Apprentissage Statistique pour la Constitution de Corpus d'évaluation[1]

Huyen-Trang Vu — Patrick Gallinari

Laboratoire d'Informatique de Paris 6 (LIP 6)
Université Pierre et Marie Curie
8 rue du capitaine Scott
75015 Paris
{vu,gallinari}@poleia.lip6.fr

RÉSUMÉ. *La constitution de corpus d'évaluation est une étape essentielle pour évaluer la performance des systèmes de recherche d'information. Le coût de développement de tels corpus est en général assez élevé à cause en particulier de l'effort humain nécessaire à l'évaluation de la pertinence des documents pour chaque requête. Cette difficulté devient un véritable goulot d'étranglement dans le cas de corpus de très grande taille. Le travail que nous présentons vise à sélectionner adaptativement les documents des corpus d'évaluation. Nous utilisons pour cela des algorithmes d'apprentissage statistique qui optimisent des critères d'ordonnancement. Nous présentons des essais d'un algorithme d'ordonnancement, dit RankBoost, sur des données de TREC et analysons les résultats obtenus au moyen de différents critères. Les résultats de ces analyses montrent que la méthode proposée permet de créer des corpus d'évaluation de qualité supérieure à la méthode de TREC.*

ABSTRACT. *Test collections play a crucial role in Information Retrieval system evaluation. Forming relevance assessment set has been recognized as the key bottleneck in test collection building, especially on very large sized document collections. This paper addresses the problem of efficiently selecting documents to be included in the assessment set. Machine learning algorithms such as RankBoost can be helpful for this purpose. This leads to smaller pools than traditional round robin pooling, thus reduces significantly the manual assessment workload. Experimental results on TREC collections consistently demonstrate the effectiveness of our approach according to different evaluation criteria.*

MOTS-CLÉS : *créer le corpus d'évaluation, apprentissage de fonction d'ordonnancement, algorithme RankBoost, validation statistique*

KEYWORDS: *test collection building, learning-to-rank algorithm, RankBoost algorithm, statistical validation*

1. catégorie : chercheur

1. Introduction

La performance d'un moteur de recherche d'information (RI) est souvent estimée en utilisant des corpus d'évaluation. Pour constituer ces corpus, il faut pour chaque requête, évaluer la pertinence des documents du corpus. Ce travail est en général effectué par des experts humains. Puisqu'il est impossible d'évaluer la pertinence de tous les documents d'un corpus de grande taille, les jugements porteront sur un nombre limité de documents choisis et qui constituent le "pool" de documents jugés. La pertinence de tous ces documents sera alors jugée par un expert, les documents qui ne sont pas choisis sont implicitement considérés comme non pertinents. Ce pool de documents à juger est beaucoup plus petit que le corpus initial. La constitution d'un pool de documents adéquat est une étape très sensible pour le processus d'évaluation. En particulier alors que les corpus deviennent de plus en plus grands, on assiste à une remise en cause des méthodes de *pooling* traditionnelles. On voit émerger le besoin de méthodes permettant la création de pools robustes.

Cet article propose une nouvelle méthode pour sélectionner des documents devant faire partie du pool de documents à juger. Elle repose sur l'utilisation de techniques d'apprentissage statistique et plus précisément d'algorithmes d'ordonnancement. A partir des informations de pertinence apportées par des humains et portant sur un nombre limité de documents, l'algorithme d'ordonnancement apprendra à trier les documents selon leur importance pour la constitution du pool, précisément leur pertinence à une requête demandée. Au final les documents jugés les plus importants constitueront le pool utilisé par la suite. Les jugements initiaux portent sur un nombre de documents suffisamment large pour que l'assesseur n'apporte pas de biais favorisant des documents particuliers ou un type de moteur de recherche spécifique. L'apprentissage est complètement transparent pour l'assesseur, les paramètres des fonctions d'ordonnancement sont estimés en se basant seulement sur les rétroactions fournies par les assesseurs, et s'adaptent au corpus.

