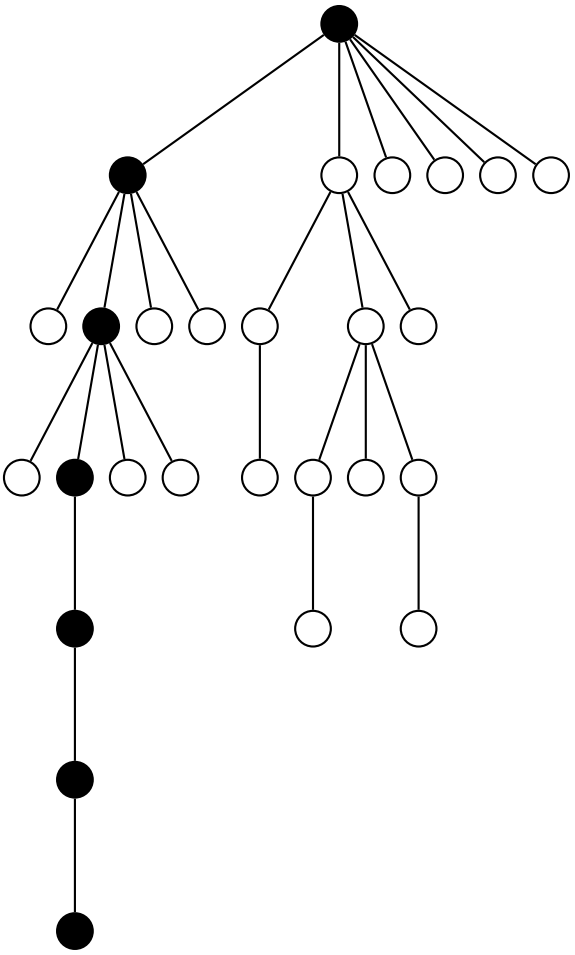
Le poids du terme t propagé est inversement proportionnel aux nombres de fils des ancêtres à qui le terme t appartient.

3.8.5 Fonction de lissage du nombre de fils

La prise en compte du nombre de fils permet d'éviter les inconvénients de la propagation basée sur la profondeur. L'idée sur laquelle nous sommes partis est de considérer que plus le nombre de fils augmente, moins la relation entre les éléments est élevée et par conséquent moins le facteur de propagation est élevé.

Le nombre de fils est par ailleurs très variable dans un document XML, il peut varier de quelques éléments à quelques dizaines, voire quelques centaines. Dans un

même fragment, au lieu de considérer le nombre de fils réel, qui ne tient pas compte de la profondeur du fragment le plus imbriqué dans l'arbre entier pour lequel il représente la racine, nous faisons appel à une fonction de lissage qui détermine le nombre de fils en fonction de la profondeur moyenne du fragment et le nombre d'éléments total.
Mais tout d'abord, on est amené à définir ce qu'est la profondeur moyenne d'un arbre XML. Il s'agit simplement de la moyenne des distances entre l'élément racine de l'arbre et ses éléments feuilles. La figure 3.31 montre un exemple de fragment XML, la profondeur réelle de l'arbre est 6 mais l'exemple montre que seule une branche a cette longueur, les autres branches ont des longueurs sensiblement inférieures, de plus, ils représentent la grande majorité des branches de l'arbre. Par le calcul de la moyenne



▶ Figure 3.31: Effet du nombre de fils

sur les longueurs de toutes les branches de l'arbre T, la profondeur sera estimée à :

$$\frac{\sum_{n \in leafs(T)} d(root(T), n)}{|leaf(T)|}$$

au lieu de :

$$\max_{n \in leafs(T)} d(root(T), n)$$

où $leafs(T)$ est l'ensemble des feuilles de l'arbre T et $root(T)$ est sa racine. Dans l'exemple de la figure 3.31, la profondeur de l'arbre devient :

$$\frac{2+3+6+3+3+2+2+3+4+3+4+2+1+1+1+1}{16} = 2.56$$

Le nombre de fils réel de l'élément racine est 6, cependant, l'exemple laisse constater que cette propriété n'est pas réaliste, car pour une profondeur de 2.56, et si chaque élément de l'arbre possède 6 fils, on s'attend à se retrouver avec 117.62 éléments (l'équation 3.13 montre comment retrouver cette valeur), or l'arbre contient 27 éléments.

Afin de retrouver des estimations non biaisées du nombre de fils d'un élément A, on considère le fragment le plus large et le plus ramifié possible qui vérifie les propriétés de l'arbre et pour lequel A représente la racine. Soient T_A ce fragment, $p = depth(T_A)$ sa profondeur et $n = N(T_A)$ le nombre d'éléments total de T_A. Si tous les éléments possèdent le même nombre de fils x (que l'on cherche à définir), alors le nombre de fils de A sera x, chaque fils de A aura de même x enfants d'où le nombre de petits fils de A sera $x^2 \ldots$ le nombre d'éléments total sera alors :

$$N(T_A) = 1 + x + x^2 + \ldots x^p = \frac{x^{p+1} - 1}{x - 1} \tag{3.13}$$

L'estimation de x est alors basée sur l'équation 3.13 :

$$x \approx \left(\frac{N(T_A)}{depth(T_A) + 1} \right)^{\frac{2}{depth(T_A)}} \tag{3.14}$$

L'annexe B fournit les détails de la résolution de l'équation 3.13 pour aboutir à l'équation 3.14. L'estimation du nombre de fils à partir de l'équation 3.14 est une fonction de lissage de la profondeur et du nombre de fils de l'arbre, le nombre de fils ainsi calculé permet de donner une appréciation de la largeur de l'arbre. La racine de l'arbre de la figure 3.31 admet alors un nombre de fils calculé comme suit :

$$x \approx \left(\frac{N(T_A)}{depth(T_A) + 1} \right)^{\frac{2}{depth(T_A)}} = \left(\frac{27}{3.56} \right)^{\frac{2}{2.56}} = 4.86$$

Une fois la représentation des documents en termes de contenu et de structure effectuée, on peut calculer la valeur de similarité entre un document XML et une requête.

3.9 Appariement document-requête par le contenu

Nous utilisons le produit scalaire entre les vecteurs associés à chaque nœud :

$$rsv_c(n, n') = \sum_{p \in n \cap n'} w(p, n) \times w(p, n') \tag{3.15}$$

où p est un terme d'indexation et $rsv_c(n, n')$ est le score relatif aux contenu de n. Nous calculons le score d'un nœud n par rapport à un nœud $n^{'}$. n est alors un nœud document et n est son nœud correspondant de la requête. Le score du contenu, combiné au score relatif à la structure, donne lieu au score final entre n et $n^{'}$. Ce score représente le score final du nœud n par rapport à toute la requête. Le calcul du score sur le contenu de chaque nœud est réalisé indépendamment de la structure :

$$rsv_c(n) = \sum_{t \in n \cap q} w(t, n)$$

où $w(p, n)$ est le poids de terme t dans un nœud n et $n \cap q$ est l'ensemble de termes appartenant à la requête et au nœud n.

3.10 Appariement document-requête par le contenu et la structure

Le score final dépend des scores estimés selon les deux critères contenu et structure. Dans le paragraphe 3.7.2, le score sur la structure rsv_s est relatif à un fragment, aucun nœud ne possède un poids sur la structure. Afin d'attribuer un score à chaque nœud du fragment, et puisque chaque nœud de celui-ci contribue à l'augmentation de son score, ce dernier sera associé à chaque nœud de ce fragment. Par ailleurs, un nœud peut apparaître dans plusieurs fragments, nous lui attribuons le score le plus élevé d'un fragment potentiellement pertinent où il apparaît. Combiné avec le score du contenu, le score de la structure engendre un score final qui servira à ordonner les nœuds selon leur pertinence.

3.10.1 Approche par combinaison linéaire

Notre objectif est de combiner l'information portée par la structure et celle portée par le contenu. Pour combiner ces deux informations (contenu et structure) nous utilisons une combinaison linéaire :

$$rsv(E_q, E_d) = \alpha \times \ rsv_c(E_q, E_d) + (1 - \alpha) \times \ rsv_s(E_q, E_d) \tag{3.16}$$

où E_q (resp. E_d) est un fragment de la requête (resp. du document) et $\alpha \in [0, 1]$ est un paramètre permettant de renforcer la recherche selon le contenu ou la structure. Cette valeur est définie par l'expérimentation mais dans des conditions réelles, elle peut être définie par l'utilisateur. Expérimentalement, le domaine de définition des scores orientés contenu est différent de ceux orientés structure, il est très difficile de les prévoir car ils dépendent principalement des conditions expérimentales, de la nature

des documents traités, de la complexité structurelle des requêtes...
L'identification de la valeur de α, qui représente le meilleur compromis entre le contenu et la structure, est alors très difficile, elle ne pourra être fixée que par l'expérimentation, or selon notre approche, elle doit être fixée au lancement du système, et indépendamment des caractéristiques du corpus et des requêtes traitées.

3.10.2 Approche par combinaison des distributions

Dans la seconde approche, le score final représente une probabilité de pertinence. Le problème est défini par la probabilité de pertinence d'un nœud sachant son score sur le contenu c et son score sur la structure s : $p(R|rsv_s = s, rsv_c = c)$.
Puisque le calcul des scores sur le contenu et la structure sont indépendants, les évènements $rsv_s = s$ et $rsv_c = c$ sont indépendants, par conséquent $p(rsv_s = s, rsv_c = c) = p(rsv_s = s) \times p(rsv_c = c)$ et $p(rsv_s = s, rsv_c = c|R) = p(rsv_s = s|R) \times p(rsv_c = c|R)$ d'où :

$$\begin{aligned} p(R|rsv_s = s, rsv_c = c) &= \frac{p(rsv_s=s,rsv_c=c|R)\times p(R)}{p(rsv_s=s,rsv_c=c)} \\ &\propto \frac{p(rsv_s=s|R)\times p(rsv_c=c|R)}{p(rsv_s=s)\times p(rsv_c=c)} \end{aligned}$$

3.10.2.1 Distribution des scores du contenu

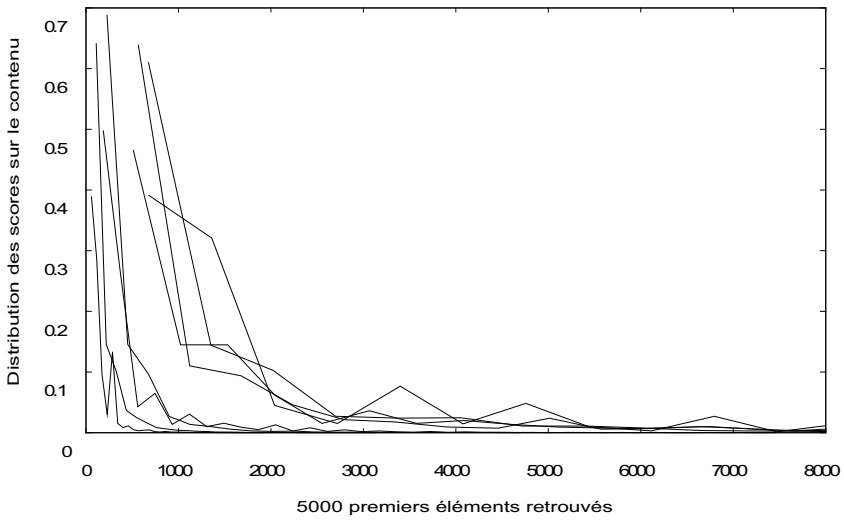
Nous nous sommes appuyés sur le corpus issu d'INEX pour estimer cette loi ainsi que ses paramètres. Nous avons considéré les 5000 scores orientés contenu les plus élevés pour chaque requête. La distribution de ces scores suit une loi de probabilité. La figure 3.32 montre que cette distribution suit la loi exponentielle, cette fonction est définie comme suit :

$$p(rsv_c = c) = \lambda_C \times exp^{-\lambda_C \times c}$$

Les scores sur le contenu des éléments pertinents suit également une loi exponentielle :

$$p(rsv_c = c|R) = \lambda_C^R \times exp^{-\lambda_C^R \times c}$$

λ_c et λ_c^R sont les paramètres respectifs des lois de distribution des scores sur le contenu de tous les éléments et des éléments pertinents.



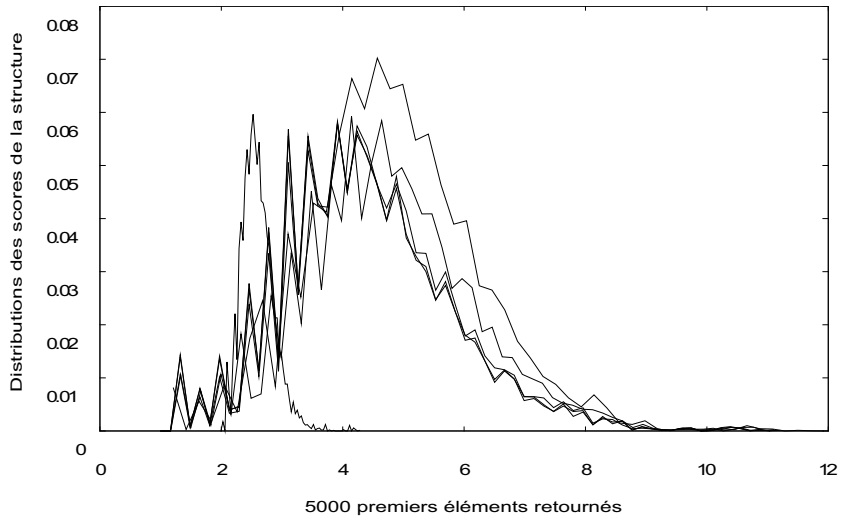
► Figure 3.32: Distribution des scores sur le contenu pour les requêtes de type VVCAS (INEX'2005)

3.10.2.2 Distribution des scores de la structure

La figure 3.33 montre que la distribution des scores sur la structure suit une loi normale, cette fonction est définie comme suit :

$$p(rsv_s = s) = \frac{1}{\sigma_S\sqrt{2 \times \pi}} \times exp^{-\frac{1}{2}\times\left(\frac{s-\mu_S}{\sigma_S}\right)^2}$$

De même, les éléments pertinents ont la même propriété :



► Figure 3.33: Distribution des scores sur la structure pour les requêtes de type VVCAS (INEX'2005)

$$p(rsv_s = s|R) = \frac{1}{\sigma_S^R\sqrt{2\times\pi}} \times exp^{-\frac{1}{2}\times\left(\frac{s-\mu_S^R}{\sigma_S^R}\right)^2}$$

Sur dix requêtes de type VVCAS, deux seulement ne vérifient pas la distribution normale, ceci est dû au fait que ces requêtes sont faiblement structurées (ces requêtes ne contiennent qu'un seul nœud), les scores sur la structure de tous les nœuds de la collection sont alors 0 ou 1.

