Approche multicritère de recherche d'information structurée basée sur l'apprentissage d'ordonnancement

Messaoud CHAA, Omar NOUALI

CERIST
Centre de recherche sur l'information scientifique et technique
05, Rue des 3 fréres aissou - Ben Aknoun - Alger
{mchaa, onouali }@cerist.dz

RÉSUMÉ. *Il est connu que, dans la recherche d'information, des meilleures performances sont obtenues lorsque plusieurs sources de pertinence sont combinées en utilisant des méthodes d'apprentissage d'ordonnancement. Dans cet article, nous proposons une approche multicritère pour la recherche d'information dans les documents structurées basée sur les méthodes d'apprentissage d'ordonnancement pour apprendre automatiquement un modèle de Ranking et mesurer l'impact de chaque source de pertinence. Des expérimentations sur une grande collection de la compagne d'évaluation de la recherche d'information XML (INEX) ont montré la performance de notre approche.*

ABSTRACT. *It's known that, in information retrieval, best performances are obtained when many sources of evidence are combined using learning to rank methods. In this paper, we propose a multiple criteria approach for XML information retrieval based on learning to rank methods to automatically learn a ranking model and measure the impact of each source of evidence. Experiments on a large collection from the XML Information Retrieval evaluation campaign (INEX) showed good performance of the approach.*

MOTS-CLÉS : XML, Recherche d'information structurée, Apprentissage d'ordonnancement, Ranking SVM, BM25.

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

1. Introduction

L'adoption accrue de XML comme format standard pour représenter les documents structurés nécessite le développement des systèmes efficaces pour la recherche d'information structurée (RIS), dont le but est de retrouver les éléments XML pertinents (paragraphe, section, sous section,…etc.) et non pas des documents entiers. Ces éléments sont ensuite présentés ordonnés en fonction de leur pertinence par rapport à la requête. L'ordre de cette liste est primordial puisqu'en général les utilisateurs s'intéressent surtout aux premiers résultats retournés pour leur requête. Le problème donc, est de savoir définir une fonction de score capable de positionner les éléments les plus pertinents au début de la liste de résultats. Pour cela, plusieurs fonctions de calcul de score de pertinence ont été mises au point.

La majorité des approches proposées dans la littérature sont des adaptations des modèles de recherche d'information classique au contexte XML (Modèle vectoriel, probabiliste et modèle de langue, etc.) (Fuhr et al., 2004). D'autres approches spécifiques à la recherche d'information structurée ont été aussi développées pour restituer une liste ordonnée d'éléments XML pertinents XFIRM GPX (Sauvagnat et al., 2006), (Geva., 2006). Quelle que soit l'approche utilisée, la pertinence des éléments XML est réduite à une fonction de score qui utilise plusieurs critères (sources de pertinence) comme : le nombre d'éléments feuilles d'un élément (Sauvagnat et al., 2006), sa taille (Kamps et al., 2005), le score des éléments feuilles, le contexte de l'élément (Sauvagnat et al., 2006), (Vittaut et al., 2006). Plusieurs paramètres comme α, β, λ ... ont été utilisés, qui indiquent l'importance accordée à chaque source. Les valeurs finales de ces paramètres, sont fixées manuellement de manière empirique après plusieurs expérimentations comme c'est souvent le cas dans le domaine de la RI classique.

Afin d'améliorer la performance du modèle, d'autres sources de pertinence peuvent être ajoutés. Bien évidement, avec le nombre croissant des sources de pertinence à prendre en compte, l'approche manuelle pour fixer les paramètres est devenue difficile à mettre en œuvre.

Ces dernières années, un nouveau domaine de recherche appelé apprentissage d'ordonnancement « Learning to Rank », a progressivement émergé, qui utilise les techniques d'apprentissage automatique pour entraîner le modèle de Ranking en évitant le paramétrage manuel.

Les algorithmes d'ordonnancement ont été utilisés avec succès dans des tâches comme le filtrage collaboratif (Harrington, 2003), la RI multimédia (Yang et al., 2008), la questions-réponses (Xu et al., 2005). Néanmoins, peu de travaux concernant l'utilisation de ces méthodes dans la tâche de la recherche d'information structurée. Dans (Vittaut et al., 2006) les auteurs ont proposé d'utiliser les caractéristiques provenant de l'élément XML, de son document et de son élément parent puis ils utilisent une méthode d'apprentissage d'ordonnancement pairwise pour combiner les trois caractéristiques. Dans (Gao et al., 2009) une méthode d'apprentissage d'ordonnancement Listwise appelée ListBM a été utilisé pour apprendre les paramètres k1 et b de la fonction OKAPI BM25 (Robertson, 1997).

