MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Adaptation de RankBoost pour l'Ordonnancement Semi-supervisé

Faïza Dammak [1], Hager Kammoun [1] et Abdelmajid Ben Hamedou[1]

1 : Laboratoire MIRACL – ISIMS SFAX

Contact : faiza.dammak@gmail.com
hager.kammoun@isd.rnu.tn
abdelmajid.benhamadou@isimsf.rnu.tn

Résumé
Cet article présente un algorithme d'ordonnancement, pour une recherche d'information (RI) efficace, basé sur un apprentissage semi-supervisé utilisant des données partiellement étiquetées. L'algorithme proposé est à caractère inductif puisqu'il est capable d'inférer un ordre sur des nouveaux exemples qui ne sont pas utilisés dans la phase d'apprentissage. Nous avons choisi d'adapter l'algorithme d'apprentissage de fonction d'ordonnancement supervisé RankBoost sur des données partiellement étiquetées. Les données non-étiquetées vont être d'abord étiquetées par une méthode transductive.

Abstract
This paper presents a ranking algorithm, for an effective information retrieval (IR), based semi-supervised machine learning, using partially labelled data. The proposed algorithm has an inductive character since it is able to infer an ordering on new examples that were not used for its training. We chose to adapt the supervised RankBoost algorithm of ranking function on partially labelled data. The unlabeled data will be initially labelled by a transductive method.

Mots-clés : Algorithmes/fonctions d'ordonnancement, apprentissage semi-supervisé.

Keywords: Ranking Algorithms/Functions, Semi-supervised machine learning.

1. Introduction

Avec le développement des technologies d'information, il est devenu nécessaire de concevoir de nouveaux cadres d'apprentissage automatique pour la recherche de données pertinentes par rapport à une requête donnée.
La communauté d'apprentissage a formulé cette problématique à travers le paradigme d'apprentissage supervisé de fonctions d'ordonnancement [1]. Dans le cas de la recherche documentaire (RD), il s'agit d'apprendre une correspondance entre un ensemble de requêtes et un ensemble de documents étiquetés (appelé ensemble des alternatives), capable d'ordonner ces derniers par rapport à une requête donnée. Ce problème correspond à l'ordonnancement d'alternatives [2].
Le principal inconvénient de ce paradigme est que la constitution d'une base étiquetée dans ce cas nécessite l'étiquetage de l'ensemble des documents entrants, ce qui nécessite l'intervention d'un expert qui doit examiner manuellement une grande quantité de données. L'intérêt s'est

donc porté, dans le cadre de l'ordonnancement, au problème d'apprentissage semi-supervisé qui prend en compte des données étiquetées et non-étiquetées [1, 3, 4, 5].

Récemment, plusieurs travaux se sont intéressés aux algorithmes d'ordonnancement dans un cadre transductif [6, 7, 8]. Ces algorithmes prennent en considération une structure de graphe dont les sommets représentent les données étiquetées et non-étiquetées et dont le but est de propager l'information des exemples étiquetés sur l'ensemble du graphe. Une fois cette fonction apprise, les exemples non-étiquetés de la base sont ordonnés en utilisant les valeurs de scores ainsi obtenues. Cependant, pour chaque nouvel ensemble, la fonction de score recherchée doit être à nouveau re-estimée. Ces méthodes bien qu'elles ont montré leur efficacité par rapport à des méthodes supervisées pures, ne sont cependant pas adaptées au cas où on cherche à inférer des scores aux nouveaux exemples absents de la phase d'apprentissage. Ces méthodes sont dites transductives par opposition aux méthodes inductives, qui sont capables d'ordonner d'autres exemples que ceux qui ont été utilisés pour apprendre.

Nous proposons dans ce papier un nouvel algorithme d'ordonnancement d'alternatives semi-supervisé inductif, capable d'apprendre une fonction d'ordonnancement, qui va assigner un score plus élevé aux exemples pertinents que ceux non-pertinents. L'idée que nous proposons est d'adapter l'algorithme d'ordonnancement supervisé inductif RankBoost [9] sur des données partiellement étiquetées.