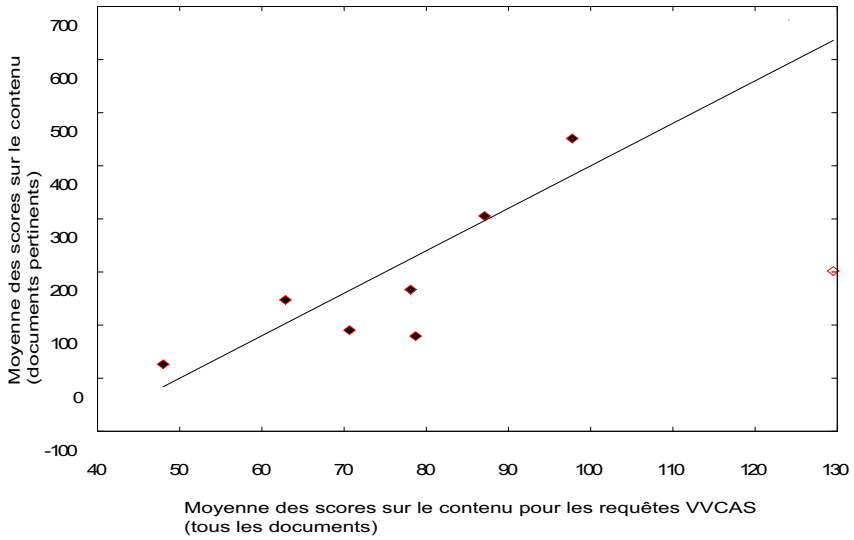
3.10.2.3 Combinaison des scores selon leurs distributions

Pour combiner les deux scores sur le contenu et sur la structure d'une manière optimale, il faut trouver le meilleur compromis pour les combiner, il suffit de déterminer les paramètres λ_C, λ_C^R, μ_S, μ_S^R, σ_S et σ_S^R. En s'appuyant sur la méthode d'estimation par le maximum de vraisemblance, ces paramètres sont estimés comme suit :

$$\begin{aligned}
\lambda_C &= \frac{1}{\mu_C} \\
\lambda_C^R &= \frac{1}{\mu_C^R} \\
\sigma_S &= \frac{1}{|B|}\sqrt{\sum_{n\in B}(rsv_s(n)-\mu_S)^2} \\
\sigma_S^R &= \frac{1}{|B\cap R|}\sqrt{\sum_{n\in B\cap R}(rsv_s(n)-\mu_S^R)^2} \\
\mu_C &= \frac{1}{|B|}\sum_{n\in B} rsv_c(n) \\
\mu_C^R &= \frac{1}{|B\cap R|}\sum_{n\in B\cap R} rsv_c(n) \\
\mu_S &= \frac{1}{|B|}\sum_{n\in B} rsv_s(n) \\
\mu_S^R &= \frac{1}{|B\cap R|}\sum_{n\in B\cap R} rsv_s(n)
\end{aligned}$$

où μ_S (resp. μ_C) est la moyenne de tous les scores des nœuds sur la structure (resp. sur le contenu), μ_S^R (resp. μ_C^R) est la moyenne sur la structure (resp. sur le contenu) pour les nœuds réellement pertinents. B est l'ensemble des nœuds de la collection et R est l'ensemble des nœuds pertinents.
Intuitivement, aucune source d'évidence n'est fournie pour l'évaluation de λ_C^R, μ_S^R et

σ_S^R, en effet dans les conditions d'expérimentation, aucun nœud approprié n'est connu. Quelques expériences prouvent qu'il y a une relation entre μ_C^R et μ_C, σ_S^R et σ_S et μ_S^R et μ_S. Chaque relation est stable pour les différentes requêtes d'INEX'2005, dans nos expérimentations nous avons considéré les 5000 premiers nœuds retournés par la recherche orientée contenu et celle orientée structure. Les Figures 3.34 et 3.35 montrent nos hypothèses. Considérons les fonctions r_λ, r_μ et r_σ définies par $\mu_C^R = r_\lambda(\mu_C)$,



► Figure 3.34: Moyenne des scores sur le contenu des éléments pertinents comme fonction de celle de tous les éléments (5000 premiers éléments retrouvés)

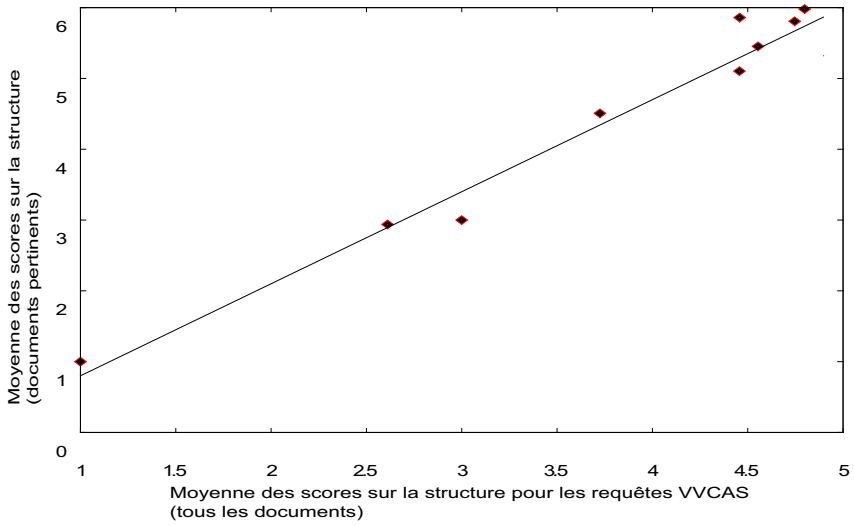
$\mu_S^R = r_\mu(\mu_S)$ et $\sigma_S^R = r_\sigma(\sigma_S)$, le score final d'un nœud N sera alors :

$$rsv = \frac{\frac{1}{\mu_C^R} \times exp^{-\frac{1}{\mu_C^R} \times c} \times \frac{1}{\sigma_S^R \sqrt{2 \times \pi}} \times exp^{-\frac{1}{2} \times \left(\frac{s - \mu_S^R}{\sigma_S^R}\right)^2}}{\frac{1}{\mu_C} \times exp^{-\frac{1}{\mu_C} \times c} \times \frac{1}{\sigma_S \sqrt{2 \times \pi}} \times exp^{-\frac{1}{2} \times \left(\frac{s - \mu_S}{\sigma_S}\right)^2}}$$

$$\propto \quad 2 \times \left(\frac{1}{\mu_C} - \frac{1}{\mu_C^R}\right) \times c + \left(\frac{s - \mu_S}{\sigma_S}\right)^2 - \left(\frac{s - \mu_S^R}{\sigma_S^R}\right)^2$$

$$= \quad 2 \times \left(\frac{1}{\mu_C} - r_\lambda\left(\frac{1}{\mu_C}\right)\right) \times c + \left(\frac{s - \mu_S}{\sigma_S}\right)^2 - \left(\frac{s - r_\mu(\mu_S)}{r_\sigma(\sigma_S)}\right)^2$$

Les fonctions r_λ (voir figure 3.34) et r_μ (voir figure 3.35) sont des fonctions affines. Elles seront définies et expérimentées dans le prochain chapitre.



► Figure 3.35: Moyenne des scores sur la structure des éléments pertinents comme fonction de celle des tous les éléments (5000 premiers éléments retrouvés)

3.11 Conclusion

Nous avons présenté dans ce chapitre une approche de recherche orientée structure basée sur la comparaison d'arbres. Combinée à un modèle d'indexation et d'appariement orienté structure, il permet de contribuer à l'estimation des scores de tous les granules documentaires de la collection. Nous avons également proposé un modèle d'appariement orienté contenu, qui se base sur la propagation de texte. La propagation est basée sur les propriétés géométriques de l'arbre et des sous arbres relatifs aux documents traités. La séparation entre les deux types d'appariement permet de mesurer l'apport de chacun dans la RIS. Dans le prochain chapitre, nous explorons les conditions expérimentales ainsi que les résultats obtenus.

Chapitre 4

Expérimentations et résultats

4.1 Introduction

Dans ce chapitre, nous présentons les expérimentations effectuées pour évaluer l'apport des différentes propositions faites au chapitre 3. Nos évaluations concernent deux collections issues de la campagne d'évaluation INEX.
De 2004 à 2005, les types de requêtes ont évolué :

- INEX'2004 : requêtes de type CO (*Content Only*), requête de type CAS (*Content And Structure*),
- INEX'2005 : requêtes de type CO, requêtes de type CO+S et requêtes de type CAS (VVCAS, SVCAS, VSCAS et SSCAS).

Nous nous focalisons sur la tâche CAS et particulièrement la tâche VVCAS, en effet elle représente une excellente illustration de la recherche structurée : notre système est conçu principalement pour répondre à cette tâche.
Nous commençons par expérimenter la recherche orientée contenu, qui servira de référence pour mesurer l'impact de la structure en RIS. Nous évaluons ensuite les différentes propositions relatives à la gestion de la structure. Nous évaluons l'apport de chaque orientation de recherche par leur combinaison selon les techniques précédemment mentionnées.
Nous interprétons dans ce chapitre les différentes méthodes de pondération adaptées à la recherche d'information structurée, et leurs impacts sur la performance de notre système.
Nous comparons nos résultats avec les participants d'INEX, plus particulièrement avec les meilleurs résultats (Max), les résultats les plus faibles (Min) et la moyenne des résultats (Moy).

4.2 INEX 2004

L'ensemble de requêtes fournies pour la campagne d'évaluation 2004 pour la tâche CO sont composées de 40 requêtes qui ont été mises à disposition des participants (avec 34 jugements de pertinence associés).
La tâche VCAS représente la seule tâche pour les requêtes de type CAS, les participants peuvent répondre de manière vague, c'est à dire même les parties qui répondent partiellement en terme de structure à une requête peuvent être jugées pertinentes. La tâche VCAS en 2004 comporte 35 requêtes et 26 jugements de pertinence associés.
Pour calculer le score final d'un nœud du document, nous avons besoin d'agréger les deux scores sur le contenu et la structure dans la même fonction, pour cela nous utilisons une combinaison linéaire de ces deux scores comme le montre l'équation 3.16. Pour les requêtes de type CO, il n'y a pas de contraintes structurelles, pour cela nous choisissons $\alpha = 0$.

4.2.1 Expérimentation de la recherche par le contenu

La tâche CO a pour but de répondre avec des éléments/documents XML à des requêtes utilisateur contenant de simples mots-clés. Aucune indication de structure dans la requête ne peut aider les SRI à déterminer la granularité de l'information à renvoyer. Dans nos expérimentations, nous utilisons les ensembles de requêtes fournis en 2004.

Le tableau 4.1 montre les résultats obtenus pour les fonctions de pondération suivantes :

- **Pondération basée sur** ief **:** le calcul du score sur le contenu est expliqué dans la section 3.8.2. Les résultats obtenus dans le tableau 4.1 sont désignés par NA_1, le cœfficient de propagation de texte utilisé est $\lambda = 0.9$.
- **Pondération basée sur** idf **:** le calcul du score sur le contenu est expliqué dans la section 3.8.2. Les résultats obtenus dans le tableau 4.1 sont désignés par NA_2.

Nous constatons que la pondération des termes selon idf est plus adaptée aux fonctions d'évaluation orientées spécificité. Ceci s'explique par le fait que les termes contribuent à la spécificité d'un nœud s'ils sont pondérés en fonction de leur appartenance à tout le document, alors que ief privilégie les nœuds dont le texte de la requête lui appartient exclusivement des autres nœuds du même document, nous constatons une nette amélioration dans les résultats relatifs aux fonctions orientées exhaustivité pour cette fonction de pondération.

Quantification	NA_1	NA_2	Max	Moy	Min
f_{strict}	0.0825	0.0625	0.0987	0.0820	0.0661
f_{gen}	0.0527	0.0554	0.1291	0.0878	0.0524
f_{sog}	0.0449	0.0525	0.1105	0.0825	0.0491
f_{s3_e321}	0.0415	0.0521	0.0973	0.0744	0.0502
f_{s3_e32}	0.0540	0.0561	0.1027	0.0816	0.0707
f_{e3_s321}	0.1086	0.0741	0.1967	0.1210	0.0760
f_{e3_s32}	0.0978	0.0106	0.1732	0.1124	0.0848
$overlap$	60.115%	64.13%	81.85%	76.619%	64.29%

► Tableau 4.1: Résultats comparatifs avec les participants officiels d'INEX'2004 pour la tâche CO

4.2.2 Expérimentation de la recherche par le contenu et la structure

Les requêtes de cette tâche prennent en compte les contraintes structurelles, pour cette raison nous avons testé plusieurs valeurs de α, les meilleurs résultats sont obtenus pour $\alpha = 0.6$. Le tableau 4.2 illustre les meilleurs résultats obtenus en utilisant la formule de pondération suivante : $w(p,n) = ief_p \times \sum_{n \to c_1 \to c_2 \ldots c_k} \frac{tf(p,c_k)}{d(n,c_k)+1}$ (voir équation 3.9).

Les résultats obtenus des participants d'INEX sont très proches les uns des autres. En effet, bien que la tâche porte essentiellement sur la structure, les requêtes ne sont pas forcément structurées et par conséquent, l'élément fondamental de pertinence reste le contenu : les systèmes qui traitent correctement le contenu obtiennent les meilleurs résultats sans trop investir sur la structure.
Les équations de traitement du contenu que nous avons utilisées nous classent dans des rangs relativement inférieurs à nos attentes car nous donnons beaucoup d'importance à la structure (nous utilisons $\alpha = 0.6$ qui laisse la recherche orientée structure contribuer de 40% aux résultats du SRI).

Quantification	$Notre\ Approche$	Max	Moy	Min
f_{strict}	0.0771	0.0987	0.0820	0.0661
f_{gen}	0.0471	0.1291	0.0878	0.0524
f_{sog}	0.0328	0.1105	0.0825	0.0491
f_{s3_e321}	0.0229	0.0973	0.0744	0.0502
f_{s3_e32}	0.0317	0.1027	0.0816	0.0707
f_{e3_s321}	0.0923	0.1967	0.1210	0.0760
f_{e3_s32}	0.0599	0.1732	0.1124	0.0848

► Tableau 4.2: Résultats comparatifs, avec les participants officiels d'INEX pour la tâche VCAS

4.3 INEX 2005

L'ensemble de requêtes fournies pour la campagne d'évaluation 2005 pour les tâches CO et COS sont composées de 40 requêtes (avec 29 jugements de pertinence associés) qui traitent principalement le contenu.
La tâche VCAS en 2004 a été remplacée par cinq tâches, la tâche COS qui reprend le même besoin en information sur le contenu que les requêtes de type CO, le rôle de cette tâche est de mesurer l'impact de la structure dans les SRIS. Quatre autres tâches orientées contenu et structure (VVCAS, SSCAS, SVCAS et VSCAS) ont été proposées, ces requêtes contiennent des informations sur le contenu et la structure.

4.3.1 Expérimentation de la recherche par le contenu

Les tableaux 4.3 et 4.4 illustrent les résultats obtenus pour la tâche CO avec et sans chevauchement. Nous avons utilisé la fonction de pondération basée sur ief. Les résultats sont désignés par les lignes du tableau 4.3 où $\alpha = 0$.

Quantification		$nxCG[10]$	$nxCG[25]$	$nxCG[50]$	$MAep$
f_{strict}	$\alpha = 0$	0.0370	0.0422	0.0436	0.0090
	$\alpha = 0.6$	0.0363	0.0441	0.0535	0.0093
	Amélioration	-1.8919%	4.5024%	22.7064%	3.3333%
	Max	0.1588	0.1996	0.2510	0.0948
	Moy	0.0471	0.0590	0.0713	0.0197
	Min	0.0000	0.0024	0.0012	0.0002
f_{gen}	$\alpha = 0$	0.1000	0.0980	0.0970	0.0360
	$\alpha = 0.6$	0.1005	0.0950	0.1011	0.0360
	Amélioration	0.5000%	-3.0612%	4.2268%	0%
	Max	0.2908	0.2520	0.2439	0.1132
	Moy	0.1777	0.1508	0.1444	0.0576
	Min	0.0070	0.0028	0.0014	0.0004
$f_{genLifted}$	$\alpha = 0$	0.1060	0.0940	0.0970	0.0380
	$\alpha = 0.6$	0.1060	0.0953	0.0970	0.0380
	Amélioration	0%	1.3830%	0%	0%
	Max	0.3194	0.2579	0.2598	0.1400
	Moy	0.1910	0.1559	0.1473	0.0651
	Min	0.0076	0.0031	0.0016	0.0005

► Tableau 4.3: Résultats comparatifs avec les participants officiels d'INEX'2005 et impact du paramètre α pour les quantifications avec chevauchement, tâche CO ($\alpha = 0$) et COS ($\alpha = 0.6$)

Quantification		$nxCG[10]$	$nxCG[25]$	$nxCG[50]$	$MAep$
f_{strict}	$\alpha = 0$	0.0502	0.0498	0.0626	0.017
	$\alpha = 0.6$	0.05	0.0478	0.0835	0.0176
	Amélioration	-0.3984%	-4.0161%	33.3866%	3.5294%
	Max	0.0824	0.1050	0.1366	0.0390
	Moy	0.0268	0.0418	0.0491	0.0113
	Min	0.0000	0.0240	0.0024	0.0003
f_{gen}	$\alpha = 0$	0.1459	0.152	0.159	0.059
	$\alpha = 0.6$	0.1593	0.1632	0.1688	0.0592
	Amélioration	9.1844%	7.3684%	6.1635%	0.3390%
	Max	0.3153	0.2858	0.2665	0.0962
	Moy	0.2098	0.1917	0.1724	0.0462
	Min	0.0572	0.0464	0.0024	0.0027
$f_{genLifted}$	$\alpha = 0$	0.172	0.175	0.175	0.036
	$\alpha = 0.6$	0.172	0.1757	0.176	0.0360
	Amélioration	0%	0.4%	0.5714%	0%
	Max	0.3375	0.3045	0.2792	0.0705
	Moy	0.2275	0.2049	0.1792	0.0337
	Min	0.0594	0.0470	0.0368	0.0016

▶ Tableau 4.4: Résultats comparatifs avec les participants officiels d'INEX'2005 et impact du paramètre α pour les quantifications sans chevauchement, tâche CO ($\alpha = 0$) et COS ($\alpha = 0.6$)

4.3.2 Impact de la structure dans la RIS

La tâche COS porte essentiellement sur le contenu, car les requêtes utilisées sont les mêmes que celles utilisées dans la tâche CO avec les mêmes jugements de pertinence, d'où la structure ne change pas les opinions des utilisateurs quant à la pertinence des éléments XML, en revanche cette tâche a pour objectif de mesurer l'impact de la prise en compte de la structure dans les réponses du SRIS.

