Apprentissage d'ordonnancements en Recherche d'Information Structurée

Jean-Noël Vittaut — Patrick Gallinari

Laboratoire d'Informatique de Paris 6
8, rue du Capitaine Scott
F-75015 Paris, France
{vittaut, gallinari}@poleia.lip6.fr

RÉSUMÉ. *Nous présentons un modèle d'apprentissage pour la Recherche d'Information Structurée qui ajuste automatiquement ses paramètres grâce à un ensemble d'exemples étiquetés composé de requêtes et de jugements de pertinence sur un ensemble de parties de documents. Notre modèle améliore la performance d'un système de base de Recherche d'Information en optimisant un critère de coût d'ordonnancement et en combinant des scores calculés sur des parties de documents et leur contexte structurel. Nous analysons la performance de notre algorithme sur la collection INEX et nous le comparons au modèle de base qui est une adaptation d'Okapi pour la Recherche d'Information Structurée.*

ABSTRACT. *We present a Machine Learning based Ranking model which can automatically learn its parameters using a training set of annotated examples composed of queries and relevance judgments on a subset of the document elements. Our model improves the performance of a baseline Information Retrieval system by optimizing a ranking loss criterion and combining scores computed from document elements and from their local structural context. We analyze the performance of our algorithm on INEX collection and compare it to the baseline model which is an adaptation of Okapi to Structured Information Retrieval.*

MOTS-CLÉS : *structure, ordonnancement, apprentissage*

KEYWORDS: *structure, ranking, machine learning*

1. Introduction

Des collections hiérarchisées de documents, eux-même encodés dans une représentation structurée comme le XML, XHTML, RDF ou RSS sont maintenant disponibles et la communauté de Recherche d'Information (RI) commence à développer des moteurs de recherche spécialement adaptés à ce type de documents [FUH 05] [BAE 04]. La structure du document offre de nouvelles possibilités comme répondre à des requêtes avec des contraintes structurelles (requêtes *Content and Structure* d'INEX[1] [FUH 05]), ou simplement fournir à l'utilisateur une liste d'éléments de documents de différentes granularités (requêtes *Content Only* d'INEX). Ces éléments peuvent correspondre à différents types d'éléments dans un document structuré. Dans cet article, nous abordons cette dernière problématique avec des requêtes basées sur le contenu. Une des difficultés de cette tâche est de comparer et d'ordonner des éléments de documents qui possèdent des caractéristiques très différentes comme leur longueur, leur chevauchement, leur homogénéité thématique, etc. Les moteurs de recherche traditionnels ont été développés pour ordonner des documents similaires et ne sont pas adaptés pour cette nouvelle tâche. Différents modèles existent déjà pour donner des scores de pertinence à des éléments dans des documents structurés. Par exemple, la théorie de l'évidence a été utilisée pour agréger de l'évidence provenant de sous-éléments d'un document [LAL 97] [LAL 00]. Les réseaux bayésiens [PIW 04] ou des modèles de langage sont d'autres formalismes utilisés pour combiner l'évidence de sous-éléments pour donner un score de pertinence à l'élément contenant.

Les algorithmes d'ordonnancement ont récemment donné lieu à différentes études et développements dans la communauté de l'apprentissage. Dans le domaine des documents textuels, ils ont été utilisés avec succès pour combiner des caractéristiques ou des relations de préférence dans des tâches comme la meta-recherche [COH 98] [BAR 94] [FRE 98], le résumé automatique [AMI 05] et plus récemment pour la combinaison de différentes sources de connaissance en RI [CRA 05]. Une des difficultés de ces méthodes est la complexité des algorithmes qui est en général quadratique en fonction du nombre d'exemples. C'est pour cette raison que l'ordonnancement concerne la plupart du temps des problèmes à deux classes. Néanmoins, des solutions linéaires ont été proposées [AMI 05] [FRE 98] et sous certaines conditions, ces méthodes peuvent converger rapidement [CLÉ 05].