Pour montrer de façon empirique que les données non-étiquetées peuvent être utiles pour l'ordonnancement d'alternatives semi-supervisé, nous proposons de comparer l'algorithme proposé avec l'algorithme d'ordonnancement supervisé RankBoost [9]. Pour évaluer la méthode proposée, nous allons nous baser sur trois mesures couramment utilisées dans la communauté de recherche d'information (RI) : la précision à une position n (P@n), la précision moyenne (MAP) et le gain cumulatif normalisé d'escompte (NDCG). Les évaluations vont être effectuées sur la collection de données OHSUMED issue du benchmark LETOR [10].

Dans la section 2, nous introduisons le principe d'ordonnancement ainsi que son intérêt dans la RI. Nous détaillons également l'algorithme RankBoost et le principe d'ordonnancement semi-supervisé. Dans la section 3, nous proposons un algorithme d'ordonnancement inductif pour une RD qui induit une fonction d'ordonnancement à base d'apprentissage semi-supervisé, en adoptant l'algorithme RankBoost et l'adaptant sur des données partiellement étiquetées. Dans la section 4, nous présentons la collection de données OHSUMED ainsi que les mesures d'évaluation utilisées pour évaluer l'algorithme proposé. En conclusion, nous présentons le travail à suivre.

2. Ordonnancement et Recherche d'Information

L'apprentissage de fonctions d'ordonnancement correspond à apprendre des fonctions qui ordonnent des éléments entre eux, plutôt que de les classer [9]. Cette problématique a récemment suscité de nombreux travaux et elle est, en particulier, très adaptée aux problèmes de RI, où les documents d'une collection doivent être ordonnés entre eux par rapport à une requête utilisateur. Dans ce cas, il s'agit de calculer un score de pertinence (par rapport à la requête) pour chaque document de la collection, puis d'ordonner les documents par ordre de scores décroissants. L'objectif est alors que les documents pertinents pour la requête obtiennent de meilleurs scores par rapport aux documents non-pertinents.

D'un point de vue théorique, l'apprentissage de fonctions d'ordonnancement dans le cadre de la classification ou de régression habituel transgresse l'hypothèse fondamentale en apprentissage supervisé qui est que les observations (requêtes) sont échantillonnées indépendamment selon une distribution fixe et inconnue. En effet, les paires d'observations constituées à partir de variables aléatoires indépendantes ne sont plus indépendantes entre elles [1, 2].

L'ordonnancement est donc un problème nouveau et important en apprentissage. Afin d'apprendre des fonctions d'ordonnancement performantes, il faut optimiser une erreur d'ordonnancement, et non de classification.

Plusieurs travaux se sont intéressés à l'apprentissage de fonctions d'ordonnancement supervisé en utilisant des algorithmes de classification binaire comme base de développement de leurs algorithmes [11, 12, 13]. D'autres algorithmes sont des adaptations du perceptron : [14], Rank-

Net [15], ou de l'algorithme AdaBoost : RankBoost [9], AdaRank [16], ou de l'algorithme des machines à vecteurs de support (SVM) : RankSVM [17].
L'ordonnancement apparaît principalement dans le domaine de la RI. Par exemple, les SR documentaire (SRD) prennent en entrée une requête utilisateur et renvoient une liste ordonnée de documents étiquetés qui doit présenter en premier les documents pertinents pour la requête. Dans ce cas, un ensemble fixé (appelé ensemble des alternatives) doit être ordonné en fonction d'une entrée donnée (une requête). Ce type d'ordonnancement est connu sous le nom d'ordonnancement d'alternatives. Formellement, cela revient à déterminer un sous-ensemble de la collection initiale des documents en rapport avec une requête donnée. Si X l'ensemble des observations de la base d'apprentissage et Y l'ensemble des sorties réelles, pour chaque observation $x_i \in$ X est associée un vecteur désiré de taille variable m_i, $y_i = (y_i^1, ..., y_i^{m_i})$, avec $y_i^k \in IR$, l'ensemble de ses alternatives candidates. Ce vecteur définit l'ordre que l'on cherche à prédire sur les alternatives dans Y. La fonction de score F que doit prédire cet ordre prend en entrée un couple (x_i, k) où x_i est une observation et k un indice d'alternative candidate pour x_i, et renvoie un score réel reflétant la similarité entre une observation et une alternative i.e. F : X × Y → IR.
L'ordonnancement d'instances est un autre type d'ordonnancement issu de la RI, où il s'agit d'ordonner les exemples (où instances) d'une collection donnée de façon à ce que les exemples jugés pertinents soient ordonnés au-dessus des exemples non-pertinents.
Dans la suite, nous présentons l'algorithme d'ordonnancement supervisé inductif RankBoost ainsi que le principe d'ordonnancement semi-supervisé.