Des résultats préliminaires de cette méthode sur un corpus particulier pour la recherche des documents XML ont été présentés dans [VU 05]. Dans ce texte, nous détaillerons la proposition ainsi que montrerons des expériences sur des benchmarks de TREC. Les résultats que nous avons obtenus confirment que le corpus de test créé par notre méthode est beaucoup plus robuste que celui de même taille créé par la méthode utilisée dans TREC. Avec un corpus d'évaluation beaucoup plus petit et donc moins coûteux à produire, notre méthode offre des jugements de performance des moteurs de recherche similaires à ceux obtenus avec le pool original de grande taille de TREC.

La suite du manuscrit est composée de cinq parties. Dans la section 2, nous présentons les travaux antérieurs concernant la construction de corpus d'évaluation. La section 3 décrit la méthode proposée. La section suivante présente les données et la conception de notre expérimentation. La section 5 est consacrée aux résultats expérimentaux effectués par l'algorithme RankBoost [FRE 03]. Nous terminons par une conclusion et des perspectives.

2. Etat de l'art

Nous présentons ci-après deux approches principales pour évaluer des moteurs de recherche d'information. La première s'agit à sélectionner des documents du corpus original sur lesquels les assesseurs sont demandés de juger la pertinence. La deuxième se focalise sur le problème de produire des valeurs de performance pour chaque système de recherche.

2.1. *Echantillonnage du corpus de documents*

L'échantillonnage du corpus de documents, dit "pooling", est l'approche principale pour constituer des corpus d'évaluation. Les premiers travaux sur le sujet datent de 1975 [SPA 75]. Cet article fondateur propose une technique dite de "round robin" qui consiste à ajouter itérativement les documents qui sont à un même rang dans les listes renvoyées par les différents moteurs : on ajoute tout d'abord au pool les documents de rang 1 de chaque moteur, puis ceux de rang 2, etc. Malgré la simplicité de cette stratégie, c'est elle qui a été utilisée dans les campagnes d'évaluation de RI comme TREC, CLEF ou NTCIR. Par la suite on parlera de stratégie TREC pour désigner cette méthode. Différentes propositions ont été faites visant à limiter la taille du pool final. Par exemple, il a été proposé de limiter soit le nombre de systèmes à fusionner, soit le nombre de documents utilisés pour chaque système. Malgré cela, le processus de jugement est toujours extrêmement coûteux. Dans le cas de TREC 8 par exemple, il y a selon les requêtes de 1046 à 2992 documents retournés pour jugement avec une moyenne à 1737. Selon des statistiques de TREC, il faut environ 30 secondes pour vérifier si un document est ou n'est pas pertinent. Cela signifie qu'il faut environ 50 journées homme pour effectuer ce travail. Notons que des documents jugés non pertinents ne contribuent pas à calculer des valeurs de performance des moteurs. Néanmoins, une analyse simple sur les six collections *ad-hoc* de TREC-3 à TREC-8 constate qu'il y a environ 94% des documents qui sont jugés non pertinents [VOO 99]. En partant de ces constats, il est possible en théorie de réduire l'effort d'évaluation humaine tout en conservant le résultat d'évaluation final si on peut déterminer les documents pertinents avant l'étape de jugement de pertinence. Différentes propositions ont été faites dans ce sens, mais on est encore très loin de l'objectif. Nous passons rapidement en revue les principales familles de proposition ci-dessous.

Zobel [ZOB 98] propose d'ajouter graduellement des documents au pool en testant une condition d'arrêt pour chaque requête. L'idée est de s'arrêter avant d'avoir jugé l'ensemble des documents renvoyés par les moteurs. A partir de statistiques effectuées sur le corpus, il apprend une fonction d'extrapolation pour prévoir le nombre de documents pertinents qu'on n'a pas encore vu. C'est cette fonction qui permettra de décider d'arrêter l'examen des documents. Carterette et Allan [CAR 05] remplacent cette fonction par un test statistique. Bien que cette heuristique ait donné des résultats assez intéressants sur les données de TREC, elle manque d'une base théorique solide et est actuellement très compliquée à mettre en œuvre car le test d'arrêt doit être effec-

tué sur toutes les paires des moteurs de recherche utilisés dans la constitution du pool. De plus, cette approche pose la question de l'objectivité des jugements puisque l'assesseur sait que les documents arrivant le plus tard sont ceux qui sont à la fin des listes de recherche et qu'ils ont beaucoup moins de chance d'être pertinents [VOO 99].