Avec les requêtes de type CO, on pourra mesurer l'efficacité du SRIS sur le contenu seulement c'est-à-dire, comme un processus de RI traditionnelle. Dans INEX'2005, les requêtes de type COS contiennent les mêmes mots-clés des requêtes de type CO, autrement dit les deux types de requêtes (CO et COS) cherchent la même information sur le contenu. Avec les requêtes COS, l'objectif des évaluations est d'évaluer à quel point l'information structurelle aide la RI structurée [128]. Dans notre modèle, l'apport de notre méthode de recherche par la structure est significatif comme illustrent les tableaux 4.3 et 4.4.
Les conditions structurelles, améliorent généralement les résultats obtenus, le pourcentage d'amélioration atteint 33.3866% pour la mesure $nxCG[50]$ avec $\alpha = 0.6$ et la quantification f_{strict} avec et sans chevauchement. Ce qui montre que plusieurs nœuds très pertinents non retrouvés lors de la recherche orientée contenu, peuvent être trouvés grâce à la prise en compte de la structure. Les scores finaux sont calculés grâce à la formule de combinaison linéaire.
La contribution du processus du recherche par la structure est significatif, en fait, nous obtenons une amélioration élevée pour $\alpha = 0.6$ [8]. On peut conclure que l'apport de la recherche par la structure est de 40% dans la RI structurée.
Pour la suite de nos évaluations, nous allons garder $\alpha = 0.6$.

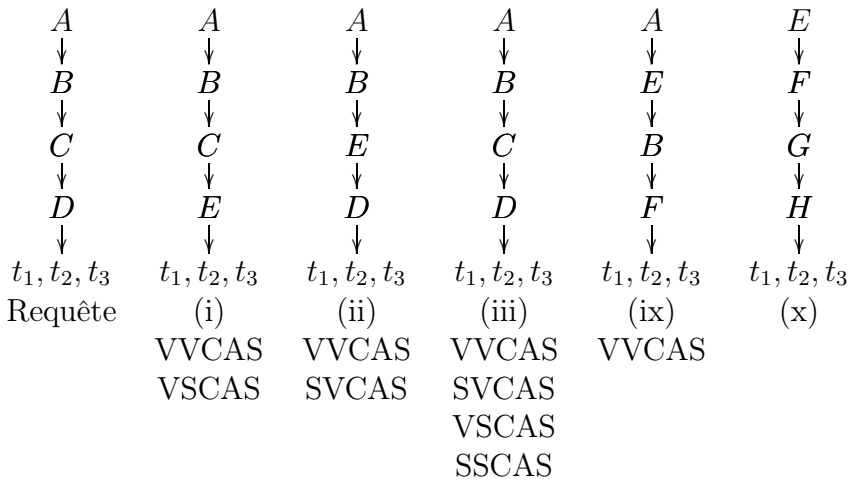
4.3.3 Requêtes de type XYCAS

Les tableaux 4.5, 4.6, 4.7 et 4.8, montrent les résultats obtenus pour les requêtes orientées contenu et structure, les résultats obtenus pour la tâche VVCAS illustré par le tableau 4.5 prouve l'efficacité de notre système de recherche. Dans cette tâche les nœuds de soutien et les nœuds cibles sont comparés d'une manière vague. Cette tâche est la plus appropriée à notre système car ces deux contraintes de recherche sont traitées d'une façon vague.
Par exemple, si le besoin en information est illustré par la requête illustrée par la figure 4.1 :

- tâche SSCAS : le résultat (iii) est le seul résultat qui pourrait être pertinent, car le nœud cible (D) et les nœuds de soutien ($A \rightarrow B \rightarrow C$) sont respectés. Ce

résultat est potentiellement pertinent à toutes les tâches XYCAS,

- tâche VSCAS : le résultat (i) pourrait être pertinent bien que le nœud cible n'est pas celui indiqué par la représentation du besoin en information, car les nœuds de soutien sont respectés,
- tâche SVCAS : le résultat (ii) pourrait être pertinent bien que les nœuds de soutien ne sont pas respectés, car on retrouve l'information recherchée dans le nœud cible recherché,
- tâche VVCAS : dès qu'un nœud parmi les nœuds de soutien et le nœud cible apparaît, le résultat pourrait être pertinent. Le résultat (x) ne contient aucun nœud de la série, par conséquent, il n'est pertinent à aucune tâche XYCAS.



► Figure 4.1: Exemples de fragments évalués dans les tâches XYCAS

Si la recherche se fait d'une façon stricte sur les nœuds de soutien, il est indispensable de considérer les relations structurelles d'une façon stricte c'est-à-dire sans pondération des chemins. Si par ailleurs la recherche de nœuds cibles est stricte, il faudra écarter tous les résultats dont le nom de la balise ne correspond pas à l'objectif de la requête. Nous avons évalué notre approche selon ces deux considérations, les résultats obtenus sont illustrés par le tableau 4.6. Nous restons tout de même au dessous des meilleurs résultats d'INEX. La performance dans ce type de mesures est très liée au traitement du contenu.

4.3.3.1 Comportement de notre système pour la quantification stricte

Pour la quantification stricte (qui évalue à quel point le SRI ne retourne que les éléments spécifiques et exhaustifs à une requête), la tâche VVCAS en est la mesure la plus

Quantification	*Notre approche*	*Max*	*Moy*	*Min*
f_{strict}	0.0990	0.0614	0.0286	0.0000
f_{gen}	0.0794	0.1283	0.0526	0.0000
$f_{genLifted}$	0.0275	0.0519	0.0197	0.0000

► Tableau 4.5: Résultats comparatifs avec les participants officiels d'INEX'2005, tâche VVCAS

Quantification	*Notre approche*	*Max*	*Moy*	*Min*
f_{strict}	0.0329	0.1334	0.0662	0.0000
f_{gen}	0.0390	0.2297	0.1073	0.0000
$f_{genLifted}$	0.0384	0.2221	0.1013	0.0000

► Tableau 4.6: Résultats comparatifs avec les participants officiels d'INEX'2005, tâche SSCAS

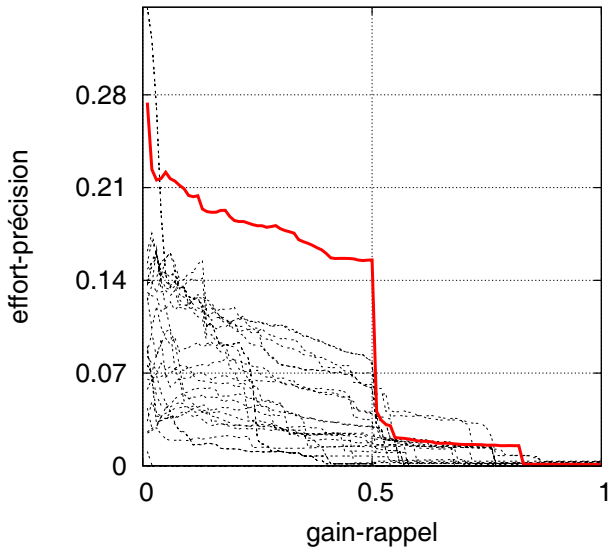
Quantification	*Notre approche*	*Max*	*Moy*	*Min*
f_{strict}	0.0400	0.0765	0.0297	0.0000
f_{gen}	0.0823	0.1497	0.0561	0.0000
$f_{genLifted}$	0.0300	0.0553	0.0198	0.0000

► Tableau 4.7: Résultats comparatifs avec les participants officiels d'INEX'2005, tâche VSCAS

Quantification	*Notre approche*	*Max*	*Moy*	*Min*
f_{strict}	0.1075	0.1762	0.0670	0.0000
f_{gen}	0.0378	0.1860	0.0732	0.0000
$f_{genLifted}$	0.0373	0.1768	0.0685	0.0000

► Tableau 4.8: Résultats comparatifs avec les participants officiels d'INEX'2005, tâche SVCAS

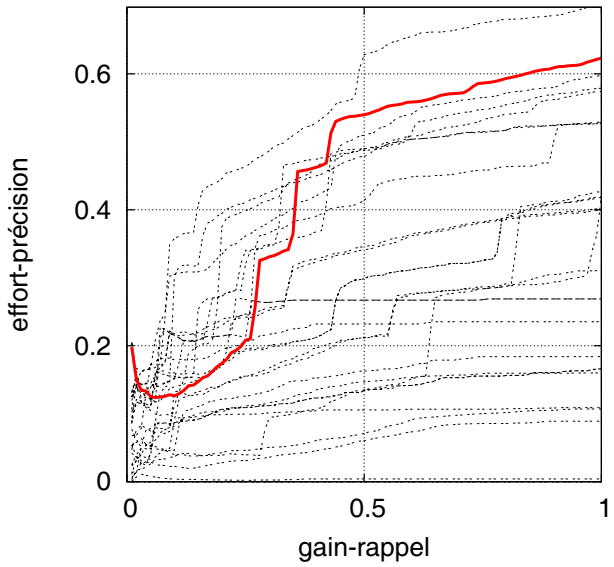
adaptée, les nœuds cibles et les nœuds de soutien de la requête sont comparés avec les fragments des documents d'une manière vague. La figure 4.2 montre que notre système est nettement plus performant que tous les participants d'INEX. De même la figure 4.3 prouve que notre système est très compétitif aux outsiders d'INEX. Cette



► Figure 4.2: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VVCAS, quantification f_{strict}

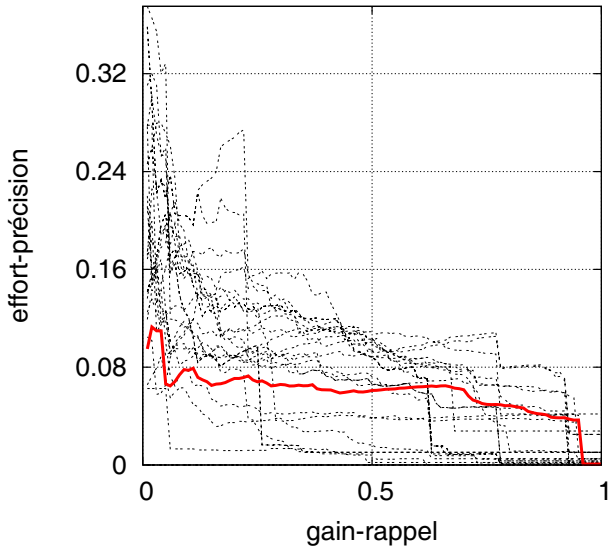
expérimentation montre à quel point la prise en compte de la structure est important pour la RIS. Cette tâche est la seule parmi toutes les tâches d'INEX qui donne de l'importance au contenu et la structure à parts presque égales. Cette tâche est par ailleurs la plus complexe, les résultats de tous les participants d'INEX sont très faibles comparaison faite avec ce qu'ils obtiennent dans les autres tâches. Nous admettons par le même que la quantification f_{strict} est la plus appropriée à cette tâche car lorsque l'utilisateur compose des requêtes structurées, il attend des résultats très spécifiques, car l'exhaustivité des fragments est nuisible au respect des conditions structurelles : si on recherche des éléments exhaustifs, on devrait amplifier les structures pertinentes par d'autres pour que l'information devienne exhaustive mais en revanche, on perd en précision structurelle.
Il est clair que par définition, la tâche SSCAS favorise la recherche par la structure, elle ne fournit par conséquent aucune pertinence graduée (les scores orientées structure sont 0 ou 1), les nœuds qui ne répondent pas à la structure sont écartés de la réponse qui sera retournée à l'utilisateur. Il n'en reste qu'ordonner les éléments restant selon leurs scores orientés contenu : La tâche SSCAS est quasiment la même que la tâche CO (CO avec filtrage a priori sur la structure).

Pour la quantification stricte de la tâche SSCAS, qui considère que les nœuds cibles et les nœuds de soutien de la requête doivent être comparés avec les fragments des

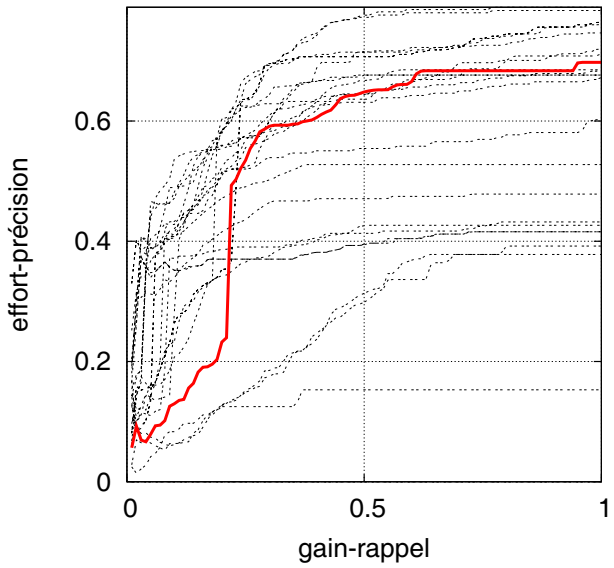


► Figure 4.3: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VVCAS, quantification f_{strict}

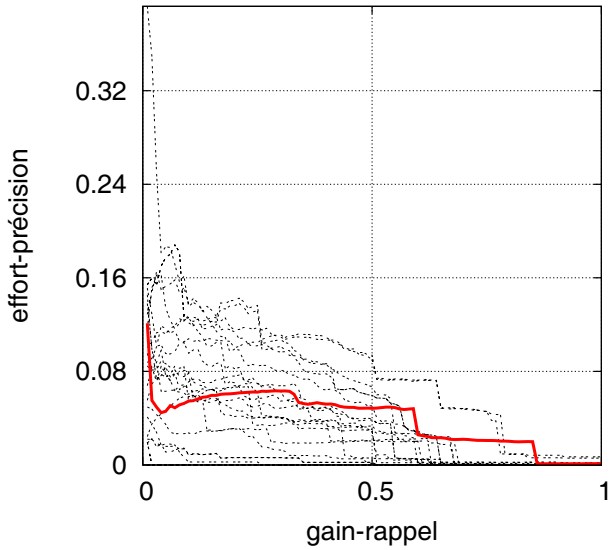
documents d'une manière stricte, la réponse d'un SRIS peut se ramener à celle d'un langage de requête exact tel que XQuery. La suite des expérimentations concernent la tâche VVCAS. Les tâches VSCAS et SVCAS suggèrent moins de flexibilité qu'en VVCAS et un peu plus qu'en SSCAS. Nous présumons que la tâche SVCAS est tout de même plus flexible car seul le nœud cible est comparé d'une façon stricte. Les figures 4.6 et 4.7 montrent les résultats obtenus selon la quantification f_{strict} de la tâche VSCAS.
Nous constatons que notre modèle est plus adapté à la tâche VVCAS qu'à cette tâche, et moins encore pour la tâche SSCAS : plus la recherche est flexible, plus notre modèle s'adapte aux besoins de l'utilisateur. Les figures 4.8 et 4.9 montrent les résultats obtenus pour la tâche SVCAS. Globalement, les résultats relatifs sont nettement meilleurs que ceux obtenus pour la tâche VSCAS, mais moins bons qu'en VVCAS. Ceci s'explique par le fait qu'en VSCAS, l'utilisateur impose une structure stricte et ne tolère la flexibilité qu'au niveau du dernier élément de sa requête. En SVCAS, c'est le traitement inverse qui devra être pris en compte. Il y a alors plus de flexibilité car généralement, le nombre de nœuds de soutien est plus grand que le nombre de nœuds cibles (1 seul nœud cible contre plusieurs nœuds de soutien).
Nous pouvons constater alors que pour la quantification f_{strict}, plus la tâche suggère des contraintes structurelle flexibles, plus notre modèle se distingue des participants officiels d'INEX.