2.1. L'algorithme RankBoost

RankBoost est l'un des premiers algorithmes d'ordonnancement d'instances supervisé, son but est d'estimer une fonction de score f_t pour chaque document [9]. Il est conçu pour des problèmes d'ordonnancement. Comme tous les algorithmes de la famille Boosting, RankBoost apprend à chaque itération une fonction de base (fonction de score) $\{f_t\}$ et les poids $\{\alpha_t\}$ avec $t \in \{1..T\}$, et il construit itérativement une combinaison linéaire de ces fonctions, en adaptant à chaque itération une distribution de probabilité D_t sur l'ensemble des paires composées d'exemples (pertinent, non-pertinent) nommées paires cruciales. La sortie de cet algorithme est la fonction de score finale F qui est définie par :

$$F = \sum_{t=1}^{T} \alpha_t f_t \quad \text{où } T \text{ est le nombre d'itérations maximal.}$$

L'idée nouvelle introduite par RankBoost est qu'il propose un algorithme de sélection des fonctions de base efficace lorsque le nombre de ces fonctions est fini. Chaque fonction de base f_t est uniquement définie par une fonction caractéristique d'entrée φ_{jt} avec $j_t \in \{1...d\}$ et un seuil θ_t :

$$f_t(x) = \begin{cases} 1, \textit{if } \varphi_{jt}(x) > \theta_t \\ 0, \textit{si non} \end{cases} \quad \text{où } \varphi_{jt}(x) \text{ est la j}^{\text{ème}} \text{ fonction caractéristique fournie pour x.}$$

2.2. Les algorithmes d'ordonnancement semi-supervisé

Le paradigme de l'apprentissage semi-supervisé a suscité un grand intérêt dans la communauté d'apprentissage. Cet intérêt est principalement motivé, par des applications en RI pour lesquelles il est généralement difficile de disposer d'une base d'exemples étiquetés. L'ensemble des algorithmes d'apprentissage semi-supervisé consistent à prendre en compte une base d'apprentissage formée d'une part d'un ensemble de données étiquetées, dont l'étiquette reflète un jugement de pertinence, et d'autre part d'un ensemble de données non-étiquetées [5]. Le but final de ces algorithmes est d'apprendre une fonction de score F à partir des deux ensembles de données étiquetées et non-étiquetées.
La plupart des algorithmes d'ordonnancement semi-supervisé sont des techniques transductives à base de graphe incorporant l'information de voisinage local, c'est-à-dire de graphe valué et non-orienté en connectant petit à petit les points les plus proches jusqu'à ce que le graphe

devienne connexe. Les nœuds sont constitués des exemples étiquetés et non-étiquetés de la base d'apprentissage et les poids reflètent la similarité entre les exemples voisins. Ce graphe est construit avec une méthode, telle que les K plus proches voisins, qui permet de trouver les étiquettes des exemples non-étiquetés en exploitant directement le graphe en propageant par exemple les étiquettes des données étiquetées à leurs voisins non-étiquetés. Il affecte ainsi un score pour chacune des instances, 1 pour les instances positives et 0 pour les autres. Les scores sont alors propagés à travers le graphe jusqu'à convergence. À la fin les scores obtenus permettent d'induire un ordre sur l'ensemble des instances non-étiquetées [8].