Cormack et al. [COR 98] proposent la méthode Move-To-Front (MTF) qui favorise les documents trouvés par les systèmes qui ont le plus contribué à découvrir des documents pertinents. Aslam et al. [ASL 03] formulent cette règle de priorité par un algorithme d'apprentissage en-ligne [FRE 97].

Notre méthode se base sur l'idée de pondérer des documents [COR 98, ASL 03]. Il y a cependant deux différences majeures. Toutes les propositions décrites ci-dessus favorisent un mode de sélection en-ligne nécessitant des séquences d'interaction avec l'assesseur. La méthode que nous proposons fonctionne hors-ligne et semble ainsi plus adaptée à la tâche car elle limite les interactions avec l'assesseur et donc le biais que celui ci pourrait introduire. Un autre avantage s'agit à l'efficacité du processus entier. L'entraînement hors-ligne nous permet de bien séparer le processus de trier des documents à celui de jugement. Le processus de jugement a lieu donc consécutivement sans aucune interruption de la durée de trier des documents. La deuxième différence est que la procédure d'apprentissage statistique que nous utilisons permet d'obtenir un modèle plus fiable à partir de moins de données étiquetées que l'estimation statistique traditionnelle ou les règles utilisées dans les propositions précédentes [ZOB 98, COR 98].

2.2. *Evaluation des moteurs par des jugements incomplets ou incertains*

Une seconde famille de méthodes propose d'estimer directement la valeur des mesures de performance à partir des jugements effectués sur une petite partie du corpus des documents. Aslam et al. [ASL 05] ainsi que Cormack et al. [COR 99] rééchantillonnent les documents déjà jugés pour pouvoir calculer des mesures traditionnelles en RI comme la précision moyenne non-interpolée (MAP), la précision après *n* documents renvoyés (P@n) ou encore *n* comme étant le nombre de documents pertinents de cette requête (R-Prec). Buckley et Voorhees [BUC 04] proposent une nouvelle mesure, dite "bpref", où tous les documents hors du pool ne sont pas pris en compte. Cette dernière mesure peut être vue comme une mesure du nombre de paires bien ordonnées dans la liste des documents renvoyés par les moteurs (un document pertinent est placé devant un document non pertinent). Nous verrons dans la section suivante que l'algorithme RankBoost utilisé dans notre expérimentation permet d'apprendre à maximiser cette performance.

Pour prendre en compte le contexte internet ou plus généralement les corpus dynamiques et bruités, quelques auteurs ont proposé d'évaluer des systèmes sans jugement de pertinence comme par exemple [NUR 06, CRE 03, JOA 02a, SOB 01]. Pour cela, ils construisent soit explicitement, soit implicitement un pseudo-corpus de test où la pertinence d'un document est incertaine. Nous ne détaillons pas plus cette piste car

notre but est de fournir des corpus de test réutilisables et non pas des pseudo-jugements temporaires comme ces approches.

La diversité et la quantité des travaux consacrés à la constitution de corpus d'évaluation présentés ci-dessus sont représentatives de l'importance de l'enjeu et de la difficulté du problème.

3. Créer le corpus de test avec le support de l'apprentissage automatique

3.1. *Cadre général*

Pour chaque requête, les documents trouvés par tous les systèmes sont ramassés. La procédure de création de corpus de test est composée de trois étapes principales. Le processus d'apprentissage lui même intervient entre deux interactions avec l'assesseur. La première étape consiste à construire un corpus d'apprentissage en demandant aux assesseurs de juger un petit nombre de documents renvoyés par un ensemble de moteurs de recherche pour toutes les requêtes disponibles. Vient ensuite l'apprentissage d'une fonction d'ordonnancement des documents à partir de cette base de documents et de requêtes étiquetée. C'est cette étape qui détermine l'ordre des documents du corpus entier. La dernière étape vise à juger les documents choisis à partir de l'ordre ci-dessus pour le pool d'évaluation.