Nous proposons de développer et d'utiliser des méthodes d'ordonnancement pour la tâche de Recherche d'Information Structurée (RIS), qui consiste à produire une liste ordonnée d'éléments de documents qui répondent à une requête de contenu. L'ordonnancement peut être particulièrement utile en RIS en raison de la difficulté intrinsèque de cette tâche, comme nous l'avons déjà signalé, et aussi parce que les moteurs de recherche actuels ne sont pas bien adaptés à cette nouvelle tâche. Nous pensons que les algorithmes d'ordonnancement peuvent améliorer les performances des techniques existantes. Les algorithmes d'ordonnancement fonctionnent en combinant des carac-

1. INEX est l'«INitiative for the Evaluation of XML Retrieval» et fait partie du réseau d'excellence DELOS

téristiques issues des objets que l'on souhaite ordonner. Dans notre cas, ces caractéristiques proviennent à la fois de l'élément du document et de son contexte structurel. A partir d'un ensemble d'exemples, l'algorithme d'ordonnancement apprend comment combiner ces différentes caractéristiques de manière optimale selon une certaine fonction de coût.

Nous présentons cet article de la manière suivante : dans la partie 2 nous décrivons le modèle d'ordonnancement, dans la partie 3 nous montrons comment il peut être adapté à la Recherche d'Information Structurée. Enfin, dans la partie 4, nous décrivons les expériences menées sur la collection INEX, et nous comparons l'algorithme à son modèle de base qui est une adaptation d'Okapi pour la RIS.

2. Cadre général

Nous présentons dans cette partie un modèle général d'ordonnancement qui peut être adapté à la RI ou la RIS. L'idée des algorithmes d'ordonnancement proposés dans la communauté de l'apprentissage est d'apprendre une relation d'ordre totale sur un ensemble $\mathcal{X}$, ce qui permet de comparer deux à deux n'importe quels éléments de cet ensemble. Etant donné cet ordre total, nous pouvons produire une liste ordonnée à partir de n'importe quel sous-ensemble de $\mathcal{X}$. Par exemple, en RI, $\mathcal{X}$ peut être l'ensemble des documents pertinents pour une requête, et la relation d'ordre totale est celle induite par les scores de ces documents.

Comme pour n'importe quelle technique d'apprentissage, nous avons besoin d'un ensemble d'exemples étiquetés afin d'apprendre comment ordonner. Cet ensemble d'apprentissage sera composé de couples ordonnés d'exemples, ce qui fournira un ordre partiel sur les éléments de $\mathcal{X}$. L'algorithme d'ordonnancement utilisera cette information pour apprendre un ordre total sur les éléments de $\mathcal{X}$, ce qui nous permettra ensuite d'ordonner de nouveaux éléments. Par exemple en RI, l'ordre partiel peut provenir de jugements de pertinence, fournis par un expert, sur différents documents et pour une certaine requête.

Nous introduisons ci-dessous quelques notations qui seront utiles pour comparer des sous-ensembles de l'ensemble partiellement ordonné $\mathcal{X}$.

2.1. *Notations*

Soit $\mathcal{X}$ un ensemble muni d'un ordre partiel $\prec$. Cela signifie que certaines paires d'éléments de $\mathcal{X}$ peuvent être comparées selon la relation $\prec$. On note $x \perp x'$ s'il n'existe aucune préférence entre deux éléments x and x'.

Par exemple en RI, $\mathcal{X}$ sera l'ensemble des couples (document, requête) pour tous les documents de la collection et toutes les requêtes possibles. Cet ensemble est partiellement ordonné grâce à des jugements de pertinence existants pour certaines requêtes et certains documents.

2.2. *Ordonnancement*

Soit f une application de $\mathcal{X}$ dans l'ensemble des réels. Nous pouvons associer à f une relation d'ordre total $\prec_T$ de la manière suivante :

$$x \prec_T x' \Leftrightarrow f(x) < f(x') \; . \tag{1}$$

Il apparaît clairement qu'apprendre l'application f revient à apprendre un ordre total sur $\mathcal{X}$. Par la suite, nous étendrons l'ordre partiel $\prec$ à un ordre total $\prec_T$, et donc nous utiliserons la même notation pour les deux relations.

Dans notre modèle, nous représenterons un élément de $\mathcal{X}$ par un vecteur réel de caractéristiques $x = (x_1, x_2, ..., x_d)$. Ici, les caractéristiques seront des scores calculés au niveau de différents éléments structurellement liés à un morceau de document. Par la suite, f sera une combinaison linéaire des coordonnées de x :

$$f_\omega(x) = \sum_{k=1}^{d} \omega_k x_k \tag{2}$$

où $\omega = (\omega_1, \omega_2, ..., \omega_d)$ sont les paramètres de la combinaison que l'on apprend.