Les algorithmes d'ordonnancement semi-supervisé transductifs sont définis par les opérations suivantes :

- calculer les distances entre les instances de la base d'apprentissage,
- connecter les instances les plus proches par un arc jusqu'à ce que le graphe devienne connexe et
- propager les étiquettes des données étiquetées à leurs voisins non-étiquetés.

Ces algorithmes ne peuvent pas étiqueter les exemples absents de la phase d'apprentissage puisqu'ils ne font pas partie des nœuds du graphe. Cet inconvénient peut être évité en utilisant une approche inductive. Nous proposons donc de combiner une méthode supervisée (adaptation de Rankboost) et une méthode transductive à base de graphe. La méthode transductive sert à attribuer des étiquettes aux données non-étiquetées, puis une version étendue de l'algorithme RankBoost sera utilisée sur l'ensemble de données. Nous détaillons cette proposition dans la section suivante.

3. Proposition d'une méthode d'ordonnancement d'alternatives semi-supervisé

Nous préconisons une approche qui consiste à apprendre une fonction d'ordonnancement à partir de l'adaptation de l'algorithme RankBoost sur une base d'exemples étiquetés et une autre base d'exemples non-étiquetés. Les données non-étiquetées vont être d'abord étiquetées par la méthode transductive des **k** plus proches voisins décrite précédemment. L'originalité de notre approche est que nous proposons un algorithme d'apprentissage semi-supervisé, pour la tâche d'ordonnancement d'alternatives.

RankBoost est un algorithme d'ordonnancement supervisé à caractère inductif. En effet, il est capable d'ordonner une liste d'exemples non vus durant la phase d'apprentissage en inférant un ordre sur cette liste [9]. Les méthodes transductives quand à elles exploitent la structure des données pour ordonner ces exemples en se basant sur leur similarité par rapport aux exemples positifs. Pour profiter des avantages de chacune, nous proposons de combiner les deux méthodes (transductive et inductive supervisée).

Nous nous intéressons donc à :

- étiqueter les exemples non-étiquetés S' avec la méthode transductive.
- apprendre une fonction de score F retenue suite à une adaptation de l'algorithme Rankboost sur les exemples étiquetés S et les exemples S' étiquetés par la méthode transductive.

Nous disposons donc d'une base d'apprentissage étiquetée S constituée de m couples (observation, sortie réelle) S = {(x_1 , y_1), ..., (x_m , y_m)}, où chaque y_i est un vecteur de taille variable m_i et m_i désigne le nombre d'alternatives candidates pour x_i . Aussi, $y_i = (y_i^1 ,..., y_i^{m_i})$, avec $y_i^k \in IR$.

Nous disposons également d'une base d'apprentissage non-étiquetée S', cette base va être étiquetée par la méthode transductive et sera de la forme : $S' = \{(x_i , y_i^{'}); i \in \{m+1,..,m+n\}\}$. Nous notons alors la base d'apprentissage B l'union de ces deux bases et est définie par :

$B = S \cup S' = \{(x_i , y_i); i \in \{1,..,m+n\}\}$, pour chaque exemple x_i , il y a b_i alternatives candidates et p_i (resp. n_i) désigne le nombre d'alternatives pertinentes (non-pertinentes).

Si X l'ensemble des observations de la base B et Y l'ensemble des sorties réelles. $Y = Y_+ \cup Y_-$ où Y_+ et Y_- les ensembles respectifs des alternatives pertinentes et non-pertinentes de B. Le but est de trouver une fonction de score F en utilisant l'information contenue dans les deux bases S et S'.

Dans ce qui suit, nous détaillons le fonctionnement de l'algorithme RankBoost appliqué à notre contexte.