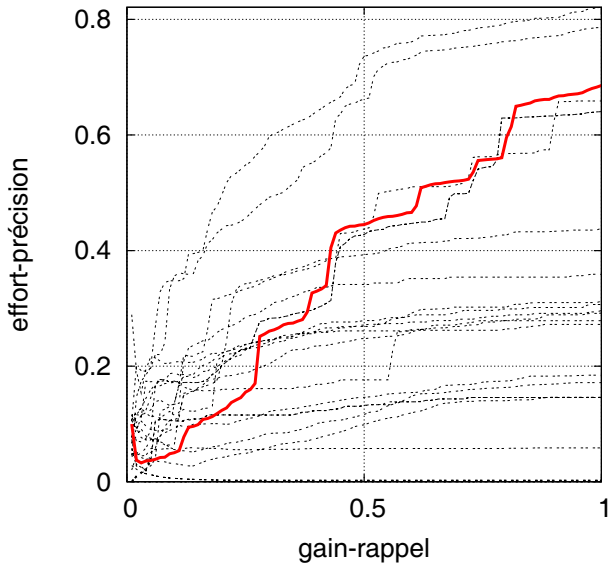
► Figure 4.4: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SSCAS, quantification f_{strict}



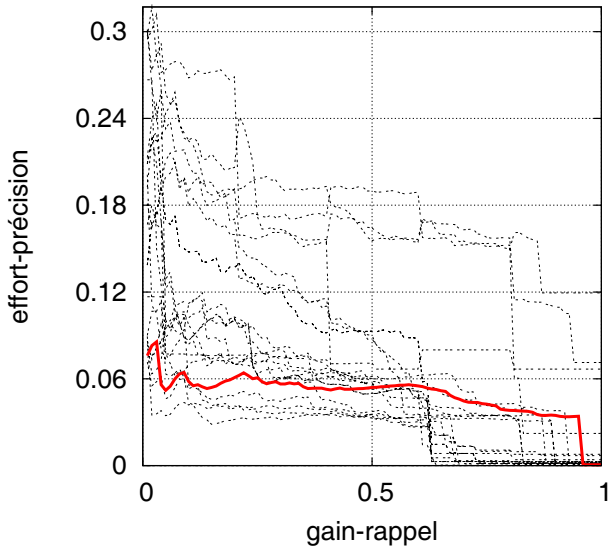
► Figure 4.5: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche SSCAS, quantification f_{strict}



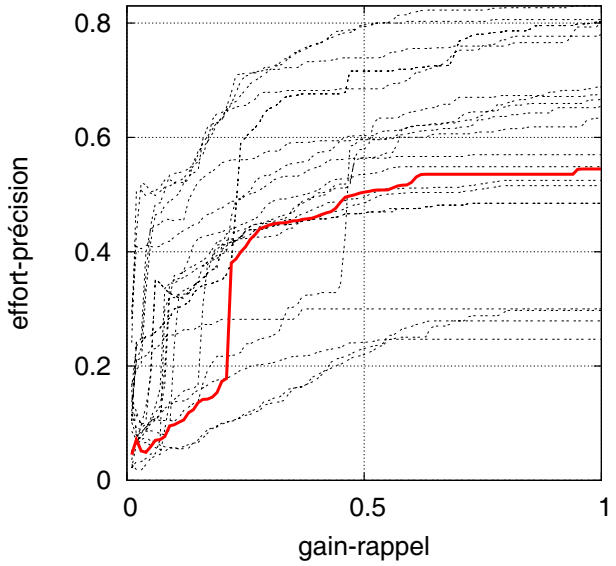
► Figure 4.6: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VSCAS, quantification f_{strict}



► Figure 4.7: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VSCAS, quantification f_{strict}



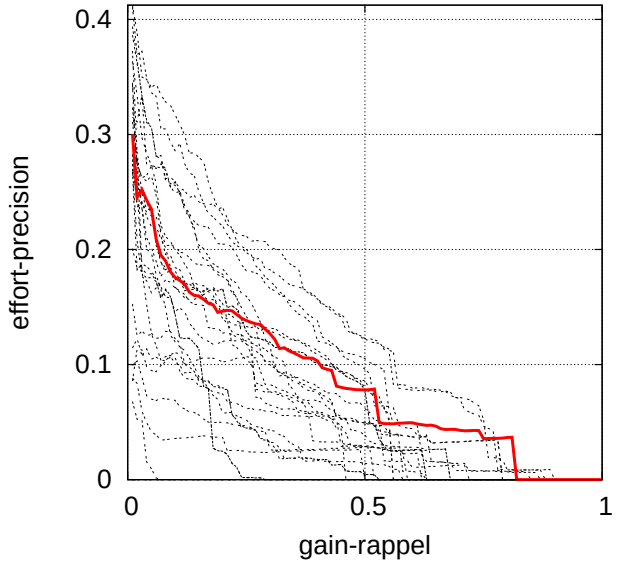
► Figure 4.8: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SVCAS, quantification f_{strict}



► Figure 4.9: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche SVCAS, quantification f_{strict}

4.3.3.2 Comportement de notre système pour la quantification généralisée

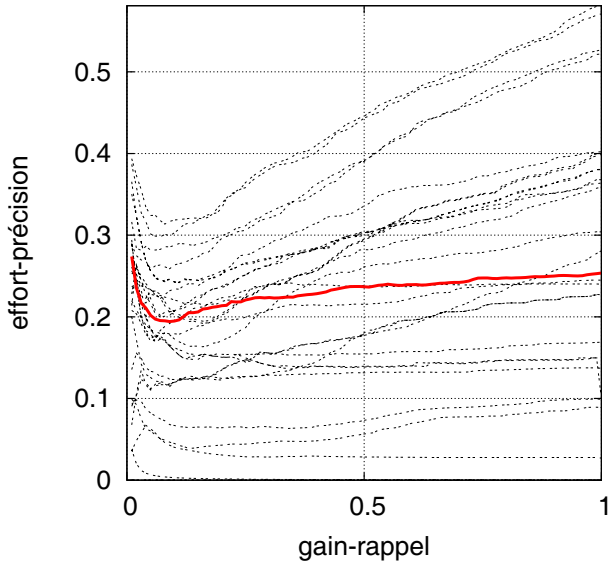
La fonction f_{gen} considère que la spécificité et l'exhaustivité d'un élément XML co-contribuent à sa pertinence. Les éléments peuvent donc être pertinents s'ils sont peut spécifiques mais très exhaustifs (ou inversement). Il'y a donc plus de tolérance en structure puisque les éléments très exhaustifs ne possèdent généralement pas la structure de la requête.
Les résultats de la tâche VVCAS selon la quantification f_{gen} illustrés par les figures 4.10 et 4.11 montrent que notre modèle, grâce à sa façon de traitement de la structure, figure parmi les meilleurs systèmes participants à INEX. Pour la tâche SSCAS, les



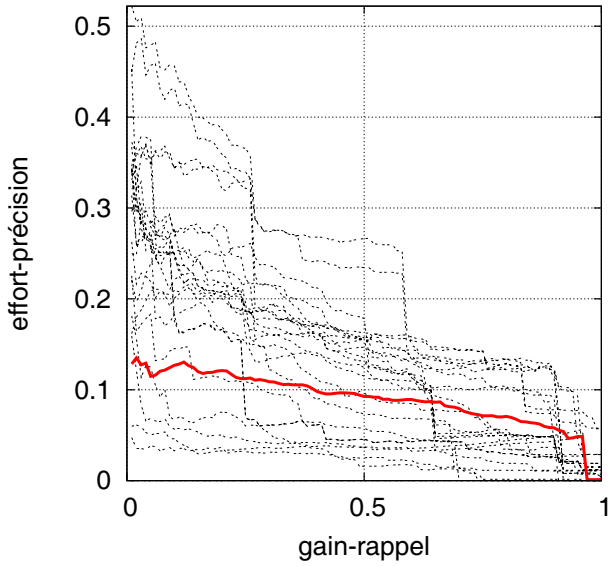
► Figure 4.10: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VVCAS, quantification f_{gen}

résultats sont très faibles car la tâche et la fonction de quantification sont orientées contenu : la structure n'est ni au niveau de la tâche, ni au niveau de l'évaluation.

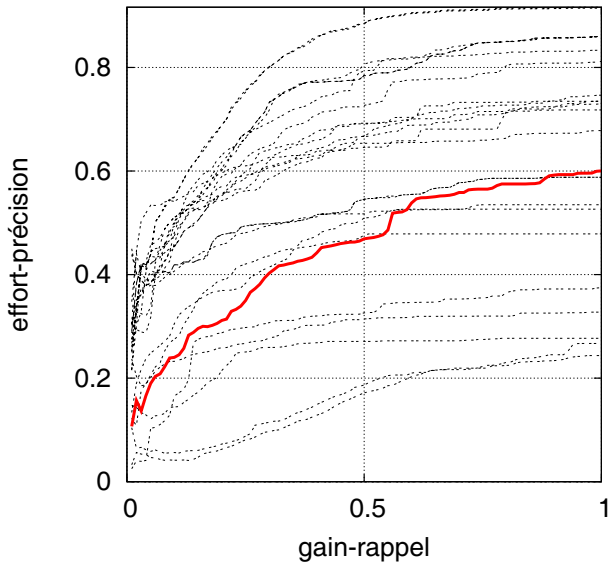
Pour la tâche VSCAS, nous enregistrons presque les mêmes résultats que pour la fonction de quantification f_{strict} (relativement aux participants officiels d'INEX). Pour la tâche SVCAS, les résultats illustrés par les figures 4.16 et 4.17 montrent que notre approche n'est pas adaptée à cette fonction de quantification, car l'évaluation privilégie les éléments qui répondent aux nœuds cibles d'une façon stricte, et en même temps elle mesure la performance du système en fonction de l'exhaustivité de ses réponses en plus de la spécificité. Les éléments retournés par l'alignement des nœuds cibles auront alors une valeur d'exhaustivité faible et une valeur de spécificité plus au moins élevée.
Le traitement de ces tâches en fonction de la quantification devrait prendre en considération les aspects d'exhaustivité et de spécificité, c'est-à-dire, tout l'art est d'estimer le



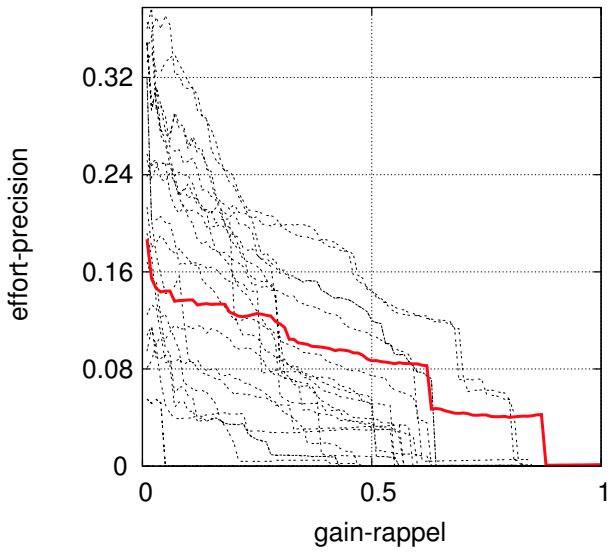
► Figure 4.11: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VVCAS, quantification f_{gen}



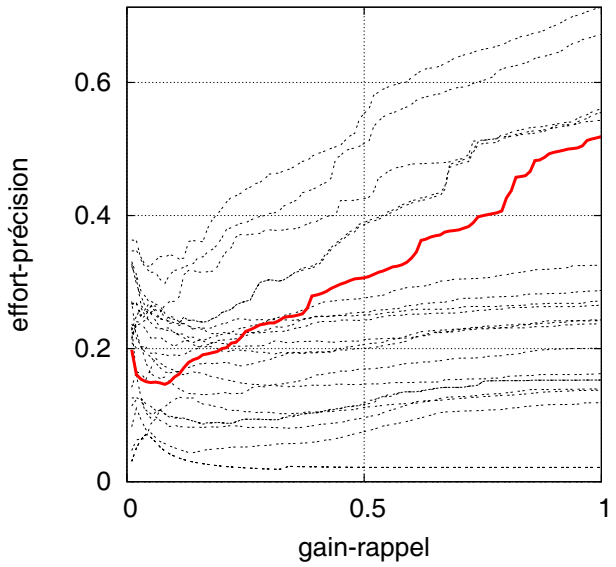
► Figure 4.12: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SSCAS, quantification f_{gen}



► Figure 4.13: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche SSCAS, quantification f_{gen}



► Figure 4.14: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VSCAS, quantification f_{gen}

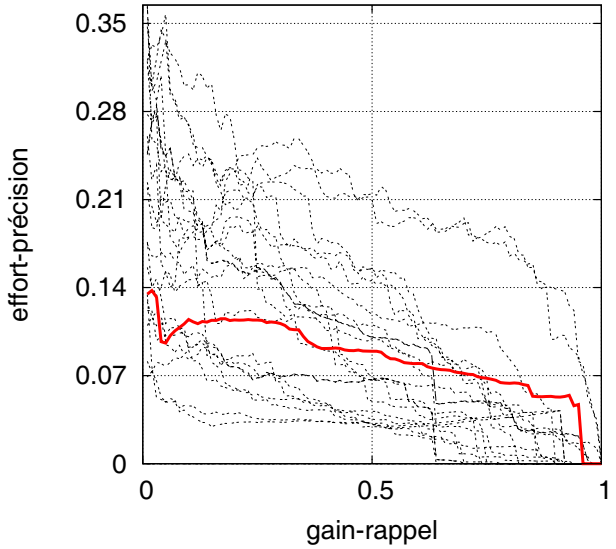


► Figure 4.15: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VSCAS, quantification f_{gen}

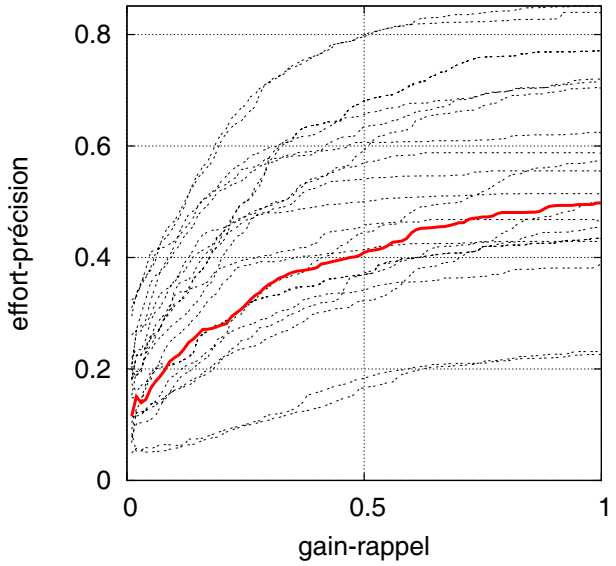
poids d'un élément XML en retournant le meilleur compromis entre les critères orientés idf et les critères orientés ief.

4.3.3.3 Comportement de notre système pour la quantification généralisée étendue

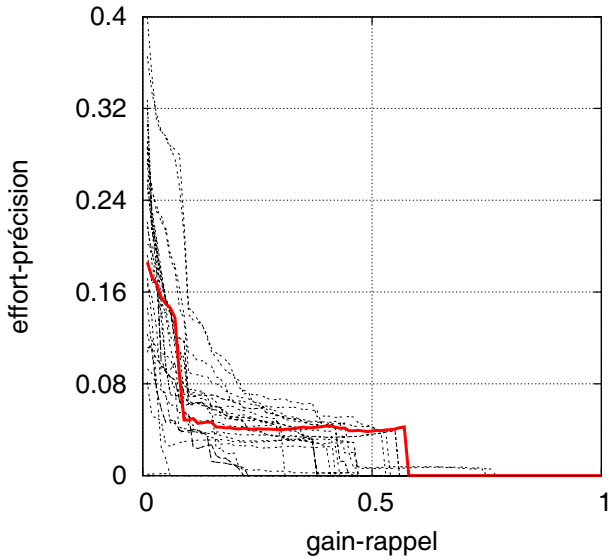
Tous les participants d'INEX retrouvent des valeurs presque nulles d'effort-précision à partir de toutes les valeurs de gain-rappel supérieures à 0.5. En effet, la valeur de la fonction $f_{genLifted}$ sont orientées spécificité, les nœuds non exhaustifs reçoivent tout de même une valeur d'évaluation non nulle.
Nous avons constaté que pour la tâche VVCAS, les valeurs d'effort-précision sont stables en fonction de toutes les valeurs de gain-rappel supérieures à 0.1 et inférieures à 0.6, contrairement à la plupart des autres participants d'INEX (voir figure 4.18). Pour la tâche SSCAS, les résultats sont faibles comparaison faite avec les autres participants d'INEX, car la fonction $f_{genLifted}$ est totalement orientée contenu, il est à noter que tous les nœuds qui répondent au contenu de la requête sont probablement les plus pertinents. Dans cette tâche, il est indispensable de prendre en considération le contenu.