2.2.1. *Coût d'ordonnancement*

On dit que f_ω respecte $x \prec x'$ si $f_\omega(x) < f_\omega(x')$. Dans ce cas, le couple (x, x') est dit bien ordonné par f_ω. Le coût d'ordonnancement [FRE 98] mesure à quel point f_ω respecte $\prec$. Par définition, il est égal au nombre de couples de $\{(x, x') \in \mathcal{X}^2 / x \prec x'\}$ mal ordonnés par f_ω :

$$R(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} \chi(x, x', \omega) \tag{3}$$

où $\chi(x, x', \omega) = 1$ si $f_\omega(x) > f_\omega(x')$ et 0 sinon.

L'ordonnancement consiste à apprendre ω pour minimiser $\omega \rightarrow R(\mathcal{X}, \omega)$.

2.2.2. *Coût exponentiel*

En pratique, cette expression n'est pas très utile car $\omega \rightarrow \chi(x, x', \omega)$ n'est pas différentiable et donc les algorithmes d'ordonnancement optimisent plutôt une autre fonction de coût appelée coût exponentiel :

$$R_e(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} e^{f_\omega(x) - f_\omega(x')} \tag{4}$$

On montre facilement que $R(\mathcal{X}, \omega) \leq R_e(\mathcal{X}, \omega)$. De plus $\omega \rightarrow R_e(\mathcal{X}, \omega)$ est différentiable et convexe, et peut donc être minimisée en utilisant des techniques classiques d'optimisation. Minimiser $\omega \rightarrow R_e(\mathcal{X}, \omega)$ conduit à minimiser $\omega \rightarrow R(\mathcal{X}, \omega)$.

Pour minimiser $\omega \rightarrow R_e(\mathcal{X}, \omega)$, on peut utiliser une descente de gradient. Les composantes du gradient de R_e sont :

$$\frac{\partial R_e}{\partial \omega_k}(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} (x_k - x'_k) e^{f_\omega(x) - f_\omega(x')} \ . \tag{5}$$

Sans autre hypothèse, la complexité du calcul de ces composantes est $O(|\mathcal{X}|^2)$.

2.2.3. *Propriétés*

De l'expression de (4), on peut déduire deux propriétés élémentaires qui permettront de réduire la complexité de l'algorithme d'apprentissage.

Propriété 1 *Si les éléments de $\mathcal{X}_r$ ne sont pas comparables aux éléments de $\mathcal{X}_{r'}$, le coût d'ordonnancement peut s'exprimer comme la somme de deux sous-coûts d'ordonnancement sur $\mathcal{X}_r$ et $\mathcal{X}_{r'}$:*

$$R_e(\mathcal{X}_r \cup \mathcal{X}_{r'}, \omega) = R_e(\mathcal{X}_r, \omega) + R_e(\mathcal{X}_{r'}, \omega) \ . \tag{6}$$

Propriété 2 *Si l'on préfère n'importe quel élément de $\mathcal{X}_r^j$ à n'importe quel élément de $\mathcal{X}_r^{j'}$, et que les éléments de $\mathcal{X}_r^j$ (resp. $\mathcal{X}_r^{j'}$) ne sont pas comparables entre eux, on note cela $\mathcal{X}_r^{j'} \prec \mathcal{X}_r^j$ et le coût d'ordonnancement peut s'exprimer comme le produit de deux sommes sur $\mathcal{X}_r^j$ et $\mathcal{X}_r^{j'}$:*

$$R_e(\mathcal{X}_r^j \cup \mathcal{X}_r^{j'}, \omega) = \left(\sum_{x \in \mathcal{X}_r^j} e^{f_\omega(x)} \right) \left(\sum_{x \in \mathcal{X}_r^{j'}} e^{-f_\omega(x)} \right) \ . \tag{7}$$