3.1. Adaptation de l'algorithme RankBoost à l'ordonnancement d'alternatives semi-supervisé

L'adaptation de RankBoost est donnée dans l'algorithme 1 : A chaque itération *t*, l'algorithme maintient une distribution λ_t sur les exemples de la base d'apprentissage B, une distribution v_t^i sur les alternatives associées à l'exemple x_i et une distribution D_t^i sur les paires cruciales d'alternatives, représentée par une distribution sur les couples (*k*, *l*) tels que $y_i^k \in Y_+$ et $y_i^l \in Y_-$ pour chaque exemple x_i. La distribution D_t^i est définie à base des deux autres :

$$\forall\, i \in \{1,..,\, m+n\}, \forall\, (k,\, l) \in \{1,..,\, b_i\}^2 \text{ tel que } y_i^k \in Y_+,\ y_i^l \in Y_- : D_t^i(k,\, l) = \lambda_t^i\ v_t^i(k)\ v_t^i(l)\,.$$

Ces distributions sont mises à jour grâce à la fonction de base f_t, sélectionnée à partir de l'algorithme de recherche des fonctions de base (Algorithme 2 qui constitue également une adaptation) qui renvoie la valeur du seuil θ_{res} résultante associée à chaque caractéristique et les valeurs possibles qui peuvent être associées à f_t, tel que :

$$f_t(x_i,k) = \begin{cases} 1 & si\ \varphi_j(x_i,k) > \theta_{res} \\ 0 & si\ \varphi_j(x_i,k) \le \theta_{res} \end{cases}$$ avec x_i est l'observation d'indice *i*, *k* est l'indice de l'alternative associée à cet x_i.

Le poids α_t est défini par :

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right) \text{ où } \quad r_t = \sum_{k,l} D_t^i(k,l)\left(f_t(x_i,k) - f_t(x_i,l)\right)$$

D'après [9] et [2], le critère d'apprentissage (ou erreur d'ordonnancement) a pour borne supérieure :

$$R_m^{OA}(F,S) \le \prod_t^T Z_t \text{ où } R_m^{OA}(F,S) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i p_i}\sum_{k:y_i^k \in Y_+}\ \sum_{l:y_i^k \in Y_-} [\![f(x_i,k) - f(x_i,l) \le 0]\!]$$

Ce critère d'apprentissage pour la fonction de score finale *F* sur l'ensemble d'apprentissage B [2] sera définie par :

$$R_m^{OA}(F, S \cup S') = \frac{1}{m+n}\sum_{i=1}^{m}\frac{1}{n_i p_i}\sum_{k:y_i^k \in Y_+}\ \sum_{l:y_i^k \in Y_-} [\![f(x_i,k) - f(x_i,l) \le 0]\!] = \sum_{i=1}^{m+n}\ \sum_{k,l:y_i^k > y_i^l} D_1^i(k,l)\, [\![f(x_i,k) \le f(x_i,l)]\!]$$

A chaque itération t, α_t est choisi de façon à minimiser Z_t. L'algorithme détaillé fait donc décroître itérativement la borne supérieure sur la fonction $R_m^{OA}(F, S \cup S')$.

Algorithme 1 : Algorithme RankBoost adapté à l'ordonnancement d'alternatives semi-supervisé

Entrées : Une base d'apprentissage $B = S \cup S' = \{(x_i\ ,\ y_i\ ); i \in \{1,..,m+n\}\}$, pour chaque exemple x_i, il y a b_i alternatives candidates

Initialisation : $\forall\ i \in \{1, ..., m+n\}, \quad \lambda_1^i = \frac{1}{m+n} \qquad v_1^i(k) = \begin{cases} \frac{1}{p_i}\ si\ y_i^k \in Y_+ \\ \frac{1}{n_i}\ si\ y_i^k \in Y_- \end{cases}$