3.2. *Trier les documents par l'algorithme RankBoost*

La partie estimation d'une fonction de score est totalement cachée à l'assesseur. Dans cet algorithme, la fonction de score pourrait être calculée de différentes façons. TREC par exemple utilise une fonction de vote très simple sur les sorties des différents systèmes. L'information de pertinence est largement sous exploitée par ce type de méthode. Nous proposons ici d'apprendre une fonction de score en utilisant un algorithme d'ordonnancement. Le sujet a suscité de nombreux travaux dans la communauté de l'apprentissage statistique. De nombreuses applications de ces méthodes ont vu le jour récemment. Nous allons présenter brièvement ci-dessous le principe de l'ordonnancement en apprentissage et l'algorithme utilisé.

Dans notre cas, le problème d'ordonnancement des documents revient à ordonner tous les documents pertinents pour la requête devant les documents non pertinents. Plus précisément, le problème se formule comme suit. Pour chaque requête, la collection de documents est divisée en deux sous-ensembles disjoints, celui des documents pertinents $\mathcal{S}_+$ et celui des documents non pertinents $\mathcal{S}_-$. R et NR sont respectivement leurs cardinalités. Une fonction d'ordonnancement $h(d)$ affectera à chaque document d une valeur réelle telle que l'ordre de ces valeurs respecte la relation de pertinence : un document pertinent aura une valeur supérieure à un document non pertinent. On note "$\triangleright$" l'ordre des documents selon leur valeur de pertinence, i.e.

$d \rhd d'$ signifie que d est plus pertinent que d'. Formellement, la condition s'écrit : $d_+ \rhd d_- \Leftrightarrow h(d_+) > h(d_-), \forall (d_+, d_-) \in \mathcal{S}_+ \times \mathcal{S}_-$.

La mesure de bon ordonnancement la plus naturelle compte la proportion de paires de documents mal ordonnées (i.e. un document pertinent est classé en dessous d'un document non pertinent) parmi toutes les paires possibles *R.NR*.

$$\mathbf{RLoss}(h) = \frac{1}{R.NR} \sum_{\substack{d_+ \in \mathcal{S}_+, \\ d_- \in \mathcal{S}_-}} [\![d_- \rhd d_+]\!] = \frac{1}{R.NR} \sum_{\substack{d_+ \in \mathcal{S}_+, \\ d_- \in \mathcal{S}_-}} [\![h(d_-) > h(d_-)]\!] \tag{1}$$

où $[\![\phi]\!]$ vaut 1 si ϕ est vrai, 0 sinon.

Cette mesure est non différentiable et difficile à étudier de manière théorique et à optimiser en pratique. On la remplace souvent par des fonctions différentiables, la plus utilisée étant la fonction de coût exponentielle ELoss.

$$\mathbf{ELoss}(h) = \frac{1}{R.NR} \sum_{(d_+, d_-)} e^{\left(h(d_-) - h(d_+)\right)} \tag{2}$$

Cette fonction ELoss étant différentiable et convexe, il est donc possible de concevoir des algorithmes d'apprentissage efficaces en utilisant des outils disponibles dans le domaine de l'optimisation convexe. Il a par ailleurs été démontré que la solution optimale de ELoss et de RLoss sont identiques [FRE 03].

RankBoost [FRE 03] est un des rares algorithmes d'ordonnancement à partir d'exemples dont la capacité de généralisation a été prouvée de manière formelle [FRE 03, CLé 05]. Cet algorithme a pour but d'estimer une fonction de score pour chaque document d en minimisant itérativement la borne supérieure exponentielle (Eq. (2)) de la fonction d'erreur d'ordonnancement (Eq. (1)). L'algorithme crée au fur et à mesure une suite de fonctions de score h_t. Ces fonctions sont ensuite combinées linéairement de façon à fournir une fonction de score globale. Les coefficients α_t de cette combinaison linéaire sont estimés par l'algorithme d'apprentissage. La fonction de score d'un document d s'écrit donc $H(d) = \sum_{t=1}^{T} \alpha_t h_t(d)$.