La figure 1 illustre les différentes hypothèses de ces propriétés et la partition qu'elles induisent. Compte tenu de cette partition, le coût exponentiel (4) se réécrit :

$$R_e(\mathcal{X}, \omega) = \sum_r \sum_{\substack{(j,j') \\ \mathcal{X}_r^{j'} \prec \mathcal{X}_r^j}} \left\{ \left(\sum_{x \in \mathcal{X}_r^j} e^{f_\omega(x)} \right) \left(\sum_{x \in \mathcal{X}_r^{j'}} e^{-f_\omega(x)} \right) \right\} . \tag{8}$$

La complexité pour calculer cette expression est en $O(K.|\mathcal{X}|)$ alors qu'elle est en $O(|\mathcal{X}|^2)$ pour (4) où K est le nombre total de sous-ensembles $\mathcal{X}_r^j$ dans la partition de $\mathcal{X}$. Le pire cas a lieu quand $K = |\mathcal{X}|$. Dans le cas de jugements selon deux classes (pertinent et non pertinent), on a $K = 2$ et donc une complexité linéaire en $|\mathcal{X}|$.

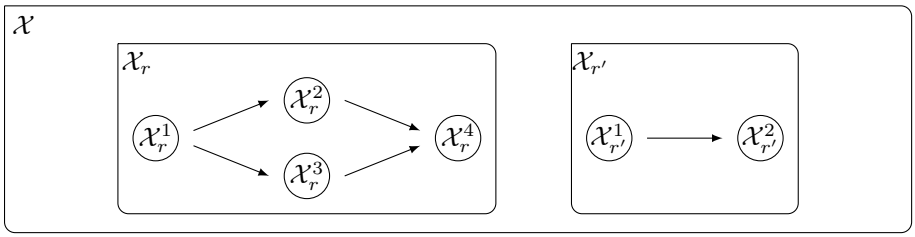

Figure 1. *Représentation d'une partition de $\mathcal{X}$. Il n'y a pas de préférence entre un élément de $\mathcal{X}_r$ et un élément de $\mathcal{X}_{r'}$ (propriété 1). A l'intérieur de $\mathcal{X}_r$, la flèche $\mathcal{X}_r^1 \rightarrow \mathcal{X}_r^2$ signifie qu'un élément de $\mathcal{X}_r^1$ est toujours préféré à un élément de $\mathcal{X}_r^2$ et qu'à l'intérieur de chaque $\mathcal{X}_r^i$, $i = 1...4$, il n'y a pas de préférence (propriété 2).*

2.2.4. *Descente de gradient*

Puisque la fonction (8) est convexe, on peut utiliser une descente gradient pour la minimiser. Les composantes du gradient ont la forme suivante :

$$\begin{aligned}\frac{\partial R_e}{\partial \omega_k}(\mathcal{X},\omega) &= \sum_r \sum_{\substack{(j,j') \\ \mathcal{X}_r^{j'} \prec \mathcal{X}_r^j}} \left\{ \left(\sum_{x \in \mathcal{X}_r^j} x_k e^{f_\omega(x)} \right) \left(\sum_{x \in \mathcal{X}_r^{j'}} e^{-f_\omega(x)} \right) \right. \\ &\quad \left. + \left(\sum_{x \in \mathcal{X}_r^j} e^{f_\omega(x)} \right) \left(\sum_{x \in \mathcal{X}_r^{j'}} -x_k e^{-f_\omega(x)} \right) \right\} . \end{aligned} \tag{9}$$

La complexité pour calculer cette expression est la même que celle de (8), c'est à dire $O(K.|\mathcal{X}|)$.

3. Application à la Recherche d'Information Structurée

3.1. *Définitions*

On suppose que l'on dispose d'une collection de documents hiérarchiquement structurés. On peut représenter chaque document par un arbre A. Chaque noeud de l'arbre possède un «type»[2] et un contenu textuel. On appelle $\mathcal{T}$ l'ensemble des types de noeuds.

Pour chaque noeud n de A, on appelle doxel du noeud n le sous arbre A_n de A ayant pour racine n.

2. Ou étiquette, mais nous n'utilisons pas cette terminologie pour éviter toute confusion avec la notion d'exemple étiqueté en apprentissage.

On utilise les notations suivantes, $\mathcal{D}$ est l'ensemble de tous les doxels de tous les documents de la collection, $\mathcal{R}$ est l'ensemble des demandes d'information (représentées par des requêtes) et $\mathcal{X} = \mathcal{R} \times \mathcal{D}$ est l'ensemble des éléments à ordonner.