Dans cet article, nous proposons une approche basée sur l'apprentissage d'ordonnancement, dont certaines sources de pertinence sont utilisées et combinées de différentes façons afin de trouver la meilleure combinaison et mesurer l'impact de chaque source sur la performance du modèle.

2. Apprentissage d'ordonnancement en RI

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

Plusieurs algorithmes ont été développés ces dernières années dans le cadre de l'apprentissage d'ordonnancement. Chaque algorithme utilise une technique d'apprentissage et une fonction d'erreur différente. Dans (Liu, 2011) l'auteur a catégorisé ces algorithmes d'ordonnancement en trois grandes approches selon la façon de considérer les documents en entrée du système d'apprentissage.

– Approche Pointwise : chaque document est considéré séparément en entrée du système d'apprentissage, l'espace de sortie est le degré de pertinence du document. Le problème d'apprentissage est généralement considéré comme un problème de régression linéaire, classification ou de régression ordinale (Nallapati, 2004), (Li, 2008).

– Approche Pairwise : des paires de documents (Xi,Xj) sont considérées en entrée du système d'apprentissage. L'espace de sortie est un jugement de préférence (Yi,j) à valeur dans {-1,1}. Si Yi,j =+1 , alors le document Xi est préféré au document Xj : donc Xi doit être classé au dessus de Xj dans la liste de résultat. Le problème d'apprentissage ici est un problème de classification. Par conséquent, la plupart des algorithmes de cette approche utilisent les méthodes de classifications existantes. Plusieurs algorithmes ont été développés dans cette approche, à savoir Ranking SVM (Joachims, 2006), Rankboost (Freund, 2003) et LambdaMART (Wu, 2010).

– Approche Listwise : les algorithmes considèrent en entrée du système d'apprentissage la liste complète des documents et en sortie la liste ordonnée des documents. Les algorithmes de cette approche sont répartis en deux sous-catégories : ceux maximisant une mesure de RI comme la MAP ou le NDCG et ceux minimisant d'autres fonctions de perte non liées aux mesures de RI (par exemple, la distribution de probabilité sur les permutations). On peut citer à titre d'exemple l'algorithme ListNet (Cao, 2007) ou encore ListMLE (Xia, 2008).

3. Approche proposée

Notre approche comprend trois étapes principales

– Préparation de la base d'apprentissage : Cette étape concerne l'échantillonnage des éléments et des Requêtes d'apprentissage, l'extraction des caractéristiques que nous allons utiliser, puis l'étiquetage des couples élément-requête.

– Apprentissage : dans cette étape un algorithme d'apprentissage d'ordonnancement sera utilisé pour apprendre des modèles pour différentes combinaisons des caractéristiques afin de mesurer l'impact de chacune.

– Test et évaluation : dans cette étape nous allons utiliser les mesures d'évaluation standard de la RIS pour évaluer la performance du modèle appris pour chaque combinaison.

3.1. *Algorithme utilisé*

RankingSVM (Joachims, 2006), est l'un des algorithmes de référence pour l'apprentissage des fonctions d'ordonnancement. Nous avons choisi cet algorithme puisque son évaluation sur de nombreux jeux de données montre que sa performance est excellente (Liu, 2011). Le principe de l'algorithme est de transformer le problème d'ordonnancement en classification par paire (approche Pairwise). Pour classer les paires positives dans la classe +1 et les paires négatives dans la classe -1, l'algorithme RankingSVM minimise la fonction objectif suivante :

$$\begin{aligned} & \min \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n} \sum_{y_{u,v}^{(i)}} \xi_{u,v}^{(i)} \\ s.t \quad & W^T(x_u^{(i)} - x_v^{(i)}) \geq 1 - \xi_{u,v}^{(i)}, if\ y_{u,v}^{(i)} = 1, \\ & \xi_{u,v}^{(i)} \geq 0, i = 1,...,n. \end{aligned} \qquad [1]$$

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

3.1. *Caractéristiques utilisées*

Le choix des caractéristiques utilisées dans notre approche repose sur les quatre intuitions suivantes :

Contexte de l'élément (score du document) : un élément d'un document pertinent qui ne contient pas tous les termes de la requête, est susceptible d'être plus pertinent que s'il est contenu dans un document non pertinent.