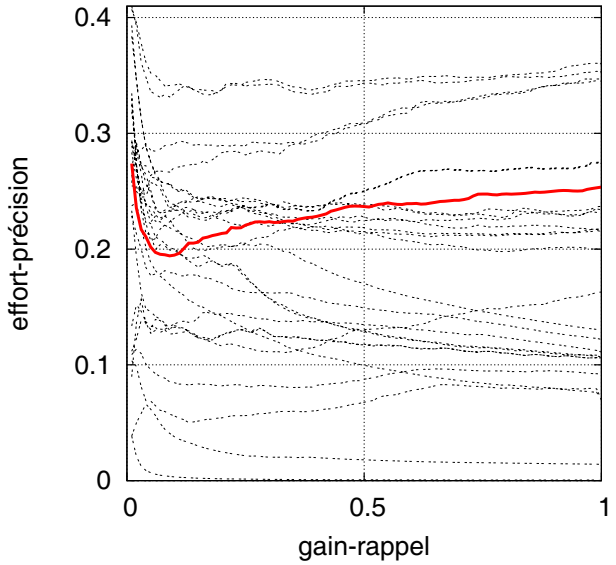
► Figure 4.16: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SVCAS, quantification f_{gen}



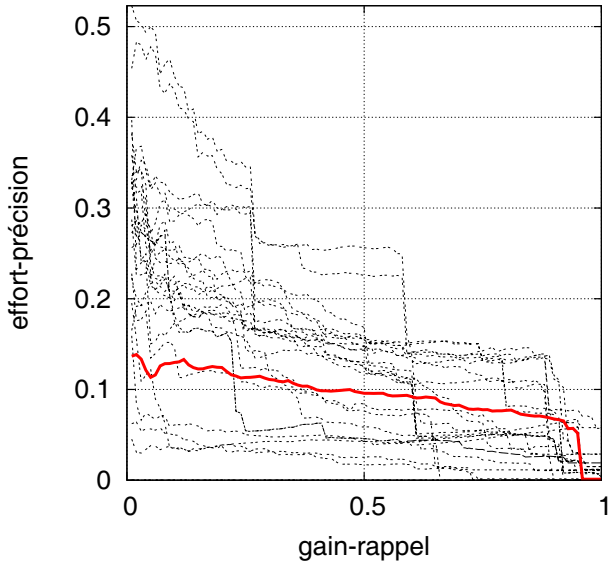
► Figure 4.17: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VVCAS, quantification f_{gen}



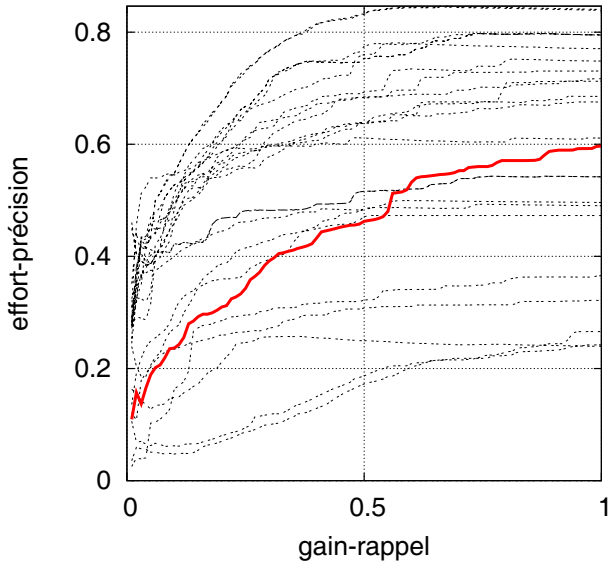
► Figure 4.18: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VVCAS, quantification $f_{genLifted}$



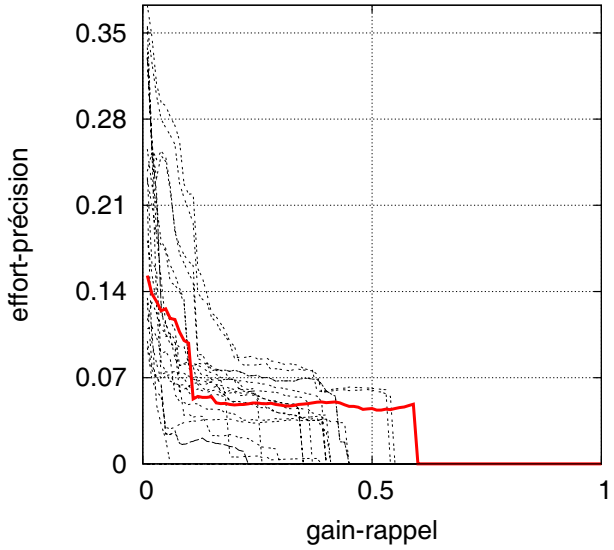
► Figure 4.19: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VVCAS, quantification $f_{genLifted}$



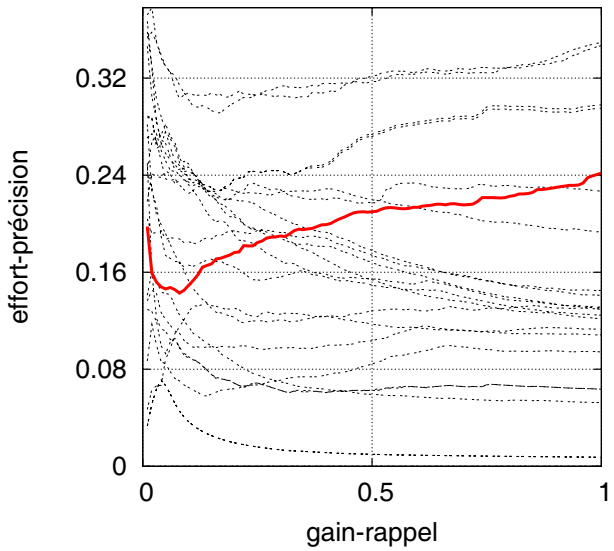
► Figure 4.20: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SSCAS, quantification $f_{genLifted}$



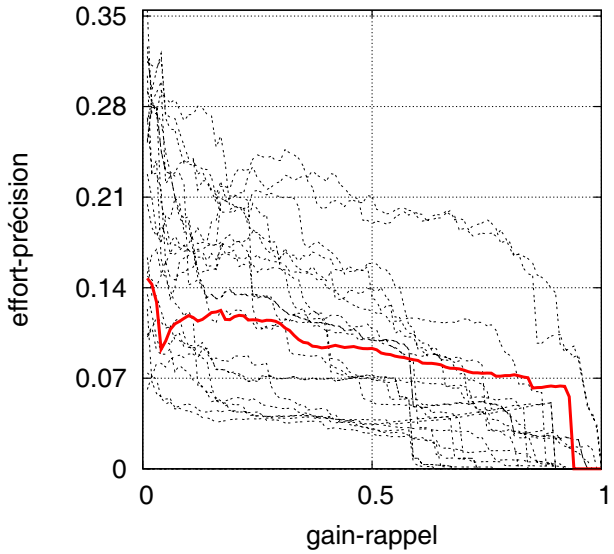
► Figure 4.21: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche SSCAS, quantification $f_{genLifted}$



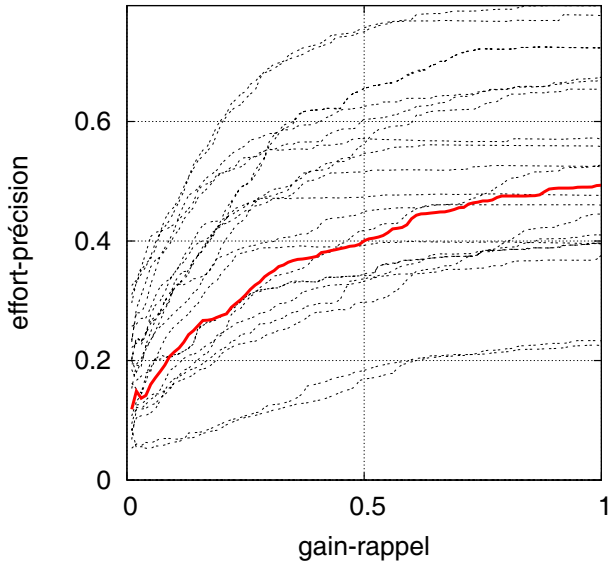
► Figure 4.22: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VSCAS, quantification $f_{genLifted}$



► Figure 4.23: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VSCAS, quantification $f_{genLifted}$



► Figure 4.24: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche SVCAS, quantification $f_{genLifted}$



► Figure 4.25: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche SVCAS, quantification $f_{genLifted}$

4.4 Expérimentations de la distribution des scores

La combinaison des scores orientés contenu et structure n'est pas toujours adaptée à leurs domaines de définition. Au lieu d'estimer le score final d'un élément à partir des scores tels qu'ils se présentent, nous avons développé une technique qui consiste à estimer le score d'un élément à partir de sa probabilité de pertinence qui est conditionnée par les distributions des scores orientés contenu et ceux orientés structure. Les premières expériences ont été réalisées pour montrer la distribution des scores du contenu et de la structure (voir figures 3.32 et 3.33). Pour chacune des requêtes, nous considérons les 5000 premiers nœuds classés par chaque orientation de recherche (contenu ou structure).
Dans la section 3.10.2 nous avons réalisé des expérimentations qui consistent à estimer la distribution des scores de chaque orientation de recherche. Comme nous avons évoqué, il est indispensable de pouvoir estimer les paramètres relatifs à la distribution des nœuds pertinents en fonction de celle de tous les nœuds. Les figures 3.34 et 3.35 montrent que les relations r_λ et r_μ sont affines, nous utilisons les fonctions suivantes : r_λ, r_μ et r_σ définies par $\mu_C^R = r_\lambda(\mu_C)$, $\mu_S^R = r_\mu(\mu_S)$ et $\sigma_S^R = r_\sigma(\sigma_S)$. Les détails de ces fonctions sont :

$$\begin{aligned} r_\lambda(\mu_C) &= 8 \times \mu_C - 400 \\ r_\mu(\mu_S) &= 1.3 \times \mu_S - 0.5 \end{aligned}$$

Les figures 3.32 et 3.33 montrent les résultats obtenus en utilisant la combinaison des scores orientés contenu et structure basée sur les distributions de scores.
Par le calcul du score en utilisant les distributions des scores orientés contenu et structure, notre système reste le plus performant dans la tâche VVCAS et la mesure MAep, le tableau 4.9 présente les résultats obtenus. Cette technique à l'avantage de donner de bonnes estimations de la combinaison qu'il faut considérer sans que les domaines de définition des scores de chaque orientation n'en a de l'effet.

4.5 Résultats obtenus en utilisant le nombre de fils

En utilisant le nombre de fils comme un paramètre pour la propagation de texte, les scores du contenu changent, les scores résultant de la recherche orientée structure restent inchangés. Le but de cette expérimentation est alors de mesurer l'impact de ce critère dans la recherche orientée contenu.
Pour la tâche SSCAS, nous avons constaté une amélioration de 100% au niveau de fonction f_{strict}. Dans cette tâche, puisque les contraintes structurelles sont strictes, les systèmes se départagent selon leur façon d'appréhender le contenu.
Par ailleurs, nous avons constaté une amélioration de 2.6% au niveau de la fonction $MAep$ (de 0.099 à 0.1016) de la tâche VVCAS ; et une amélioration de 50% dans la tâche VSCAS (de 0.04 à 0.06).

Quantification	nxCG[10]			
	Notre approche	*Max*	*Moy*	*Min*
f_{strict}	0.1778	0.1889	0.0757	0.0000
f_{gen}	0.2643	0.4289	0.2137	0.0000
$f_{genLifted}$	0.2716	0.4459	0.2323	0.0000
	nxCG[25]			
	Notre approche	*Max*	*Moy*	*Min*
f_{strict}	0.1533	0.1622	0.0785	0.0000
f_{gen}	0.2348	0.3653	0.1994	0.0000
$f_{genLifted}$	0.2378	0.3902	0.2153	0.0000
	nxCG[50]			
	Notre approche	*Max*	*Moy*	*Min*
f_{strict}	0.1336	0.1640	0.0793	0.0000
f_{gen}	0.2093	0.3383	0.1798	0.0000
$f_{genLifted}$	0.2144	0.3645	0.1956	0.0000
	MAep			
	Notre approche	*Max*	*Moy*	*Min*
f_{strict}	0.0994	0.0614	0.0286	0.0000
f_{gen}	0.0794	0.1283	0.0526	0.0000
$f_{genLifted}$	0.0276	0.0519	0.0197	0.0000

▶ Tableau 4.9: Résultats comparatifs avec les participants d'INEX'2005, tâche VVCAS

Le tableau 4.10 illustre les résultats obtenus dans la tâche SSCAS en tenant compte du nombre de fils dans la procédure de propagation de texte.

Quantification	$nxCG[10]$	$nxCG[25]$	$nxCG[50]$	$MAep$
f_{strict}	0.0100	0.0800	0.0550	0.0527
f_{gen}	0.0868	0.0653	0.0537	0.0453
$f_{genLifted}$	0.0921	0.0550	0.1013	0.0440

▶ Tableau 4.10: Résultats obtenus pour la tâche SSCAS en utilisant le nombre de fils

4.6 Résultats obtenus en utilisant l'indexation des balises

Pour la tâche VVCAS dans la quantification stricte nous avons eu une amélioration de la valeur de MAep qui est passée de 0.99 à 0.1016 et pour la même quantification pour la tâche VSCAS nous avons eu une amélioration de $19,7\%$. La principale amélioration a été enregistrée pour la tâche SSCAS, où les performances ont presque doublé. Le tableau 4.11 illustre les résultats obtenus pour la tâche SSCAS en utilisant l'indexation des balises.

Quantification	$nxCG[10]$	$nxCG[10]$	$nxCG[10]$	$MAep$
f_{strict}	0.1000	0.0800	0.055	0.0528
f_{gen}	0.868	0.0653	0.0537	0.0453
$f_{genLifted}$	0.0921	0.0680	0.0550	0.0442

▶ Tableau 4.11: Résultats obtenus pour la tâche SSCAS avec indexation des balises

4.7 Expérimentation de la méthode de radicalisation

Nous avons effectué une série d'expérimentations sur des termes de la langue anglaise. L'expérimentation consiste à :

- définir les poids des $NGrams$, en fonction de leurs apparitions dans un grand document,
- définir une valeur T_{ress} (seuil de ressemblance) permettant de retrouver les classes d'équivalence de termes d'indexation, cette phase est similaire à l'apprentissage.

La meilleure valeur du seuil retenue à travers nos expérimentations est $T_{ress} = 0.15$. Les figures 4.26 et 4.27 illustrent la courbe gain-rappel/effort-précision de la tâche VVCAS selon la quantification stricte. Les résultats montrent que notre méthode de radicalisation est très compétitive à la méthode de Porter, en effet, notre système reste le plus performant pour la mesure $MAep$. Concernant la mesure $nxCG$, nous avons constaté que notre méthode de radicalisation améliore les résultats et classe notre système parmi les premiers pour toutes les valeurs de gain-rappel entre 0 et 0.5. Pour ces valeurs, la radicalisation en tenant compte de la position du $NGram$ dans un mot est bénéfique, en effet, la courbe gain-rappel/effort-précision est largement au dessus de celle relative à l'utilisation des $NGrams$ sans tenir compte de leurs positions.

4.7.1 Expérimentation en utilisant $2Grams$

Nos expérimentations ont été menées sur plusieurs valeurs de $NGrams$, nous avons eu les meilleurs résultats avec la composition d'un mot en $2Grams$. Les figures 4.26 et 4.27 montrent les résultats obtenus par les deux approches (avec et sans position).

Quantification	$nxCG[10]$	$nxCG[10]$	$nxCG[10]$	$MAep$
f_{strict}	0.1667	0.1267	0.1301	0.0885
f_{gen}	0.1960	0.1507	0.1340	0.0323
$f_{genLifted}$	0.2343	0.1958	0.1785	0.0127

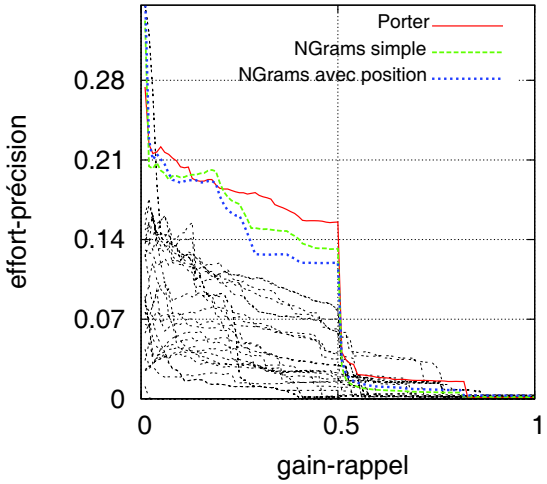
▶ Tableau 4.12: Résultats obtenus avec un seuil=0.15, sans tenir compte du facteur de la position

Quantification	$nxCG[10]$	$nxCG[10]$	$nxCG[10]$	$MAep$
f_{strict}	0.1778	0.1400	0.1440	0.0843
f_{gen}	0.1960	0.1627	0.1432	0.0256
$f_{genLifted}$	0.2377	0.2067	0.1870	0.0115

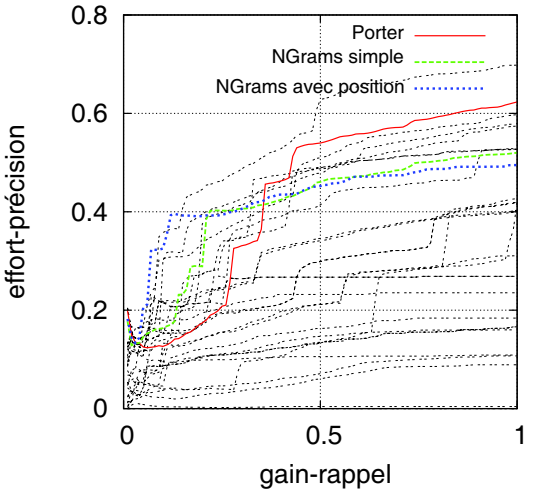
▶ Tableau 4.13: Les précisions obtenus avec un seuil=0.15, en tenant compte du facteur de la position

4.7.2 Expérimentation en utilisant $3Grams$

Nous avons réalisé quelques expérimentations en composant chaque mot de la collection en 3 lettres ($3Grams$). Le tableau 4.14 illustre nos meilleurs résultats en utilisant $3Grams$. Les valeurs montrées dans ce tableau prouve que l'utilisation des $2Grams$ pour calculer la similarité entre deux termes est beaucoup plus efficace qu'en utilisant $3Grams$. Contrairement aux résultats obtenus en utilisant $2Grams$, avec $3Grams$ l'injection du facteur de la position a baissé la performance du système.