On suppose de plus qu'il existe un ordre partiel $\prec$ sur $\mathcal{X} = \mathcal{R} \times \mathcal{D}$. Cet ordre partiel reflète, pour un certain nombre de demandes d'information, la connaissance que nous avons sur les préférences entre doxels. Cette connaissance peut provenir d'un retour utilisateur, ou d'un ensemble de jugements d'experts sur le corpus de RIS. Il est à noter que seule une partie de la collection nécessite ces jugements de pertinence. On considère ici la tâche qui consiste à produire une liste ordonnée de doxels qui répondent à une demande d'information $r \in \mathcal{R}$. Pour cela, on entraîne le modèle d'ordonnancement pour qu'il apprenne un ordre total strict sur $\mathcal{X}$ [3].

3.2. *Représentation*

On représente chaque élément $x \in \mathcal{X}$ par un vecteur $(x_1, x_2, ..., x_d)$ où x_k représente une caractéristique dont on pense qu'elle pourrait être utile pour ordonner les éléments de $\mathcal{X}$. Pour prendre en compte l'information de structure, nous utiliserons plusieurs informations : d'une part celle fournie par le contexte du doxel, et d'autre part celle correspondant au type de noeud. Pour la première, nous utiliserons des caractéristiques sur le doxel, son parent et la totalité du document. Pour la seconde, la connaissance du type permet d'effectuer une première étape où l'on ordonne les doxels du même type, (ce que revient à apprendre une fonction $f_{\omega|t}$ pour chaque type de noeud t) puis une seconde étape d'ordonnancement où l'on utilise les $f_{\omega|t}(x)_{t=1..|\mathcal{T}|}$ comme caractéristiques.

Ces deux étapes se ramènent à une seule en considérant la représentation vectorielle suivante. En notant $x^t = (x_1^t, x_2^t, ..., x_d^t)$ le vecteur représentant x pour le type de noeud t, on a $x_i^t = x_i$ si t est le type de noeud de la racine de x et $x_i^t = 0$ sinon. On considère alors les vecteurs définis de la manière suivante :

$$x = \left((x_1^{t_1}, x_2^{t_1}, ..., x_d^{t_1}), (x_1^{t_2}, x_2^{t_2}, ..., x_d^{t_2}), ..., (x_1^{t_{|\mathcal{T}|}}, x_2^{t_{|\mathcal{T}|}}, ..., x_d^{t_{|\mathcal{T}|}}) \right)$$

où $t_i \in \mathcal{T}$ et $|\mathcal{T}|$ est le nombre de type de noeuds différents dans la collection. Dans l'expression précédente, toutes les composantes du vecteur de la forme $(x_i^{t_i}, x_2^{t_i}, ..., x_d^{t_i})$ seront égales à $(0, ..., 0)$ sauf pour celle correspondant au type de noeud de x. Cette représentation nous permet de calculer en une étape l'ordonnancement de doxels de types différents.

---

3. Pour simplifier, on considère ici un ordre strict sur les doxels, et non le cas où des doxels différents ont la même préférence

3.3. *Réduction de la complexité*

3.3.1. *Grâce à la propriété 1*

Afin de réduire la complexité, il faut trouver des sous-ensembles de $\mathcal{X}$ qui vérifient les hypothèses de la propriété 1. On sait que comparer des scores de doxels pour des requêtes différentes n'a pas de sens. On peut donc partitionner l'ensemble suivant les différentes demandes d'information. Les éléments de la partition sont donc les :

$$\mathcal{X}_r = \{x = (r, d) \ / \ d \in \mathcal{D}\} \tag{10}$$

pour toute requête $r \in \mathcal{R}$. $\mathcal{X}_r$ est l'ensemble des couples (demande d'information, doxel) qui possèdent la même demande d'information r.