Le choix de RankBoost pour nos expériences a également d'autres raisons que ces résultats formels concernant les capacités de généralisation de l'algorithme. Tout d'abord, RankBoost est capable de prendre en compte des valeurs d'entrée relatives : il n'y a pas de besoin de normaliser les scores des moteurs de recherche pour utiliser l'algorithme. Ensuite, cet algorithme n'exige pas l'indépendance des listes renvoyées par les moteurs de recherche. Ces deux propriétés sont particulièrement précieuses pour fusionner des systèmes hétérogènes avec différentes échelles de score de pertinence. Finalement, dans le cas des jugements binaires qui est le plus courant en RI (i.e. un document est soit pertinent soit non pertinent pour une requête), il existe une implantation très efficace de RankBoost dont la complexité est linéaire par rapport au nombre d'exemples d'entraînement (*cf.* le texte original de l'algorithme).

4. Conception d'expérimentation

La méthode proposée est utilisable dans le même contexte que celle mise en œuvre dans TREC. Des résultats sur la recherche des documents XML ont été décrits dans [VU 05]. Des résultats que nous avons obtenus sur les trois corpus de la tâche ad-hoc de TREC-6, TREC-7 et TREC-8 sont similaires. Faut de place, nous ne présenterons dans ce texte que les résultats obtenus avec la collection de TREC-8. Cette collection comprend les résultats de 129 moteurs de recherche pour 50 requêtes. Pour chaque requête, un moteur ne peut renvoyer que les 1000 premiers documents dont seulement les 100 premiers documents de 79 moteurs sont rajoutés au pool TREC [VOO 99]. Ce pool de documents jugés par des experts sera appelé dans la suite pool de référence. Nous comparerons les résultats obtenus à partir des pools construits par RankBoost à ceux obtenus par ce pool de référence ou encore par la méthode TREC en limitant le nombre de documents utilisés. Nous notons Depth-n le corpus créé par la stratégie de TREC jusqu'à la profondeur n. Ce pool contient m documents au total. On note RBoost-m le pool produit par RankBoost. Nous avons échantillonné sur des valeurs de $n = 1, \cdots, 10, 15, 20, 25, 50, 100$. Dans les graphes suivants, des valeurs de n seront explicitées sur des courbes. Des pools sont visualisés jusqu'à la taille de 1000 documents.

Pour examiner l'avantage de pondération des systèmes de RankBoost, nous rajoutons aussi MTF [COR 98] dans l'expérimentation. MTF (*cf.* section 2.1) est une stratégie simple de pondérer des systèmes. Cormack et al. ont testé deux modes de MTF, le mode global et le mode local. Le premier examine toutes les requêtes en même temps, le nombre de requêtes actives est possiblement diminué. Le deuxième constitue le pool pour chaque requête indépendamment. Nous ne font pas la sélection des requêtes à juger, le mode global n'est alors pas vraiment compatible avec notre approche. Nous allons détailler le comportement du mode local et notons MTF-m.

Une des difficultés pour évaluer la qualité des pools et les comparer est qu'il n'existe pas de mesure standard de qualité et qu'aucune mesure n'est à elle seule pleinement satisfaisante. Nous employons ici différents critères pour mesurer cette qualité. Ils permettent des analyses complémentaires des résultats. Comme des expériences antérieures de ce sujet, nous avons d'abord étudié la corrélation entre les listes de moteurs ordonnés par les deux techniques de pooling. Nous regardons ensuite la capacité du système de pooling à bien distinguer les moteurs, notamment pour ceux qui sont statistiquement très proches du meilleur moteur.

Nous choisissons le critère MAP[1] proposé par TREC pour mesurer la performance d'un moteur de recherche. Cette mesure est montrée fiable pour les grandes collections de documents [HAW 03] ainsi que pour des bases de requêtes peu nombreuses [BUC 00].