► Figure 4.26: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $MAep$, tâche VVCAS, quantification f_{strict}, en utilisant $2Grams$ avec positions, sans positions et Porter



► Figure 4.27: Comparaison avec les participants officiels d'INEX'2005 selon la mesure $nxCG$, tâche VVCAS, quantification f_{strict}, en utilisant $2Grams$ avec positions, sans positions et Porter

Quantification	$nxCG[10]$	
	sans position	avec position
f_{strict}	0.1444	0.111
f_{gen}	0.1678	0.058
$f_{genLifted}$	0.1961	0.0804
	$nxCG[25]$	
	sans position	avec position
f_{strict}	0.1356	0.1378
f_{gen}	0.1506	0.0621
$f_{genLifted}$	0.1858	0.0809
	$nxCG[50]$	
	sans position	avec position
f_{strict}	0.1345	0.1314
f_{gen}	0.1408	0.0452
$f_{genLifted}$	0.1775	0.057
	$MAep$	
	sans position	avec position
f_{strict}	0.0478	0.0292
f_{gen}	0.0286	0.0032
$f_{genLifted}$	0.0118	0.0015

▶ Tableau 4.14: Résultats obtenus en utilisant $3Grams$

4.8 Conclusion

Nous avons testé dans ce chapitre nos approches présentées dans le chapitre 3. Nos expérimentations ont été réalisées sur les deux collections issues d'INEX en 2004 et 2005. A travers, nous expérimentations sur les différentes pondérations possibles pour la RIS ($tf \times idf$ et $tf \times ief$), nous avons constaté qu'avec la pondération $tf \times ief$ notre système était plus efficace. Nous avons obtenu une efficacité plus importante dans les tâches où la comparaison se fait d'une manière flexible, en effet notre système se base sur la comparaison d'arbres étendus, qui s'avère adaptée au besoin de la tâche. Plusieurs paramètres de traitement du contenu ont été expérimentés, nous avons constaté qu'ils ont un effet positif sur la recherche orientée contenu.
Les expérimentations concernant notre méthode de radicalisation a montré beaucoup d'efficacité comparaison faite avec l'algorithme de Porter pour la langue anglaise. Cette méthode permet de classifier les termes indépendamment de la langue, grâce à son caractère statistique.

Conclusion et perspectives

Les travaux présentés dans ce mémoire se situent dans le cadre de la recherche d'information structurée. Un système de recherche d'information structurée (SRIS) combine la structure et le contenu des documents pour répondre de la manière la plus spécifique et exhaustive possible au besoin en information de l'utilisateur. Un SRIS renvoie des parties de documents XML. Nous nous sommes intéressés dans ces travaux à proposer une solution permettant un traitement flexible de la structure.
Plus précisément notre approche repose sur plusieurs points importants :

1. L'approche concerne principalement la proposition d'une nouvelle méthode d'indexation de la structure. Pour chercher les parties pertinentes des documents XML, notre approche se base sur la comparaison d'arbres. Nous avons développé un algorithme d'aspect statistique, basé sur le principe de relaxation de requête. Cet algorithme permet en même temps d'effectuer la comparaison entre les arbres et de calculer leurs appariements. Les arbres que nous traitons sont les arbres représentatifs des documents XML et ceux des requêtes, auxquels nous ajoutons des structures additionnelles permettant ainsi de favoriser la recherche flexible de structures potentiellement pertinentes. L'algorithme est de complexité relativement très réduite et permet de produire des scores de pertinence graduée en terme de structure.

2. La comparaison d'arbres nous a permis d'explorer les propriétés des arbres tant au niveau formel qu'au niveau technique. L'intégration de ces propriétés dans la recherche par le contenu nous a été extrêmement bénéfique : elle nous a permis d'évaluer le pouvoir descriminatoire de chacune sur la perception de la pertinence utilisateur. Nous avons pris en considération, outre la profondeur de l'arbre qui est l'élément fondamental utilisé en littérature, la largeur qui représente une autre propriété presque aussi importante que la profondeur.

3. Le traitement séparé du contenu et de la structure de chaque élément XML engendre deux scores : un score pour le contenu et un score pour la structure. Leur combinaison en un score définitif permet de les ordonner selon leur pertinence potentielle. Nous avons développé deux techniques pour la combinaison des scores : une technique basée sur une combinaison linéaire et une deuxième

technique basée sur l'étude des distributions des scores. Par ailleurs, et dans des conditions réelles d'utilisation, la séparation entre le contenu et la structure permet une utilisation plus libérée du système, en effet l'utilisateur pourra orienter sa recherche vers le contenu ou la structure selon son besoin en information.

La séparation entre le contenu et la structure nous a permis d'une part de se doter de moyens indépendants pour améliorer chaque dimension de recherche, et d'autre part, ceci nous a permis d'évaluer l'apport de chaque orientation sur la recherche d'information structurée. L'expérimentation nous a permis de tirer une conclusion très importante : la structure contribue à la performance de la recherche d'information structurée presque autant que le contenu.

4. L'aproche a été expérimentée selon la norme INEX, dans des conditions de flexibilité et d'exhaustivité variées. Les résultats obtenus montrent l'efficacité de notre approche d'une part, et l'importance de la structure dans la RIS. A titre de précision, nous obtenons visiblement la meilleure performance dans la mesure stricte de la tâche VVCAS qui impose au système de se mettre dans les conditions du plus grand niveau de flexibilité. Cette tâche est par essence la plus complexe, elle impose l'installation de méthodes de recherche orientées structure, et de se dissocier des méthodes traditionnelles de recherche d'information et des méthodes d'interrogation par des requêtes sursemblales de SQL ou de XQuery.

Nous avons par ailleurs développé une méthode statistique pour la radicalisation des termes. Contrairement à la méthode de Porter, qui applique des règles de transformation adaptatives pour la construction du radical d'un terme, notre méthode est inspirée de la mesure $tf \times idf$, elle permet de calculer un degré de similarité entre deux termes, sans prendre en compte leurs aspects morphologiques des transformations qui peuvent être appliquées sur un mot et qui dépendent du langage utilisé. Au lieu de construire un radical, nous construisons des classes de termes ayant le même radical. Cette méthode a également été expérimentée et nous avons constaté qu'elle est très compétitive à la méthode de radicalisation de Porter. Grâce à son aspect statistique, elle pourra être appliquée à d'autres langages.
Nous proposons dans ce qui suit quelques perspectives à ces travaux, pour améliorer notre approche de recherche d'information dans les documents XML. A très cours terme, nous envisageons de mettre en route notre système sur les nouvelles collections d'INEX, notamment la base Wikipédia. A moyen terme, nous envisageons d'étendre les mesures de pondération des termes dans les éléments XML en combinant les mesures $tf \times idf$ et $tf \times ief$. Chaque mesure est plus performante dans un contexte particulier et il serait intéressant de tirer profit des avantages de chacune dans une mesure unifiée. D'autre part, nous envisageons d'intégrer la longueur des éléments (nombre de termes) puisqu'en RI traditionnelle, la longueur des documents est un critère indispensable pour la pondération des termes.
Une autre piste de recherche pourra être entamée, dont le contexte est adéquat à notre approche : c'est la tâche qui consiste à interroger des corpus de documents hétérogènes,

c'est-à-dire des documents ayant des DTD différents ; comme notre approche est indépendante de la description des documents traités, elle pourra être employée pour interroger de tels corpus. Dans la même perspective, nous envisageons de tester notre approche sur des bases multimédia dont la description est en XML augmentée d'objets images : INEX fournit de telles bases.

A long terme, nous envisageons d'améliorer notre approche par d'autres extensions, en particulier la reformulation automatique de la requête [47]. Cette piste pourra être déployée au profit de la recherche par la structure. Partant du principe que la recherche pourra être réalisée sur la structure indépendamment du contenu, il serait envisageable de proposer des méthodes de reformulation de la structure de la requête et de l'améliorer en utilisant les techniques habituelles de reformulation automatique de la requête par le contenu.

Nous prévoyons dans la même lancée d'intégrer des sources de données hétérogènes ayant des formats proches de XML, en l'occurrence, nous pouvons citer un concept fortement recommandé pour la gestion des connaissances : les ontologies [31] [32].

Enfin, nous souhaiterions développer une interface pour l'interrogation et pour la présentation des résultats à l'utilisateur. L'interface d'interrogation devrait guider l'utilisateur dans la formulation de la requête, si possible de façon dynamique, en lui présentant par exemple les éléments de structure sur lesquels il peut interroger le système. La présentation des résultats soulève un grand nombre de questions : les résultats doivent-ils être présentés au sein du document ? ou bien Doit-on regrouper les resultat par document ou bien présenter une simple liste triée de résultats ? Pour répondre au mieux au besoin de l'utilisateur, ces éléments pourrraient être regroupés, et les résultats seraient alors présentés à l'utilisateur sous forme d'une liste de groupes d'éléments.

Annexe A

La famille XML

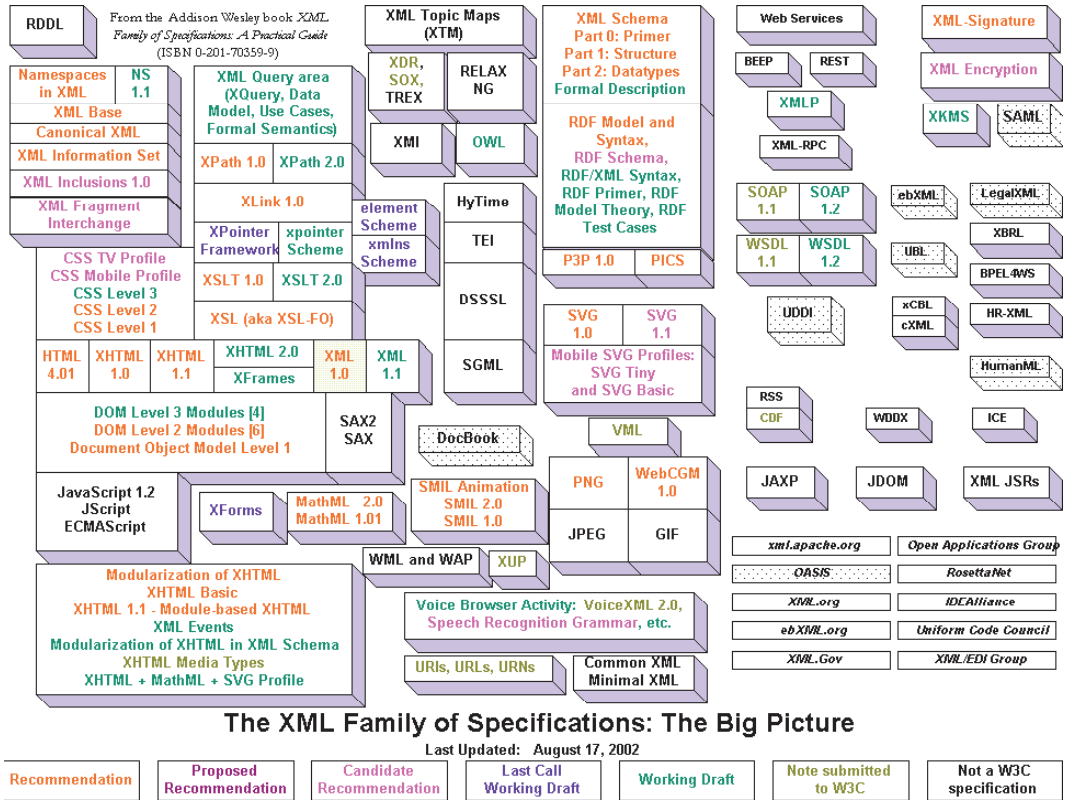
Le langage XML [21][81] [132] constitue le centre de la famille. Autour de sa spécification, de nombreuses technologies proposées par le W3C sont apparues. La figure A.1 présente les plus importants.

A.1 Les namespaces

Les espaces de noms ont été introduits dans un but bien précis : assurer l'unicité des éléments XML. En effet, le langage XML permet à l'utilisateur de définir ses propres balises. Aussi, il est fort probable que plusieurs utilisateurs définissent leurs propres significations d'un élément portant le même nom. L'usage des espaces de nom empêche cette collision en associant chaque élément à son espace de noms. Cette association peut s'effectuer dans la balise ouvrante de n'importe quel élément par l'intermédiaire de l'attribut prédéfini *xmlns* dont la valeur est identifiée par un URI (*Uniform Ressource Identifier*).

A.2 DTD et XML Schéma

XML Schema a pour but de remplacer les DTD (*Document Type Définition*) existants. La DTD d'un document XML contient des informations de structure et de types des données du document XML. XML Schema présente des améliorations par rapport aux DTD, notamment une plus grande flexibilité et un typage plus précis des données [51].



▶ Figure A.1: Les technologies de la famille XML

```
<livres xmlns:eda="editeur-A"  xmlns:edb="editeur-B">
    <eda :livre>
         <eda: auteur>...</auteur>
         <eda: titre>...</titre>
    </eda:livre>
     <edb:livre  titre="..."  auteur="..."/>
</livres>
```

▶ Figure A.2: Exemple illustrant utilisant l'espace des noms

A.3 XSL (eXtensible Stylesheet Language)

XSL est un langage de feuilles de styles [3]. Il est composé de deux parties principales :

- XSLT (*XSL Transformation*) : langage qui permet de transformer facilement des documents XML en d'autres documents standards.
- XSL-FO (*XSL Flow Object*) : jeu d'instructions de formatage en XML destiné à la présentation.

A.4 XPointer

XPointer définit la syntaxe pour les fragments des documents XML [43]. Le but est de pouvoir adresser d'une manière précise des parties d'une ressource XML et de la représenter. XPointer est une extension de XPath, il réutilise en grande partie les mêmes concepts et syntaxe, il est aussi possible avec XPointer de faire des recherches avec des chaînes de caractères.

A.5 XLink

XLink fournit les constructions hypertexte de XML. Il permet d'effectuer des liens simples ou étendus (multisource, multicible) et des annotations (ressources contenant d'autres liens) [30]. De plus, chaque élément peut devenir un lien.

A.6 XQuery

XQuery est un langage de requête pour l'interrogation des documents XML [67], proposé par le W3C comme un standard qui prend avantage à la fois du contenu et de la structure spécifiés dans la collection, pour retourner les éléments XML qui satisfont strictement les conditions logiques spécifiées dans la requête. Le but est de combiner la flexibilité des moteurs de recherche plein texte, tout en tirant le maximum d'avantages de la connaissance de la structure comme dans les bases de données.