3.3.2. *Grâce à la propriété 2*

Pour tout $\mathcal{X}_r$, les préférences entre doxels peuvent être exprimés selon une échelle discrète à plusieurs dimensions. Par exemple dans INEX, nous avons :

- une information d'exhaustivité, qui mesure à quel point un doxel répond à la totalité de la demande d'information (0 pas du tout exhaustif, ..., 3 signifie totalement exhaustif)
- une information de spécificité, qui mesure à quel point le doxel répond uniquement à la demande d'information (0 pas du tout spécifique, ..., 3 signifie totalement spécifique)

Un doxel étiqueté E_3S_3 (ce qui signifie totalement exhaustif et spécifique) est préféré à un doxel étiqueté E_1S_3 (ce qui signifie partiellement exhaustif et totalement spécifique). Si une telle échelle discrète multidimensionnelle existe, on peut trouver une partition vérifiant les hypothèses de la propriété 2 en considérant les ensembles $\mathcal{X}_r^{(e,s)}$ pour lesquels les doxels partagent la même valeur de préférence selon toutes les dimensions :

$$\mathcal{X}_r^{(e,s)} = \{x = (r, d) \ / \ d \text{ est jugé } E_eS_s \text{ pour } r\} \ . \tag{11}$$

Par exemple, $\mathcal{X}_r^{(1,3)}$ sera l'ensemble des doxels jugés E_1S_3 pour la requête r.

4. Expériences

4.1. *Collection utilisée*

Pour évaluer notre méthode, nous avons utilisé la collection de documents, de requêtes et de jugements d'INEX. Cette collection contient 16819 documents XML représentant le contenu d'articles du journal IEEE Computer Society's de 1995 à 2004. Ces documents sont représentés avec la même DTD. En 2003, 36 requêtes orientées sur le contenu et les jugements correspondants ont été produits. En 2004, 40

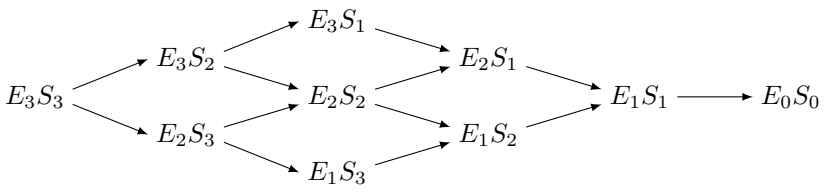

Figure 2. *Treillis représentant l'ordre entre les éléments pour une requête donnée, selon l'échelle discrète bidimensionnelle d'INEX. Les doxels étiquetés E_3S_3 doivent être positionnés le plus haut possible dans la liste.*

requête supplémentaire ont été ajoutées. Ces jugements de 2003 et 2004 portent uniquement sur 12107 documents de 1995 à 2002, puisque les journaux des années suivantes n'étaient pas encore disponibles. En 2005, la collection a été complétée par de nouveaux articles de 2003 et 2004 et 40 requêtes supplémentaires ont été produites.

Une requête orientée sur le contenu est une liste de mots représentant une demande d'information. Un jugement évalue à quel point un doxel de la collection répond correctement à une demande d'information. Dans INEX, les jugements sont exprimés selon une échelle discrète multidimensionnelle qui a été décrite plus haut dans la partie 3.3. Les jugements et le treillis représentant l'ordre partiel entre ces jugements sont donnés sur la figure 2.

4.2. *Représentation*

Pour calculer les caractéristiques, nous avons utilisé le modèle Okapi [ROB 92], qui est un des modèles de référence pour la recherche de documents plats. Il a été adapté afin d'obtenir de bonnes performances sur la collection de documents structurés. Cette adaptation consiste à utiliser les doxels plutôt que les documents dans le calcul des fréquences de termes, et d'utiliser la taille moyenne des doxels de même type comme facteur de normalisation pour chaque doxel.

Dans les expériences, nous avons utilisé un terme de biais et trois caractéristiques pour la combinaison :

- $x_1 = 1$ (le terme de biais)
- $x_2 =$ le score Okapi du doxel
- $x_3 =$ le score Okapi du parent du doxel
- $x_4 =$ le score Okapi du document entier.

Ces caractéristiques fournissent de l'information sur le contexte structurel d'un doxel. L'ensemble des types de noeuds a été défini suivant la DTD de la collection de documents : article, abstract, sections, paragraphs, lists... Le type de noeud a été introduit dans la représentation suivant la méthode décrite à la partie 3.2.