1 Début
2 pour *t* :=1..*T* faire
3 Sélectionner la fonction de base f_t à partir de D_t
4 Calculer α_t en utilisant la formule définie ci-haut
5 $\forall\ i \in \{1, .., m+n\}, \forall\ (k, l) \in \{1, ..., b_i\}^2$ t.q. $y_i^k \in Y_+,\ y_i^l \in Y_-$,
6 mettre à jour $D_{t+1}^i(k, l)$: $D_{t+1}^i(k, l) = \lambda_{t+1}\ v_{t+1}^i(k)\ v_{t+1}^i(l)$
7 $\forall\ i \in \{1, .., m+n\}, \ \lambda_{t+1}^i = \frac{\lambda_t^i Z_t^{-1i} Z_t^{1i}}{Z_t}$

$$v_{t+1}^{i}(k)=\begin{cases}\dfrac{v_t^{i}(k)\exp(-\alpha_t f_t(x_i,k))}{Z_t^{1i}} \; si \; y_i^k \in Y_+\\ \dfrac{v_t^{i}(k)\exp(\alpha_t f_t(x_i,k))}{Z_t^{-1i}} \; si \; y_i^k \in Y_-\end{cases}$$

où Z_t^{1i}, Z_t^{-1i} et Z_t sont défini par: $Z_t^{1i} = \sum_{k:y_i^k \in Y_+} v_t^i(k)\exp(-\alpha_t f_t(x_i,k))$

$$Z_t^{-1i} = \sum_{l:y_i^l \in Y_-} v_t^i(l)\exp(\alpha_t f_t(x_i,l)),\; Z_t = \sum_{i=1}^{m} \lambda_t^i Z_t^{-1i} Z_t^{1i}$$

8 **Fin**

Sortie : La fonction de score finale $F = \sum_{t=1}^{T} \alpha_t f_t$

3.2. Adaptation de l'algorithme de sélection de fonctions de base au cas de l'ordonnancement d'alternatives

L'algorithme de sélection de fonctions de base (Algorithme 2) permet de trouver, avec une complexité linéaire au nombre d'alternatives, une fonction f_t qui minimise r_t dans un cas particulier où les fonctions f_t sont à valeurs dans {0, 1} et créées en seuillant des caractéristiques réelles associées aux exemples.

Supposons que chaque observation x_i possède un ensemble de caractéristiques fournies par des fonctions φ_j, j = 1...d. Pour chaque j, φ_j (x_i , k) est une valeur réelle. Ainsi, il s'agit d'utiliser un seuillage de la fonction caractéristique φ_j pour créer des valeurs binaires. L'ensemble des fonctions de base à combiner est créé en définissant a priori un ensemble de seuils. D'une façon générale, ces seuils dépendent de la caractéristique φ_j considérée.

Algorithme 2 : Algorithme de sélection de fonctions de base (adaptation)

Entrées : Une distribution $D_t^i(k, l) = \lambda_t^i v_t^i(k) v_t^i(l)$ sur la base B $\forall$ i $\in$ {1,…, $m + n$}, (k, l) $\in$ {1,…, b_i} tel que $y_i^k \in Y_+$ et $y_i^l \in Y_-$

Un ensemble de caractéristiques $\{\varphi_j(x_i,k)\}_{j=1}^{d}$

Pour chaque φ_j, un ensemble de seuils $\{\theta_q\}_{q=1}^{Q}$ tel que $\theta_1 > ... > \theta_q$

Initialisation : $\forall$ i $\in$ {1,…, m+n}, (k, l) $\in$ {1,…, b_i}, π (x_i , k) = $y_i^k \lambda_1^i v_1^i(k) \sum_{l:y_i^l \neq y_i^k} v_1^i(l)$

r* = 0

1 **début**
2 **pour** j :=1,…, d **faire**
3 L ← 0
4 **pour** q :=1,…, Q **faire**
5 L← L + $\sum_{i=1}^{m}\sum_{k:\varphi_j(x_i,k)} \pi(x_i,k) + \sum_{i=m+1}^{m+n}\sum_{k':\varphi_j(x_i',k')} \pi'(x_i',k')$
6 **si** $|L| > |r^*|$ **alors**
7 r*← L
8 j^*← j
9 θ^*← θ_q
10 **Fin**

Sortie : (φ_j *, θ*)