Annexe B

Résolution de l'équation $\frac{n^p-1}{n-1} = N$

Problème :
Nous cherchons $n \in]1, +\infty[$ tel que $\forall\ p \in \mathbb{N}^*$, on a :

$$\frac{n^p - 1}{n - 1} = N \tag{B.1}$$

où $N \in \mathbb{R}_+^*$ est un réel fixé.
Lemme :

$$\forall\ n \in]1, +\infty[,\ \forall\ p \in \mathbb{N}^*,\ \text{on a}\ \frac{n^p - 1}{n - 1} \geq p\ \sqrt[p]{\prod_{k=0}^{p-1} n^k} \tag{B.2}$$

Etapes :

1. La convexité de la fonction $x \mapsto -\log(x)$
   Définition : $f :\ I \to \mathbb{R}$ est dite convexe sur $I\ \Leftrightarrow$
   f est deux fois dérivable et $\forall\ x \in I, f''(x) \geq 0$ par la suite :
   $$\forall\ x, y \in I, \forall\ \lambda \in [0, 1[, f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \tag{B.3}$$
2. Soient $p \in \mathbb{N}^*$, $(\alpha_1 \dots \alpha_p) \in (\mathbb{R}_+^*)^p$ et $(x_1 \dots x_p) \in (\mathbb{R}_+^*)$, alors on a :
   $$\sqrt[p]{x_1 \dots x_p} \leq \frac{x_1 \dots x_p}{p} \tag{B.4}$$

On veut démontrer l'inégalité de l'équation B.2. On pose $n = m^2$, d'après l'équation B.2 nous aurons :

$$\forall\ n \in]1, +\infty[,\ \forall\ p \in \mathbb{N}^*,\ \text{on a}\ \frac{1 - n^p}{1 - n} = \sum_{k=0}^{p-1} m^{2k} \geq p\ \sqrt[p]{\prod_{k}^{p-1} m^{2k}} \tag{B.5}$$

Théorème de l'inégalité de Jensen (généralisation de l'équation B.3) : Etant donnée une application f convexe sur I, alors $\forall\ x_1 \dots x_p \in I,\ \forall \lambda_1 \dots \lambda_p \in [0,1]$ tels que $\lambda_1 \dots \lambda_p = 1$, on a :

$$f\left(\sum_{i=1}^{p} \lambda_i X_i\right) \leq \sum_{i=1}^{p} \lambda_i f(X_i)$$

$$\Leftrightarrow f(\lambda_1 X_1 \dots \lambda_p X_p) \leq \lambda_1 f(X_1) \dots \lambda_p f(X_p)$$

$f = -\log$ est convexe, soient $X_1 \dots X_p \in]0, +\infty[$ et $\lambda_1 = \dots = \lambda_p = \frac{1}{p}$ $(\sum_{i=1}^{p} \lambda_i = 1)$,

d'après le théorème de Jensen on aura : $\begin{array}{rcl} -\log\left(\frac{X_1+\dots+X_p}{p}\right) & \leq & -\frac{1}{p}(\log(X_1) + \dots + \log(X_p)) \\ \Leftrightarrow \quad log\left(\frac{X_1+\dots+X_p}{p}\right) & \geq & \frac{1}{p}\log(X_1 \times \dots \times X_p) \end{array}$

On applique la fonction $t \mapsto \exp^t \quad \Rightarrow$

$$\frac{X_1 + \dots + X_p}{p} \geq \exp^{\frac{1}{p}\log(X_1 \times \dots \times X_p)}$$

$$\exp^{\frac{1}{p}\log(X_1 \times \dots \times X_p)} = (X_1 \times \dots \times X_p)^{\frac{1}{p}} = \sqrt[p]{X_1 \times \dots \times X_p}$$

Nous remplaçons X_i par m^2 nous démontrons donc l'équation B.5, par la suite :

$\begin{array}{rcl} \sum_{k=0}^{p-1} m^{2k} & \geq & p\ \sqrt[p]{\prod_{k}^{p-1} m^{2k}} \ \Rightarrow \\ \frac{m^{2p}-1}{m^2-1} & \geq & p\sqrt[p]{m^{\sum_{k=0}^{p-1} 2k}} = p\sqrt[p]{m^{2p\frac{p-1}{2}}} \end{array}$

on aura par la suite :

$$p\sqrt[p]{m^{p(p-1)}} = p\ m^{p-1}$$

en remplaçant m par $\sqrt{n}$, on aura $\frac{n^p-1}{n-1} \geq p\ n^{\frac{p-1}{2}}$, d'après l'équation B.1 on a :

$$p\ n^{\frac{p-1}{2}} \approx N$$

d'où :

$$n \approx \left(\frac{N}{p}\right)^{\frac{2}{p-1}} \tag{B.6}$$

Annexe C

Les *NGrams*

C.1 Manipulation de requêtes avancées

Le processus débute par ajouter des marques de début et de fin pour chaque terme, soit #. Les 3Grams du terme *information* sont alors # *in*, *inf*, *for*, *orm*, *rma*, *mat*, *ati*, *tio*, *ion* et *on*#. L'index contient tous les NGrams pouvant apparaître au moins une fois dans un terme de la collection, chacun fait référence à tous les termes d'index dans lesquels il apparaît.
Considérons maintenant la requête *in*on*, le SRI doit retourner les documents contenant au moins un terme qui commence par *in* et qui se termine par *on*. Il suffit de traiter la requête booléenne # *in* et *on*#, une telle requête retourne les termes *information*, *invocation*, *inspiration*... Nous pouvons traiter différents types de requêtes en utilisant de simples transformations :

- x est transformée en $x\#$
- $x*$ est transformée en $x*$
- $*x$ est transformée en $x\#*$
- $*x*$ est transformée en $x*$
- $x * y$ est transformée en $y\#x*$

où x et y sont des chaines de caractères de la requête.
Le choix de la valeur de N pour la définition des NGrams n'est parfois pas adapté à certaines requêtes. Considérons la requête *inc**, si l'index est construit par des 3Grams, il faudra convertir la requête en # *in* et *inc*, outre les termes *incomparable* et *incorporé*,

des termes comme *instinctif* seront aussi retournés bien qu'ils ne représentent aucun intérêt pour l'utilisateur, il n'existe par ailleurs aucun autre moyen de convertir la requête en une forme booléenne qui utilise les 3Grams de l'index. Une solution assez simple consiste à un posttraitement de tous les termes retournés dont l'intérêt est de filtrer le résultat de la requête et d'en retenir que les termes qui répondent à la requête. Les SRI doivent implémenter des mécanismes de combinaison de requêtes en utilisant des opérateurs booléens. par exemple *re*d et fe*ri*. La forme de la requête la plus appropriée à cette requête est une conjonction de disjonctions de NGrams, puisque chaque requête est interprétée comme un disjonction de termes : le SRI accède à tous les documents qui contiennent au moins un terme de la forme *re*d* et un terme de la forme *fe*ri*. Même sans la combinaison de requêtes complexes, le traitement d'une requête avancée est coûteux, car il nécessite l'accès à un index spécial (celui des NGrams), le filtrage des résultats et ensuite l'accès à l'index principal. Un SRI peut fournir cette fonctionnalité, mais habituellement, elle est dissimulée par une interface dont la plupart des utilisateurs n'utilisent pas. Exposer une telle fonctionnalité les encourage à l'utiliser même si le besoin en information ne la demande pas.

C.2 La correction orthographique

La distance d'édition de Levenshtein que nous avons définie dans 3.3.4 est utilsée dans la correction orthographique [29]. Cependant, elle est très coûteuse car elle exige un ensemble de termes du vocabulaire avec lesquels on calcule une distance d'édition pour le terme à corriger. L'index NGrams peut être utilisé pour assister l'utilisateur à retrouver les termes du vocabulaire ayant de petites distances d'édition avec la requête. Il suffit d'identifier les termes qui partagent le maximum de NGrams avec la requête. La figure C.2 montre une portion de l'index des NGrams (dans ce cas, 2Grams) qui apparaissent dans la requête. Supposons que l'on cherche à identifier les termes qui contiennent au moins deux 2Grams de la requête, l'exemple de la figure C.2 permet d'énumérer les termes *forme*, *coordonnée*, *ferme* et *terme*.
Cette application retourne seulement les termes qui contiennent un nombre bien déterminé de NGrams de la requête : un terme comme *ferme*, une correction implausible de *foorme*, sera énuméré. Le besoin exige des mesures plus nuancées de l'intersection en NGrams entre la requête et les termes du vocabulaire. Nous pouvons utiliser la distance de Jaccard entre les ensembles A et B, définie par :

$$\frac{A \cap B}{A \cup B}$$

où A (resp. B) est l'ensemble des NGrams dans la requête (resp. dans un terme du vaocabulaire). On calcule pour chaque terme du vocabulaire sa distance à la requête, si elle dépasse un seuil, il fera partie de l'ensemble des termes candidats à la correction. Nous pouvons choisir parmi les termes candidats à la correction celui qui aura la distance d'édition la moins élevée.

	forme	focalise	coordonnée	orbite	sortie	ferme	armateur	terme	thème
fo	√	√							
oo			√						
or	√		√	√	√				
rm	√					√	√	√	
me	√					√		√	√

▶ Figure C.1: Exemple de correction orthographique par les 2Grams

Nous proposons de classifier un ensemble de termes selon leur appartenance à des catégories lexicales granulaires. Par opposition aux autres approches, notre approche est purement statistique et elle est indépendante des règles grammaticales et lexicales de la langue [5]. Nous basons notre approche sur le calcul de similarité de deux termes en se basant sur le contenu de chacun [113], nous nous sommes inspirés des approches de calcul de similarité entre les documents et en particulier de la mesure $tf \times idf$ utilisée pour la pondération des termes : on utilise une mesure semblable pour la pondération des éléments du contenu de chaque terme.

Annexe D

Prototype

L'ensemble des modules proposés a donné lieu au développement d'un prototype permettant l'indexation et l'interrogation de collections de documents XML. Le prototype est réalisé entièrement en langage Java (1.5) en utilisant des API telles que l'API DOM de Xerces pour parser les documents XML, l'utilisation de cette API (DOM) est recommandée pour faciliter l'accès aux données XML.

D.1 Indexation

L'indexation se fait sur deux volets :

- Indexation de la structure : référencer l'ensemble des nœuds dans la collection, ajouter et pondérer les arcs virtuels.
- Indexations du contenu : extraire les différentes caractéristiques de chaque terme de la collection.

Les deux indexes (contenu et structure) seront stockés dans une base MySql.

D.2 Représentation de la requête

Afin de pouvoir formuler des requêtes en XML, de nombreux langages ont été développés comme par exemple XQuery [33]. Le langage XQuery a été retenu comme un

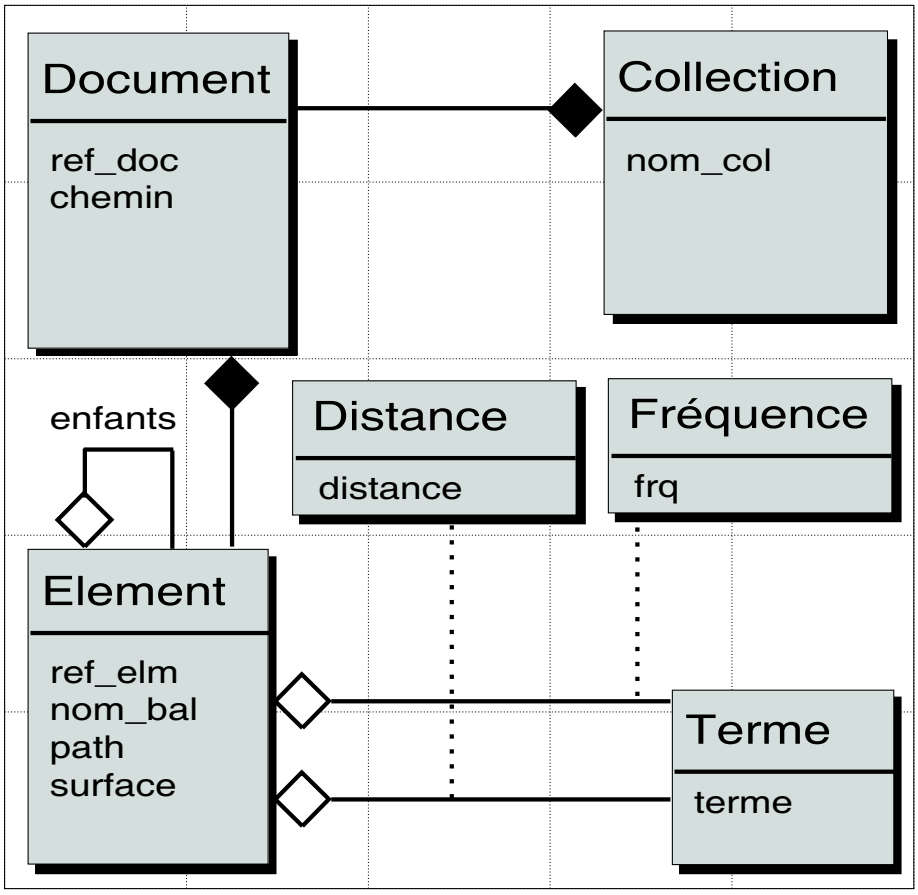
langage standard d'interrogation de documents XML, il permet de représenter la requête sous forme hiérarchique à base d'expression de chemins spécifique à sa syntaxe. En se basant sur l'idée de structurer la requête, le langage NEXI proposé dans le cadre de la compagne d'évaluation INEX est basé sur XPath [26]. Pour répondre à notre besoin (comparaison d'arbres), nous proposons de représenter la requête sous forme hiérarchique exprimée à l'aide d'arbre. Pour ce faire, nous avons défini une méthode simple de représentation de la requête, qui permet de transformer les requêtes exprimées en NEXI en un document XML.

D.3 Architecture macroscopique de notre modèle de recherche

L'index est stocké sous forme de tables dans une base de données relationnelle MySQL. La création de cet index se fait à travers une application Java. Ainsi les documents XML sont parcourus à l'aide d'un parseur DOM afin d'extraire les termes composant l'index de chaque document. La figure D.1 représente le diagramme des classes sur lesquels se base notre application. Ce diagramme est composé des classes *Document* comportant les caractéristiques d'un document de la collection. Une collection est décrite par son nom et comporte des attributs dérivés servant à faciliter la construction de l'index. Un document XML est composé d'éléments et chaque élément est une agrégation de termes.
Ces classes sont transformées en cinq tables relationnelles :

- **Collection**(*nom_ col*)
- **Document**(*ref_ doc*, chemin)
- **Element**(*ref_ elm*, nom_bal, path, surface)
- **Terme**(*terme*)
- **Distance**(distance)
- **Fréquence**(frq).

Le tableau D.3 représente la description des attributs de chaque relation. Il faut noter qu'il y a une dépendance des champs de type BLOB dans la table *Terme*, En effet l'ordre de mise à jour de ces champs est important, par exemple pour chaque élément il faut indiquer le cœfficient de propagation d'un terme dans cet élément (donc il y a une dépendance entre les champ *elements* et *distances*).



► Figure D.1: Diagramme de classe de notre système d'indexation

Les classes	**Description**
Terme	*terme* : représente la clé unique pour la table *Terme* qui a pour valeur le terme lui même.
Element	*ref_ elm* : représente la clé primaire de cette table et indique la référence de chaque élément de la collection. *nom_ bal* : le nom de l'élément. *path* : chemin du l'élément (basé sur la langage XPath). *surface* : l'approximation de propagation pour chaque sous-arbre de la collection (détaillé dans la section 3.8.5) (type BLOB).
Document	*ref_ doc* : représente la clé primaire de cette table et indique la référence de ce document. *chemin* : le chemin de ce document dans la collection (URL).
Collection	*nom_ col* : nom de la collection, clé primaire de la table.
Fréquence	*frq* : représente la fréquence du terme dans un élément.
Distance	*distance* : représente la distance qui sépare le terme à un élément.

Bibliographie

[1] M. Abolhassani and N. Fhur. Applying the divergence from randomness approach for content-only search in xml documents. *ECIR*, pages 409–419, 2004.

[2] G. W. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10(7-8):253–260, 1974.

[3] S. Alder. extensible stylesheet language (xsl), version 1.0. *Technical report, World Wide Web Consortium (W3C), W3C Recommandation*, 2001.

[4] V. N. Anh and A. Moffat. Compression and an ir approach to xml retrieval. *Proceedings of INEX 2002 Workshop, Dungstuhl, Germany*, 2002.

[5] M. Ben Aouicha, M. Tmar, M. Boughanem, and M. Abid. Une approche statistique basée sur les ngrams pour la classification de termes. *MAnifestation des Jeunes Chercheurs STIC, Lorient, 22/11/2006-24/11/2006*, pages Université de Bretagne–Sud, novembre 2006.

[6] M. Ben Aouicha, M. Tmar, M. Boughanem, and M. Abid. Experiments on element and document statistics for xml retrieval. *International Conference on Data, Information and Knowledge Management*, pages 88–94, 2008.

[7] M. Ben Aouicha, M. Tmar, M. Boughanem, and M. Abid. Modèle de recherche d'information structurée basé sur la relaxation de requêtes. *INFORSID*, 2008.

[8] M. Ben Aouicha, M. Tmar, M. Boughanem, and M. Abid. Restitution des fragments pertinents pour la recherche d'information dans des documents xml. *African Conference on Research in Computer Science and Applied Mathematics, CARI*, 2008.

[9] M. Ben Aouicha, M. Tmar, M. Boughanem, and M. Abid. Xml information retrieval based on tree matching. *IEEE International Conference on Engeneering of Computer-Based Systems ECBS*, pages 499–500, 2008.

[10] ApacheXindice. The apache xml project. *http://xml.apache.org/xindice/index.html*.

[11] L. Audibert. Etude des critères de désanbiguïsation sémantique automatique : résultats sur les occurences. *TALN*, 2003.

[12] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.

[13] N. J. Belkin, R. N. Oddy, and H. M. Brooks. Ask for information retrieval: part i.: background and theory. *Readings in information retrieval*, pages 299–304, 1997.

[14] P. Borlund. Measures of relative relevance and ranked halflife: performance indicators for interactive ir. *International ACM-SIGIR conference*, pages 24–28, 1998.

[15] M. Boughanem, A. Brini, and D. Dubois. Possibilistic networks for information retrieval. *International ACM SIGIR Conference - Workshop Information retrieval and applications of graphical models*, pages 67–89, 2007.

[16] M. Boughanem, C. Chrisment, J. Mothe, C. Soulé-Dupuy, and L. Tamine. Connectionist and genetic approaches to achieve ir. *Soft Computing in Information Retrieval Techniques and Applications*, pages 173–198, 2000.

[17] M. Boughanem, C. Chrisment, and C. Soulé-Dupuy. Query modification based on relevance back-propagation in adhoc environnement. *Information Processing & Management*, 35:121–139, 1999.

[18] M. Boughanem, W. Kraaij, and J-Y Nie. Modèles de langue pour la recherche d'information. *Les systèmes de recherche d'informations*, pages 163–182, 2004.