Nous avons utilisé comme base d'apprentissage l'ensemble des requêtes et des jugements produits en 2003 et 2004. Ceux produits en 2005 ont servi de base de test.

4.3. *Filtrage*

Un système de RIS doit éviter de renvoyer des doxels se chevauchant, ce qui signifie par exemple qu'il ne devrait pas renvoyer à la fois un paragraphe et la section qui le contient. Afin de supprimer le chevauchement dans une liste existante, nous avons utilisé la stratégie qui consiste à retirer tous les éléments qui chevauchent un autre élément classé plus haut dans la liste. On peut trouver d'autres méthodes de diminution du chevauchement dans [KAZ 01]. Deux types d'évaluations ont été menées : un avec chevauchement, et un sans chevauchement.

4.4. *Evaluation*

Nous avons utilisé deux mesures pour évaluer notre approche :

- une mesure de précision-rappel qui ne prend pas en compte le phénomène de chevauchement ;
- une mesure basée sur le gain cumulé [JäR 02]. Cette mesure, développée pour l'évaluation d'INEX [KAZ 05], prend en compte la dépendance entre les éléments d'un document XML, et peut pénaliser les listes avec chevauchement.

4.5. *Résultats*

4.5.1. *Sans chevauchement*

Nous avons tracé sur la figure 3 l'évaluation des listes produites par l'algorithme d'ordonnancement et par le modèle Okapi modifié, dans le cas où le chevauchement a été supprimé. On peut voir pour les deux mesures que l'algorithme d'ordonnancement a de meilleurs performances que le modèle Okapi. La différence pour la mesure de précision-rappel n'est pas très importante et est due au filtrage de listes a posteriori. Les listes ordonnées n'ont pas été optimisées pour que les éléments ne se chevauchent pas, puisqu'il n'y a pas de notion de chevauchement dans le coût exponentiel.

4.5.2. *Avec chevauchement*

La figure 4 représente l'évaluation des listes produites par l'algorithme d'ordonnancement et par le modèle Okapi modifié, dans le cas où le chevauchement était autorisé. On peut voir pour les deux mesures que l'algorithme d'ordonnancement est clairement plus performant que le modèle Okapi, et que la différence de performance est plus grande que dans le cas sans chevauchement.

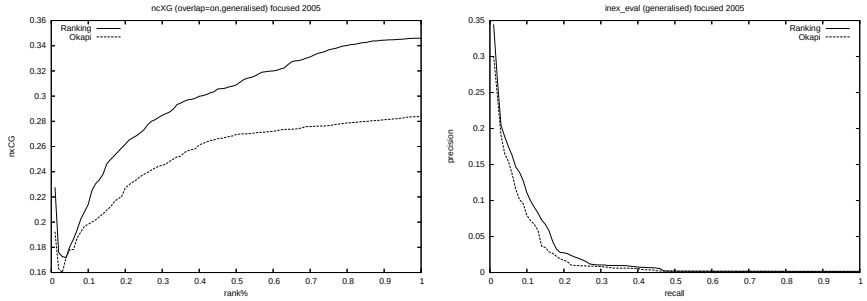

Figure 3. *Performance de l'algorithme d'ordonnancement et du modèle Okapi sans chevauchement évaluée avec la mesure de gain cumulé (à gauche) et la mesure de précision-rappel (à droite).*

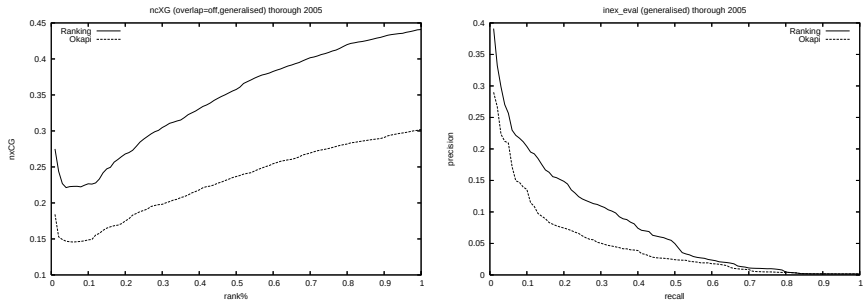

Figure 4. *Performance de l'algorithme d'ordonnancement et du modèle Okapi avec chevauchement évaluée avec la mesure de gain cumulé (à gauche) et la mesure de précision-rappel (à droite).*

Pour les deux séries d'expériences, l'algorithme d'ordonnancement a pu améliorer de manière significative la performance du modèle de base Okapi. Les méthodes d'ordonnancement paraissent être un axe de recherche prometteur pour améliorer les moteurs de recherche en RIS. Restent à conduire d'autres tests avec des caractéristiques supplémentaires (par exemple les scores d'autres systèmes de RI).