Pour constituer la base d'apprentissage, nous avons utilisé les 5 premiers résultats renvoyés par les moteurs de recherche. Cette profondeur a été choisie arbitrairement

1. disponible dans le package "trec_eval"

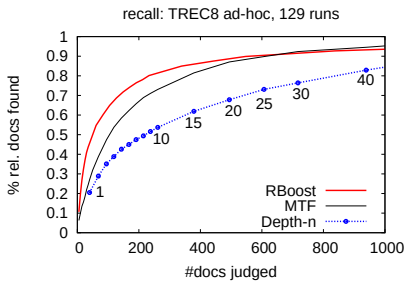


Figure 1 – Comparaison de la qualité des pools RBoost-*m*, MTF-*m* et de Depth-*n* en fonction du nombre de documents pertinents du pool qui ont été jugés.

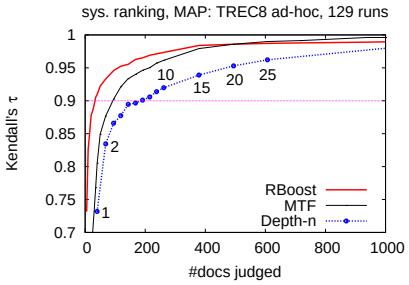


Figure 2 – Corrélation de l'ordre des moteurs selon RBoost-*m*, MTF-*m* ou Depth-*n* par rapport à l'ordre de référence de TREC.

pour fournir une quantité suffisante de données d'entraînement et pour que le biais lié à l'assesseur soit suffisamment faible.

Pour valider la solution avec un petit nombre de données d'entraînement, nous avons employé la stratégie de "leave-one-out" : un modèle est appris pour chaque requête en utilisant les jugements de pertinence de toutes les autres requêtes. Il est testé sur la requête initiale. Les résultats sont ensuite moyennés. Concernant RankBoost, nous avons utilisé 100 itérations de boosting, et imposé des contraintes de positivité sur les coefficients α_t (ce qui permet dans une certaine mesure de contrôler la généralisation de l'algorithme).

5. Résultats expérimentaux

5.1. *Identification des documents pertinents*

La figure 1 donne le taux de documents pertinents trouvés par notre procédure et ceux par MTF et par TREC. La courbe obtenue avec RBoost est largement au dessus de celle de la procédure TREC. La courbe de MTF est au milieu au début et s'approche à celle de RBoost après avoir au moins 600 documents dans le pool. Autrement dit, l'utilisation de RankBoost a permis de faire un très bon échantillonnage des documents pertinents de la collection.

5.2. *Corrélation des listes de systèmes ordonnées suivant leur performance*

Un pool ayant été obtenu par une méthode donnée, les assesseurs vont proposer des jugements sur tous les documents de ce pool. Ces jugements serviront à classer les moteurs de recherche. Nous comparons ici la liste obtenue par le pool de référence

de TREC (*cf.* section 4) avec les listes ordonnées des moteurs de recherche obtenues par le pool Depth-n pour une profondeur n variant de 1 à 100. Nous comparons de même la liste obtenue par le pool de référence à celle obtenue avec le pool RBoost-m ou MTF-m. Nous utilisons classiquement le τ de Kendall pour mesurer la corrélation entre deux listes ordonnées. Le résultat est donné dans la figure 2. On voit que la courbe RBoost est très nettement supérieure à celle obtenue par Depth-n. La courbe MTF est toujours au milieu au début, s'approche et un peu s'avance à celle de RBoost aux pools de grande taille à partir de 500 documents.