4. Préparation de la partie expérimentale

Pour valider notre approche, nous choisissons d'utiliser la collection de données OHSUMED issue du benchmark standard LETOR (*LEarning TO Rank*) [10] qui a été réalisé pour l'apprentissage de fonctions d'ordonnancement dans le domaine de la RI. LETOR additionne à OHSUMED une autre collection de données appelée .GOV (TREC2003 et TREC2004) (Text Retrieval Conference) et propose un ensemble de résultats issus de plusieurs algorithmes d'ordonnancement utilisant ces deux collections tels que RankBoost [9] et RankSVM [17]. Pour l'évaluation des résultats, nous choisissons de comparer notre algorithme avec RankBoost. Ce dernier a été implémenté et testé. L'implémentation de l'adaptation de cet algorithme avec les données non-étiquetées en utilisant la méthode transductive est en cours de finalisation.

Dans une première étape, l'algorithme proposé a été seulement testé sur l'ensemble des données OHSUMED. Des expériences sur la collection .GOV sont à mener dans une seconde étape.

OHSUMED définit des paires requête-document, chacune est constituée d'un vecteur de caractéristiques et d'un jugement de pertinence correspondant. Les jugements de pertinence sont à trois niveaux dans le sous-ensemble OHSUMED: {0, 1, 2} correspondant respectivement à « non pertinent », « partiellement pertinent », et « certainement pertinent ».

Nous choisissons la version courante de LETOR (version 3.0)[1], utilisant 45 caractéristiques extraites à partir de OHSUMED [19]. Nous nous servons, dans la partie expérimentale, de ces caractéristiques pour déterminer les valeurs des caractéristiques φ_j afin de compléter par la suite les valeurs des fonctions de base (section 3.2).

La stratégie de prélèvement utilisée pour la collection OHSUMED est celle adoptée pour les trois versions de LETOR (1.0, 2.0, 3.0) : Etant donnée une requête, seulement les documents jugés sont choisis pour l'extraction de caractéristiques. En adoptant une telle stratégie d'échantillonnage, une requête a environ 152 documents en moyenne pour l'extraction de caractéristiques [18].

Chaque sous-ensemble de OHSUMED a été divisé en cinq parties, dénotées S1, S2, S3, S4, et S5. L'ensemble d'apprentissage est utilisé pour apprendre le modèle d'ordonnancement proposé. L'ensemble de validation est employé pour corriger les paramètres du modèle d'ordonnancement, tels que le nombre d'itérations *T* de notre algorithme. L'ensemble de test est employé pour vérifier la performance du modèle d'ordonnancement. Il est à noter que puisque nous conduisons la validation sur la base de ces cinq ensembles, le calcul effectué est réellement la moyenne à travers ces différents ensembles.

Afin de comparer la performance et évaluer l'efficacité de l'algorithme proposé, nous utilisons trois mesures d'évaluation largement utilisées pour comparer des SRI et prouvées par LETOR :

Precision at position n (P@n) : la précision à une position n mesure la pertinence dans les n instances les mieux ordonnées de la base test. Elle représente la proportion d'exemples positifs parmi ces n exemples.

$$P@n = \frac{\#docs\ pertinents\ dans\ les\ n\ instances\ les\ mieux\ ordonnées}{n}$$

Mean Average Precision (MAP) : la précision moyenne est la moyenne des précisions calculées en tronquant la liste ordonnée des exemples de test après chaque exemple positif. Pour une simple requête, la précision moyenne est définie comme la moyenne des valeurs de P@n pour tous les documents pertinents :

$$AP = \frac{\sum_{n=1}^{N} (P@n * rel(n))}{\#total\ docs\ pertinents\ pour\ cette\ requête}$$

où *N* est le nombre de documents recherchés, et le *rel(n)* est une fonction binaire sur la pertinence du n^{ième} document.

Normalized Discounted Cumulative Gain (NDCG) : La valeur de NDCG de la liste ordonnée à une position n est calculée comme suit :

$N(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{\log(1+j)}$ où r(j) est le classement du jème document dans la liste d'ordonnancement .