5. Conclusion

Nous avons décrit un nouveau modèle pour la Recherche d'Information Structurée. Il repose sur la combinaison de scores d'un modèle Okapi et prend en compte la structure du document. Cette combinaison est apprise automatiquement à partir d'un ensemble étiqueté par un algorithme d'ordonnancement. Nous avons montré que l'apprentissage pour ordonner des doxels améliore de manière significative le modèle de base Okapi, qui est déjà connu pour être efficace en RI sur des documents plats.

Remerciements

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

6. Bibliographie

[AMI 05] AMINI M.-R., USUNIER N., GALLINARI P., « Automatic Text Summarization Based on Word-Clusters and Ranking Algorithms. », *ECIR*, 2005, p. 142-156.

[BAE 04] BAEZA-YATES R., MAAREK Y. S., ROELLEKE T., DE VRIES A. P., « Third edition of the "XML and information retrieval" workshop », *SIGIR Forum*, , 2004.

[BAR 94] BARTELL B. T., COTTRELL G. W., BELEW R. K., « Automatic Combination of Multiple Ranked Retrieval Systems », *Research and Development in Information Retrieval*, 1994, p. 173-181.

[CLÉ 05] CLÉMENÇON S., LUGOSI G., VAYATIS N., « Ranking and Scoring Using Empirical Risk Minimization. », AUER P., MEIR R., Eds., *COLT*, vol. 3559 de *Lecture Notes in Computer Science*, Springer, 2005, p. 1-15.

[COH 98] COHEN W. W., SCHAPIRE R. E., SINGER Y., « Learning to Order Things », JORDAN M. I., KEARNS M. J., SOLLA S. A., Eds., *Advances in Neural Information Processing Systems*, vol. 10, The MIT Press, 1998.

[CRA 05] CRASWELL N., ROBERTSON S., ZARAGOZA H., TAYLOR M., « Relevance weighting for query independent evidence », *SIGIR '05 : Proceedings of the 28th annual international ACM SIGIR conference*, 2005.

[FRE 98] FREUND Y., IYER R., SCHAPIRE R. E., SINGER Y., « An efficient boosting algorithm for combining preferences », *Proceedings of ICML-98, 15th International Conference on Machine Learning*, 1998.

[FUH 05] FUHR N., LALMAS M., MALIK S., SZLÁVIK Z., Eds., *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004*, vol. 3493 de *Lecture Notes in Computer Science*, Springer, 2005.

[JäR 02] JÄRVELIN K., KEKÄLÄINEN J., « Cumulated gain-based evaluation of IR techniques », *ACM Trans. Inf. Syst.*, , 2002, p. 422–446, ACM Press.

[KAZ 01] KAZAI G., LALMAS M., RÖLLEKE T., « A Model for the Representation and Focussed Retrieval of Structured Documents Based on Fuzzy Aggregation. », *SPIRE*, 2001.

[KAZ 05] KAZAI G., LALMAS M., « INEX 2005 Evaluation Metrics », , 2005.

[LAL 97] LALMAS M., « Dempster-Shafer's Theory of Evidence Applied to Structured Documents : Modelling Uncertainty », 1997.

[LAL 00] LALMAS M., MOUTOGIANNI E., « A Dempster-Shafer indexing for the focussed retrieval of hierarchically structured documents : Implementation and experiments on a web museum collection », 2000, RIAO, Paris, France.

[PIW 04] PIWOWARSKI B., GALLINARI P., « A Bayesian Network for XML Information Retrieval : Searching and Learning with the INEX Collection », *Information Retrieval*, , 2004.

[ROB 92] ROBERTSON S. E., WALKER S., HANCOCK-BEAULIEU M., GULL A., LAU M., « Okapi at TREC », *Text REtrieval Conference*, 1992, p. 21-30.