Score de l'élément : plus un élément contient d'occurrences d'un terme de la requête, plus il est pertinent par rapport à cette requête.

Proximité des termes de la requête : plus les termes de la requête se retrouvent proches dans un élément, plus cet élément est pertinent et doit être positionné en tête de la liste des résultats.

Taille de l'élément : c'est à dire le nombre de termes qu'il contient est un paramètre très important dans le calcul de la pertinence des éléments, mais le problème est de savoir comment introduire ce paramètre.

Chaque intuition est modélisée par une fonction. Nous détaillons par la suite les quatre fonctions utilisées : *BM25Doc, BM25Elem, ProxElemt* et *SizeElem.*

3.1.1. *Score de l'élément (BM25Doc)*

Dans la recherche d'information, BM25 (Robertson, 1997) est la plus citées fonction de score utilisée pour estimer la pertinence des documents pour une requête donnée. Elle est basée sur le modèle probabiliste. Dans notre approche, cette fonction est utilisée pour estimer le score des documents à une requête donnée.

$$BM25Doc(d,q) = \sum_{t \in q} idf(t) \frac{(k1+1).tf(t,d)}{k1.((1-b)+b\frac{len(d)}{avdl})+tf(t,d)}, \quad [2]$$

$$idf(t) = \log \frac{|D|}{df(t)}, \quad [3]$$

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

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

3.1.2. *Score de l'élément (BM25Elem)*

Quoique, à l'origine, la fonction BM25 est utilisée pour calculer le score des documents dans la RI des documents plats. Elle a été introduite dans la RI XML ces dernières années et les modèles qui l'utilisent comme une fonction de base fonctionnent le mieux dans INEX (Itakura, 2008), (Geva, 2010) (Gao et al., 2011). De ce fait, nous allons utiliser la fonction BM25 pour calculer les scores des éléments dans notre approche.

$$BM25Elem(e,q) = \sum_{t \in q} ief(t) \frac{(k1+1).tf(t,e)}{k1.((1-b)+b\frac{len(e)}{avel})+tf(t,e)}, \quad [4]$$

$$ief(t) = \log \frac{|E|}{ef(t)} \quad [5]$$

Où $|E|$ le nombre d'éléments dans la collection; $ef(t)$ est le nombre d'éléments contenant le terme t (pas tous les éléments mais un sous ensemble d'éléments comme utilisé dans (Lu et al., 2005)); $tf(t,e)$ est la fréquence de terme *t* dans l'élément *e*; $len(e)$ désigne la longueur de l'élément et *avel* est la longueur moyenne des éléments.

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

3.1.3. *Proximité des termes de la requête (ProxElem)*

La distance entre les termes de la requête a été utilisée au contexte de la RI structurée de plusieurs façons (Abbaci et al., 2008), (Broschart et al., 2008). Dans notre approche, nous calculons la distance entre deux mots-clés de la requête ti et tj dans un élément *e* comme la distance minimale entre toutes les occurrences de *ti* et *tj* dans l'élément *e*.

$$Dis(e,t_i,t_j) = \min_{t_i \in e, t_j \in e} dist(e,t_i,t_j), \quad [6]$$

$$\Pr oxElem(e,t_i,t_j) = \frac{1}{Dis(e,t_i,t_j)}, \quad [7]$$

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

$$\Pr oxElem(e,q) = \sum_{i=1..n, j=1...n, i \neq j} \frac{1}{Dist(e,t_i,t_j)} \quad [8]$$

3.1.3. *Taille de l'élément (SizeElem)*

Pour tenir compte de la grande différence des tailles d'éléments dans les documents XML, et en particulier le nombre importants des petits éléments (éléments feuilles) par rapport aux éléments de taille plus grande (articles ou éléments intermédiaires) dans les collections des documents structurés. Plusieurs méthodes ont été adoptées dans la RIS pour résoudre ce problème. Dans (Kamps et al., 2005) une préférence en faveur des éléments de grande taille a été établie et un seuil a été défini pour la suppression des éléments de petite taille de l'index. Dans notre approche nous avons proposé un nouveau score pour la taille de l'élément (*SizeElem*) qui est basé sur les deux propriétés suivantes :