[19] M. Boughanem, Y. Loiseau, and H. Prade. Rank-ordering documents according to their relevance in information retrieval using refinements of ordered-weighted aggregations. *Proc. of the 3rd International Workshop on Adaptive Multimedia Retrieval (AMR'05) , Glasgow, UK*, pages 44–54, 2005.

[20] M. Boughanem, Y. Loiseau, and H. Prade. Refining aggregation functions for improving document ranking in information retrieval. *International Conference on Scalable Uncertainty Management (SUM 2007), Washington,DC, USA, 10/10/07-12/10/07*, 4772/2007:255–267, 2007.

[21] Neil Bradley. *The XML Companion*. Addison Wesley, 3rd edition, 2002.

[22] A. Brini, M. Boughanem, and D. Dubois. A model for information retrieval based on possibilistic networks. *String Processing and Information Retrieval (SPIRE)*, pages 271–282, 2005.

[23] J. P. Callan, W. B. Croft, and Hardings. The inquery retrieval system. *DEXA*, pages 78–83, 1992.

[24] L. M. Campos, J. M. Fernández-Luna, and J. F. Huete. Collaborative recommendations using bayesian networks and linguistic modelling. *ICAISC*, pages 1185–1197, 2008.

[25] D. Carmel, Y. Maarek, S. Mandelbrod, M. Mass, and A. Soffer. Searching xml documents via xml fragments. *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 151–158, 2003.

[26] C. Clark and S. Derose. Xml path language (xpath), version 1.0,w3c recommendation. 1999.

[27] C. CLEVERDON. Progress in documentation. evaluation of information retrieval systems. *Journal of Documentation*, 26:55–67, 1970.

[28] C. W. Cleverdon, J. Mills, and M. Keen. Fators determining the performance of indexing systems. *ASLIB Cranfield Research Project, Cranfield (UK).*, 33(1):1–14, 1997.

[29] F. J. Damerau. A technique for computer detection and correction of spelling errors. *CACM*, 7(3):171–176, 1964.

[30] S. Derose, E. Maler, and D. Orchard. Xml linking language (xlink), version 1.0. *Technical report, World Wide Web Consortium (W3C), W3C Recommandation*, 2001.

[31] R. Faiz. Identifying relevant sentences in news articles for event information extraction. *International Journal of Computer Processing of Oriental Languages (IJCPOL)*, 19:1–19, 2006.

[32] R. Faiz and A. Elkhlifi. Approche d'annotation automatique des événements. *Revue des Nouvelles Technologies de l'Information (RNTI)*, 1:37–42, 2008.

[33] M. F. Fernàndez, T. Jim, K. Morton, N. Onose, and J. Siméon. Highly distributed xquery with dxq. *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1159–1161, 2007.

[34] C. Fox. Lexical analysis and stoplists. *Frakes WB, Baeza-Yates (eds) Prentice Hall, New Jersey*, pages 102–132, 1992.

[35] E. A. Fox. Composite documents extended retrieval. *In Proceedings of ACM SIGIR, Montréal*, pages 45–53, 1985.

[36] N. Fuhr and K. Großjohann. Xirql: A query language for information retrieval in xml documents. *SIGIR*, pages 172–180, 2001.

[37] N. Fuhr, M. Lalmas, and S. Malik. Inex 2003 workshop proceedings. 2003.

[38] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[39] G. W. Furnas, S. Deerwester, S.T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. *SIGIR*, pages 465–480, 1988.

[40] M. Gilloux, E. Lassalle, and J. M. Ombrouck. Interrogation en langage naturel du minitel guide des services. *Data Compression Conference*, pages 1–20, 1993.

[41] N. Govert, M. Abolhassani, N. Fuhr, and K. Grossjohann. Content-oriented xml retrieval with hyrex. *Proceedings of INEX 2002 Workshop, Dungstuhl, Germany*, pages 26–32, 2002.

[42] N. Govert, G. Kazai, N. Fuhr, and M. Lalmas. Evaluating the effectiveness of content-oriented xml retrieval. *Technisher Bericht, University of Dortmund, Computer Science 6*, 2003.

[43] P. Grosso, E. Maler, J. Marsh, and N. Walsh. Xml pointer language (xpointer). *Technical report, World Wide Web Consortium (W3C), W3C Recommandation*, 2003.

[44] T. Grust. Accelerating xpath location steps. *In Proceedings of the ACM SIGMOD International Conference on MAnagement of Data, Madison, Wiscon, USA*, 2002.

[45] A. Gutierrez, R. Motz, and D. Viera. Building databases with information extracted from web documents. *SCCC '00: Proceedings of the XX International Conference of the Chilean Computer Science Society (SCCC'00)*, page 41, 2000.

[46] S. Harter. Psychological relevance and information science. *Journal of the American Society for Information Science (JASIS)*, 43:602–615, 1992.

[47] L. Hlaoua, M. Boughanem, and K. P. Sauvagnat. Combination of evidences in relevance feedback for xml retrieval. *ACM Conference on Information and Knowledge Management (CIKM)*, pages 893–896, 2007.

[48] L. Hlaoua, M. Torjmen, K. P. Sauvagnat, and M. Boughanem. Xfirm at inex 2006. ad-hoc, relevance feedback and multimedia tracks. *International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Allemagne, 18/12/2006-20/12/2006*, pages 373–386, 2007.

[49] G. Huck, I. Macherius, and P. Fankhauser. Pdom : Lightweigt persistency support for the document object model. *In Succeeding with object Databases, John Wiley*, 2000.

[50] D. A. Hull. Using statistical testing in the evaluation of retrieval experiments. *SIGIR*, pages 329–338, 1993.

[51] D. C. Fallside (IBM). Xml schema. *W3C Recommandation*, 2001.

[52] C. Jacquemin, B. Daille, J. Royanté, and X. Polanco. In vitro evaluation of a program for machine-aided indexing. *Inf. Process. Manage.*, 38(6):765–792, 2002.

[53] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[54] W. M. Shaw JR, R. Burgin, and P. Howell. Performance standards and evaluations in ir test collections: Cluster-based retrieval models. *Inf. Process. Manage.*, 33(1):1–14, 1997.

[55] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. *Proc. SIGIR*, pages 80–87, 2004.

[56] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Xml retrieval: what to retrieve? *SIGIR*, pages 409–410, 2003.

[57] Carl-Christian Kanne and Guido Moerkotte. Efficient storage of xml data. *ICDE*, page 198, 2000.

[58] G. Kazai and M. Lalmas. Inex 2005 evaluation meausures. *http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf*, pages 16–19, 2005.

[59] G. Kazai, M. Lalmas, and T. Roelleke. Focused document retrieval. *International Symposium on string processing and information retrieval, Lisbon, Portugal*, 2002.

[60] R. R. Korfhage. *Information Storage and Retrieval*. Wiley, 1997.

[61] K. L. Kwok. A neural network for probabilistic information retrieval. *SIGIR*, pages 21–30, 1989.

[62] M. Lalmas and B. Piwowarski. Inex 2005 relevance assessment guide. *https://inex.is.informatik.uniduisburg.de/2005/pdf/relevance Assessment2005.pdf*, 2005.

[63] R. R. Larson. Cheshire ii at inex: Using a hybrid logistic regression and boolean model for xml retrieval. *INEX Workshop*, pages 18–25, 2002.

[64] Y. K. Lee, S-J. Yoo, K. Yoon, and P. B. Berra. Index structures for structured documents. *In Proceedings of the ACM International Conference on Digital libraries, Bethesda, Maryland, USA*, pages 91–99, 1996.

[65] M. Lesk. Grab - inverted indexes with low storage overhead. *Computing Systems*, 1(3):207–220, 1988.

[66] A. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *SOV.PHY.DOhL*, pages 707–710, 1966.

[67] A. Levy, M. Fernandez, D. Suciu, D. Florescu, and A. deutsch. Xml-ql : A query language for xml. *Technical report, World Wide Web Consortium (W3C), Number NOTE-xml-ql-19980819*, 1998.

[68] R. W. Luka, H. Leong, T. S. Diloon, A. T. SHAN, W. B. Croft, and J. Allan. A survey in indexing and searching xml documents. *JASIST*, 53(6):415–437, 2002.

[69] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.

[70] Y. Mass and M. Mandelbrod. Retrieving the most relevant xml compnents. *Proceedings of INEX2003, Dagstuhl, Germany*, 2003.

[71] S. Mizzaro. Relevance, the whole (hi) story. *Journal of the American Society for Information Science*, 48:810–832, 1997.

[72] A. Moffat and L. Stuiver. Exploiting clustering in inverted file compression. *Data Compression Conference*, pages 82–91, 1996.

[73] P. Ogilvie and J. Callan. Using language models for flat text queries in xml retrieval. *Proceedings of INEX 2003 Workshop, Dungstuhl, Germany*, pages 12–18, 2003.

[74] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, CA, 1988.

[75] B. Piwowarski and P. Gallinari. A bayesian framework for xml information retrieval: Searching and learning with the inex collection. *Inf. Retr.*, 8(4):655–681, 2005.

[76] B. Piwowarski and M. Lalmas. Interface pour l'évaluation de systèmes de recherche sur des documents xml. *CORIA'2004, Toulouse, France*, pages 109–121, 2004.

[77] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. *Proc. SIGIR*, pages 275–281, 1998.

[78] E. Popovici, G. Ménier, and P. F. Marteau. Sirius: A lightweight xml indexing and approximate search system. *INEX*, pages 321–335, 2006.

[79] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[80] Y. Qiu and H. P. Frei. Improving the retrieval effectiveness by a similarity thesaurus. Technical Report 225, Zürich, Switzerland, 1995.

[81] Word Wide Web Consortium (W3C) Recommandation. Extensible markup language (xml).

[82] B. A. N. Ribeiro and R. Muntz. A belief network model for ir. *SIGIR*, pages 253–260, 1996.

[83] C. J. V. Rijisbergen. *Information Retrieval*. Dep of computer Science, University of Glasgow, 1979.

[84] C. J. Rijsbergen. A theoritical basis for use of co-occurrence data information retrieval. *Journal of Documentation*, 33(2):106–119, 1977.

[85] S. E. Robertson. How Okapi came to TREC. pages 287–299, 2005.

[86] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *JASIS*, 27:129–146, 1976.

[87] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland*, pages 232–241, 1994.

[88] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *SIGIR*, pages 232–241, 1994.

[89] J. J. Rocchio. Relevance feedback in information retrieval. *In The SMART Retrieval System Experiments in Automatic Document Processing*, pages 313–323, 1971.

[90] T. Rölleke, M. Lalmas, G. Kazai, I. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. *ECIR*, pages 284–302, 2002.

[91] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362, 1986.

[92] I. Ruthven and M. Lalmas. Using dempster-shafer's theory of evidence to combine aspects of information use. *J. Intell. Inf. Syst.*, 19(3):267–301, 2002.

[93] S. Jones M. Hancock-Beaulieu S. Robertson, S.Walker and M. Gatford. Okapi at trec 3. *In Proceedings of the 3rd Text REtrieval Conference*, 1994.

[94] G. Salton. A comparison between manual and automatic indexing methods. *Journal of the ACM (JACM)*, 15(4):493–513, 1968.

[95] G. Salton. The smart information retrieval system after 30 years - panel. *SIGIR*, pages 356–358, 1991.

[96] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. *SIGIR*, pages 49–58, 1993.

[97] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.

[98] G. Salton, E. Fox, and H. Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[99] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, 1968.

[100] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[101] G. Salton, A. W., and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[102] A. F. Sameaton. Using nlp or nlp ressources for information retrieval tasks. *In Natural Language Information Retrieval*, pages 99–1111, 1999.

[103] T. Saracevic. Relevance : A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science (JASIS)*, 2:321–343, 1975.

[104] T. Saracevic. Relevance reconsidered. *International Conference on Conceptions of Library and Information Science (COLIS)*, 39(3):201–218, 1996.

[105] T. Saracevic. Relevance reconsidered. *Conceptions of Library and Information Science*, pages 201–218, 1996.

[106] K. Sauvagnat and M. Boughanem. The impact of leaf nodes relevance values evaluation in a propagation method for xml retrieval. pages 19–22, 2004.

[107] K. Sauvagnat and M. Boughanem. Le langage de requête xfirm pour la recherche d'information dans des documents xml: de la recherche par simples mots-clés à l'utilisation de la structure des documents. *Congrès Informatique des Organisations et Systèmes d'Information et de Décision, INFORSID*, pages 107–124, 2004.

[108] K. Sauvagnat and M. Boughanem. Xfirm : A flexible information retrieval model for indexing and searching xml documents. *European Conference on Information Retrieval, ECIR*, pages 17–18, 2004.

[109] K. Sauvagnat and M. Boughanem. A la recherche de noeuds informatifs dans des corpus de documents xml. *CORIA*, pages 119–134, 2005.

[110] K. Sauvagnat, M. Boughanem, and C. Chrisment. Answering content and structure-based queries on xml documents using relevance propagation. *Information Systems*, 31(7):621–635, 2006.

[111] K. P. Sauvagnat, M. Boughanem, and C. Chrisment. Answering content-and-structure-based queries on xml documents using relevance propagation. *Information Systems*, 31:621–635, 2006.

[112] K. P. Sauvagnat, L. Hlaoua, and M. Boughanem. Xfirm at inex 2005: adhoc and relevance feedback tracks. *INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany, 28/11/05-30/11/05*, pages 88–103, 2005.

[113] J. Savoy. Indexation manuelle et automatique: une évaluation comparative basée sur un corpus en langue française. *CORIA*, pages 9–24, 2005.

[114] T. Schileder and H. Meuss. Querying and ranking xml documents. *Journal of the American Society for Information Science and Technology*, 53(6):489–503, 2002.

[115] M. Selkow. The tree-to-tree edition problem. *Information processing letters*, pages 184–186, 1997.

[116] D. Shasha and K. Zhang. Fast algorithms for the unit cost editing distance between trees. *J. Algorithms*, 11(4):581–621, 1990.

[117] D. Shin, H. Jang, and H. Jin. Bus : an effective indexing and retrieval scheme in structured documents. *In Proceedings of the ACM International Conference on Digital libraries, Pittsburgh, Pennsylvania, USA*, pages 235–243, 1998.

[118] B. Sigurbjisson, J. Kamps, and M. de Rijke. An element-based approach to xml retrieval. *Proceedings of INEX 2003 Workshop, Dagsthul, Germany*, 2003.

[119] B. Sigurbjisson, A. Trotman, S.Geva, M. Lalmas, B. Larsen, and S. Malik. Inex 2005 guidelines for topic development. *http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/TD05.pdf*, 2005.

[120] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. *SIGIR*, pages 21–29, 1996.

[121] M. Smith. Aspects of the p-norm model of information retrieval: Syntactic query generation, efficiency, and theoretical properties. Technical report, Ithaca, NY, USA, 1990.

[122] K-C Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.

[123] H. Tebri. *Formalisation et spécification d'un système de filtrage incrémental d'information*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2004.

[124] M. Tmar. *Modele auto-adaptatif de filtrage d'information : apprentissage incremental du profil et de la fonction de decision*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, juillet 2002.

[125] M. Tmar and M. Boughanem. Learning profile in routing: Comparison between relevance and gradient back-propagation. *SPIRE*, pages 253–259, 2000.

[126] G. Torsten. Storage and retrieval of xml documents within a cluter of database systems. *Thèse de doctorat, Institut fédéral de technologie, Zurich, Suisse*, 2003.

[127] A. Trotman, S. Geva, and J. Kamps, editors. *Proceedings of the SIGIR 2007 Workshop on Focused Retrieval*. University of Otago, Dunedin New Zealand, 2007.

[128] A. Trotman and M. Lalmas. Why structural hints in queries do not help xml-retrieval. *ACM SIGIR conference on research and development in Information Retrieval*, pages 711–712, 2006.

[129] A. Trotman and B. Sigurbjisson. Narrowed extended xpath i (nexi). *In INEX 2003 Proceedings, Dagsthul, Allemangne*, pages 219–237, 2004.

[130] H. Trutle and W. B. Croft. Inference networks for document retrieval. *Proc. SIGIR*, pages 1–24, 1989.

[131] H. Trutle and W. B. Croft. Evaluation of an inference network-based retrieval model. *TOIS*, 9(3):187–222, 1991.

[132] W3C. Extensible markup language (xml) 1.0.

[133] Y. Wang, D. J. DeWitt, and J. Y. Cai. X-diff : An effective change detection algorithm for xml documents. *IEEE International Conference on Data Engineering*, pages 519–530, 2003.

[134] G. Weikum, A. C. König, and S. Deßloch, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*. ACM, 2004.

[135] R. Wilkinson and P. Hingston. Using the cosine measure in a neural network for document. *SIGIR*, pages 202–210, 1991.

[136] S. A. Yahia, S. Cho, and D. Srivastava. Tree pattern relaxation. *EDBT*, pages 496–513, 2002.

[137] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[138] G. K. Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.