[1] http://research.microsoft.com/users/LETOR/

5. Conclusion

Pour améliorer la qualité de la RD, nous avons proposé un algorithme inductif pour une RI efficace en générant une fonction d'ordonnancement d'alternatives. Cette fonction est issue d'un apprentissage semi-supervisé en utilisant des données partiellement étiquetées. L'originalité de notre approche est qu'elle combine deux techniques d'ordonnancement : une inductive et une transductive. L'algorithme proposé est capable d'inférer un ordre sur une base de test non utilisée pendant la phase d'apprentissage, ce qui le rend plus générique qu'une méthode transductive pure.
Nous nous proposons dans l'étape à suivre de compléter la partie expérimentale en intégrant la méthode transductive et d'étendre l'évaluation en ajoutant la collection de données .GOV, puis d'étudier dans quelle mesure les fonctions d'ordonnancement semi-supervisé permettent d'améliorer le résultat de la reformulation de requêtes.

Bibliographie

1. M.-R Amini., *Apprentissage de Fonctions de Classification et d'Ordonnancement avec des Données Partiellement Étiquetées*, Habilitation, Laboratoire d'Informatique de Paris 6, 2007.
2. N. Usunier, *Apprentissage de fonctions d'ordonnancement : une étude théorique de la réduction à la classification et deux applications à la Recherche d'Information*. Thèse de Doctorat, 2006.
3. M.-R. Amini, V. Truong et C. Goutte, A Boosting Algorithm for Learning Bipartite Ranking Functions with Partially Labeled Data, *In Proceedings of SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.
4. V. Truong, M.-R. Amini et P. Gallinari, Apprentissage de fonctions d'ordonnancement semi-supervisé inductives, *Xème Conférence sur l'Apprentissage artificiel*, CAP, 2008.
5. V. Truong, M.-R. Amini, Apprentissage semi-supervisé de fonctions d'ordonnancement, *7ème Conférence en Extraction et Gestion des Connaissances* (EGC) : 497-507, 2007.
6. A. Agarwal, S. Chakrabarti, Learning random walks to rank nodes in graphs, In Proceeding of the 24th International Conference on Machine Learning (ICML): 9-16, 2007.
7. A. Agarwal, Ranking on graph data, In Proceedings of the 23rd international conference on Machine learning (ICML), 2006, p. 25-32.
8. Zhou, D., O. Bousquet, T. Lal, J. Weston, et B. Schölkopf. Learning with local andglobal consistency. Volume 16, Cambridge, MA, USA : 321–328, 2004.
9. Y. Freund., R. Iyer., R. E Schapire et Y. Singer, An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research : 933-969, 2003.
10. T.-Y. Liu, J. Xu, T. Qin, W.-Y. Xiong et H. Li. LETOR : Benchmark dataset for research on learning to rank for information retrieval. In SIGIR 2007.
11. W. W. Cohen, R. E. Schapire et Y. Singer, Learning to Order Things, JORDAN M. I., KEARNS M. J., SOLLA S. A., Eds., *Advances in Neural Information Processing Systems*, 1998.
12. K. Crammer et Y. Signer, A family of additive online algorithms for category ranking, Journal of Machine Learning Research, 3(6) : 1025-1058, 2003.
13. O. Dekel, C. Manning et Y. Singer, Log-linear models for label ranking, *In Advances in Neural Information Processing Systems 16*, 2004.
14. L. Shen et A.K. Joshi, Ranking and Reranking with Perceptron, *Machine Learning, Special Issue on Learning in Speech and Language Technologies*, 2004.
15. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton et G. Hullender, Learning to rank using gradient descent, *In Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.
16. J. Xu et H. Li. AdaRank: A boosting algorithm for information retrieval, *Proceedings of the 30th Annual International ACM SIGIRl*, Amsterdam, The Netherlands, July 23-27, 2007.
17. Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang et H.-W.Hon, Adapting ranking SVM to document retrieval. *In Proceedings of ACM SIGIR 2006*, New York, NY, USA : 186-193, 2006.
18. T. Qin, T.Y. Liu, J. Xu, H. Li, How to Make LETOR More Useful and Reliable, *Proceedings of SIGIR08 Workshop on Learning to Rank for Information Retrieval*, 2008.