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

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

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

$$SizeElem = \begin{cases} \dfrac{len(e)}{IdealLenght} & if\ \ len(e) \le IdealLenght \\ \dfrac{len(e) - (MaxLenght + 1)}{IdealLenght - (MaxLenght + 1)} & if\ \ len(e) \ge IdealLenght \end{cases} \quad [9]$$

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

Dans les expérimentations, nous avons mis *IdealLenght =100 termes*.

4. Expérimentations et évaluation

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

4.1. *Données expérimentales*

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

4.1.1. *Echantillonnage des requêtes d'apprentissage*

Les requêtes utilisées dans notre approche sont les 29 requêtes de type CO (Content Only), qui ont des jugements de pertinence, numérotées de 202 à 241.

Afin de rendre l'évaluation plus complète, nous avons utilisé dans notre expérience le « k-fold cross validation » de la validation croisée, avec k=3. L'ensemble de requête est divisé en trois parties avec environ le même nombre de requêtes, notées S1, S2 et S3. Pour chaque étape, deux parties sont utilisées pour l'apprentissage du modèle d'ordonnancement et la partie restante pour évaluer la performance du modèle appris (voir tableau 1). La moyenne des résultats des trois différentes expériences de la validation croisée est finalement utilisée comme résultat de l'évaluation de notre approche.

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

Tableau 1. *Partitionnement des requêtes pour trois-fold cross validation.*

4.1.2. *Echantillonnage des éléments d'apprentissage*

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

4.1.3. *Etiquetage des couples élément-requête*

L'objectif de l'étiquetage de données est d'associer à chaque couple élément-requête une étiquette, qui décrit le niveau de pertinence de l'élément par rapport à la requête. Dans la collection que nous allons utiliser pour l'apprentissage et le test, les jugements de pertinence sont définis par des experts humains (participants de la compagne INEX), suivant deux dimensions : l'exhaustivité (e) et la spécificité (s) (Chiaramella et al. 1996). Pour assigner chaque élément avec une étiquette nous avons utilisé la fonction d'agrégation "généralisée" $f(e,s)= s*e$. Nous avons proposé d'utiliser quatre niveaux de jugement de pertinence pour associer aux éléments qui ont des jugements de pertinence très proches la même étiquette, comme montre le tableau suivant.

$f(e,s)= s*e$	Etiquette	Degré de pertinence
$1.5<=f(e,s)<=2$	3	Excellent
$1<=f(e,s)<1.5$	2	Bon
$0<f(e,s)<1$	1	Moyen
$f(e,s)=0$	0	Mauvais

Tableau 2. *Etiquetage des couples élément-requêtes.*

4.2. Evaluation des résultats

L'évaluation de notre approche se fait à travers deux mesures : la courbe d'effort-précision par rapport au gain-rappel et le gain cumulé étendu normalisé

(nxCG). Ce dernier est le gain cumulé par rapport à un seuil. L'évaluation des résultats a été faite par l'outil Evalj[1].

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

4.2.1. *Combinaison de toutes les caractéristiques*

La figure 1 montre les résultats obtenus sur la mesure Maep de notre approche en combinant toutes les caractéristiques *(BM25Elem, BM25Doc, ProxElem, et SizeElem)*. Ces résultats sont comparés avec toutes les expérimentations officielles relatives aux requêtes de type CO, soumises par l'ensemble des participants à INEX 2005, pour la tache *thorough*. On peut voir par cette mesure que les performances de notre approche sont meilleures que celles des autres expérimentations, en particulier, au début de la liste de classement. Un bon système de recherche d'information structurée positionne les éléments les plus pertinents au début de la liste.

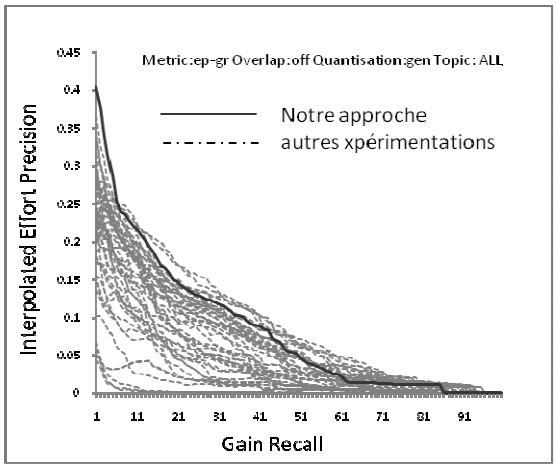

Figure 1. *Courbe de ep/gr de notre approche et des résultats officiels de la campagne d'évaluation INEX 2005, tâche CO, quantification fgen.*

Le tableau 3 montre les résultats obtenus sur la mesure ManXCG (Mean average normalized extended cumulated gain) de notre approche, BM25 (modèle de base) et les expérimentations officielles de l'université de Kaiserslautern et du laboratoire informatique de Paris 6, Lip6 (Vittaut et al., 2006), qui sont les meilleures expérimentations soumises à la compagne INEX 2005, pour la tâche *thorough*. Pour cette mesure, notre approche montre des améliorations significatives.

4.2.2. *Etude de l'impact de chaque caractéristique*

Pour mesurer l'impact de chaque caractéristique sur la performance de notre approche. Nous avons évalué les résultats obtenus en enlevant à chaque fois une caractéristique et en combinant les trois restantes. Le tableau 4 montre les résultats trouvés pour chaque combinaison.

[1] Projet JAVA dans lequel toutes les mesures utilisées durant INEX 2005 ont été implémentées. Peut être téléchargé à partir de l'adresse : https://sourceforge.net/projects/evalj

Run Name	ManXCG[r]					
	1	***5***	***10***	***15***	***25***	***50***
Notre approche	**0.3895**	**0.3439**	**0.3317**	**0.3228**	**0.3104**	**0.2899**
Univ de Kaiserslautern	0.3073	0.2345	0.3124	0.3083	0.2986	0.2772
Lip6	0.3845	0.3156	0.2874	0.2755	0.2624	0.2425
BM25(Modèle de Base)	0.2737	0.2372	0.2286	0.2229	0.2187	0.2128

Tableau 3. *Evaluation des résultats de notre Approche, BM25 et les deux meilleures expérimentations soumises à la compagne INEX 2005 sur la mesure MAnxCG*

Caractéristique enlevée	MAnxCG[r]					
	1	5	10	15	50	100
BM25Elem	0.2239	0.2457	0.2352	0.2273	0.2188	0.2106
	-42.52%	**-28.55%**	**-29.09%**	**-29.58%**	**-29.51%**	**-27.35%**
SizeElem	0.22	0.2552	0.2632	0.2653	0.2701	0.2665
	-43.52%	-25.79%	-20.65%	-17.81%	-12.98%	-8.07%
ProxElem	0.2865	0.2986	0.2922	0.2868	0.282	0.2658
	-26.44%	-13.17%	-11.91%	-11.15%	-9.15%	-8.31%
BM25Doc	0.3248	0.3181	0.3151	0.3099	0.3034	0.2799
	-16.61%	-7.50%	-5.00%	-4.00%	-2.26%	-3.45%
Aucun	**0.3895**	**0.3439**	**0.3317**	**0.3228**	**0.3104**	**0.2899**

Tableau 4. *Résultats montrent l'impact de chaque caractéristique.*

D'après les résultats, nous constatons que la meilleure performance de l'approche est obtenue en combinant toutes les caractéristiques. Les résultats montrent clairement que la suppression du BM25Elem ou SizeElem donne des mauvais résultats (par exemple MAnxCG@1 a diminué de 0,3895 à 0,2239 (-42.52%) lorsqu'on supprime BM25Elem et à 0,2200 (-43.52%) lorsqu'on supprime SizeElem). Alors l'impact de ces deux caractéristiques est très élevé par rapport à celui de BM25Doc et ProxElem. Il apparaît également que l'impact de BM25Elem est très fort sur tous les résultats trouvés MAnxCG@1~100 alors que l'impact de SizeElem est très fort sur les cinq premiers résultats uniquement MAnxCG@1~5 mais l'impact est moyennement fort pour MAnxCG@10~100 (-8.07% uniquement pour MAnxCG@100). Il est clair aussi que l'impact de ProxElem est important comparativement à l'impact de BM25Doc.

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

5. Conclusion

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

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

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

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

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

Nous souhaitons par la suite, étendre l'évaluation de notre approche sur une grande collection, telle que la collection qui contient plus de 650000 articles de l'encyclopédie Wikipedia (Denoyer et al., 2006), utiliser d'autres caractéristiques liées à la structure et le contenu des éléments XML qui peuvent améliorer le mieux la RIS.

6. Bibliographie

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

Broschart, A., Schenkel, R., Theobald, M., « Experiments with Proximity-Aware Scoring for XML Retrieval at INEX 2008», INEX(2008), p. 29-32.

Cao, Z., Qin, T., Liu, T., Tsai, M., Li, H., « Learning to rank: from pairwise approach to listwise approach », ICML2007, p. 129-136

Chiaramella, y., Mulhem, p., and Fourel, f., A model for multimedia information retrieval, tech. rep., university of glasgow, 1996.

Denoyer, L., Gallinari, P., « The Wikipedia XML corpus », SIGIR Forum(2006), p. 64-69.

Freund, Y., Iyer, R.D., Schapire, R.E., Singer, Y., « An efficient boosting algorithm for combining preferences », Journal of Machine Learning Research, vol. 4, 2003, p.933–969.

Fuhr, N., Lalmas, M., «Report on the INEX 2003 workshop», SIGIR Forum(2004), p. 46-51.

Gao, N., Deng, Z., Xiang, Y., Yu, H., «ListBM: A Learning-to-Rank Method for XML Keyword Search», INEX 2009, p. 81-87.

Gao, N., Deng, Z., Yu, H., Jiang, J., «ListOPT: Learning to Optimize for XML Ranking», PAKDD (2)'11, p. 482-492.

Geva, S., Kamps, J., Lethonen, M., Schenkel, R., Thom, J.A., Trotman, A. « Overview of the INEX 2009 Ad Hoc Track », In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, p. 4–25. Springer, Heidelberg (2010).

Geva, S., «GPX - Gardens Point XML IR at INEX 2006», INEX'06, p. 137-150.

Harrington, E.F., « Online ranking/collaborative filtering using the perceptron algorithm ». *20th International Conference on Machine Learning ICML 2003*, p. 250–257.

Itakura, K.Y., Clarke, C.L.A., « University of Waterloo at INEX 2008: Adhoc, Book, and Link-the-Wiki Tracks», INEX 2008, p. 132-139.

Joachims, T., « Training linear SVMs in linear time », KDD 2006, p. 217-226.

Kamps, J., Rijke, M.D., Sigurbjörnsson, B., « The Importance of Length Normalization for XML Retrieval », Inf. Retr. 2005, p. 631-654.

Li, P., Burges, C., Wu, Q., « McRank: Learning to rank using multiple classification and gradient boosting », *Advances in Neural Information Processing Systems 20, ed. J. Platt, D. Koller, Y. Singer, and S. Roweis*, p. 897– 904, MIT Press, Cambridge, MA, 2008.

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

Nallapati, R., « Discriminative models for information retrieval », *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, p. 64–71.

Robertson, S. E, « Overview of the okapi projects, Journal of. Documentation », Vol. 53, No. 1, 1997, p. 3-7.

Sauvagnat, K., Hlaoua, L., Boughanem, M, « XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks », INEX 2005 , p. 88-103.

Liu.T.Y Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3):225–331, 2009.

Vittaut, J., Gallinari, P., « Machine Learning Ranking for Structured Information Retrieval. », ECIR 2006, p. 338-349.

Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J., « Adapting boosting for information retrieval measures », Inf. Retr. 2010, p. 254-270.

Xia, F., Liu, T., Wang, J., Zhang, W., Li, H., « Listwise approach to learning to rank: theory and algorithm. », ICML 2008, p. 1192-1199.

Xu, J., Cao, Y., Li, H., Zhao, M, « Ranking definitions with supervised learning methods », *Proceedings of the 14th International Conference onWorldWideWeb(WWW2005)*, p. 811–819.

Yang, Y.H., Hsu, W.H., « Video search reranking via online ordinal reranking », *Proceedings of IEEE 2008 International Conference on Multimedia and Expo (ICME 2008)*, p. 285– 88.