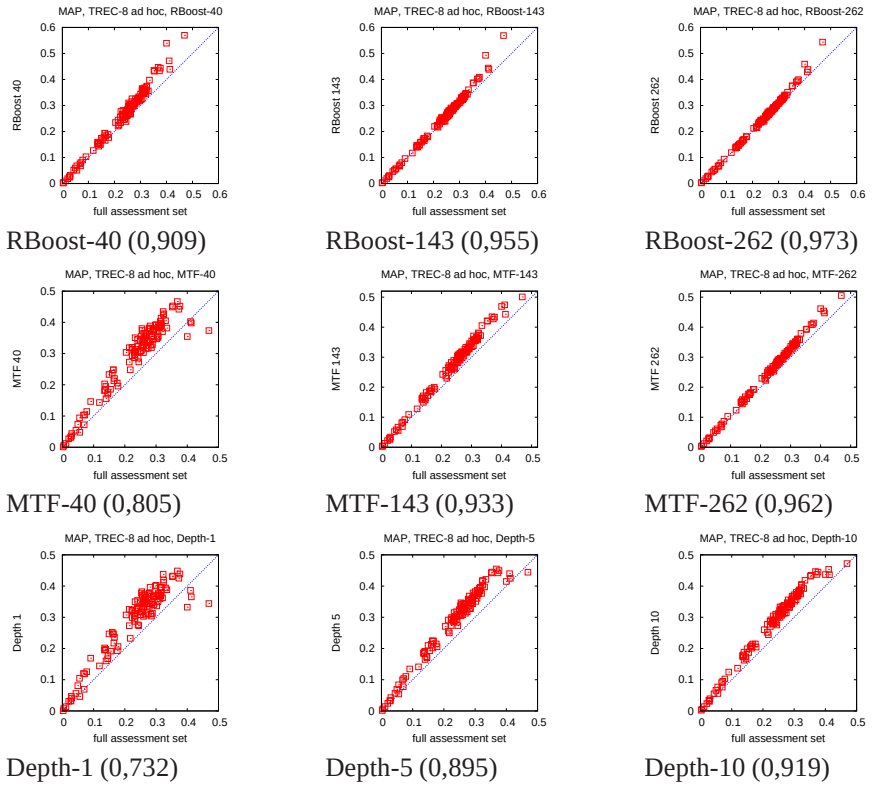
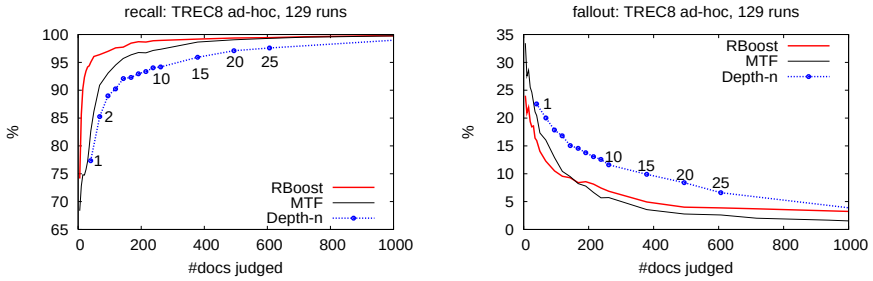


Figure 3 – MAP des moteurs selon le corpus de RBoost-m, MTF-m et Depth-n resp. vs. du corpus original. La valeur τ de Kendall est entre parenthèses.

Des études récentes de la constitution de corpus d'évaluation comme par exemple [CAR 05, SAN 04, COR 98] considèrent le τ de Kendall comme la statistique *de facto* pour évaluer la fiabilité d'un pool et la valeur $0,9$ comme le seuil plafond d'un bon pool. Notons que la liste produite à partir du pool RBoost-35 passe ce seuil, i.e. avec un pool plus petit que Depth-1. En réalité, cette statistique ne fournit pas toute l'information pertinente pour cela. Elle ne tient compte pas du tout la position d'un moteur

dans une liste, ni le fait que des moteurs peuvent ne pas être différents statistiquement. Il est possible de vérifier ce fait par des graphes. La figure 3 présente la valeur du critère MAP qui sert à ordonner les systèmes pour les petits pools obtenus par RBoost-*m*, MTF-*m* et Depth-*n* versus celles obtenues pour le pool de référence. Nous remarquons que les quatre premiers moteurs ont été mal évalués par les petits pools de Depth-*n* et de MTF-*m*. En revanche, le pool correspondant de RBoost fonctionne bien.



(a) rappel (b) taux de faux positifs

Figure 4 – Comparaison de la qualité des pools RBoost-*m*, MTF-*m* et Depth-*n* en fonction des paires de systèmes significativement différents.

5.3. *Validation statistique*

5.3.1. *Détection des différences significatives*

Nous avons analysé pour un pool donné les couples de systèmes qui ont des résultats significativement différents l'un de l'autre sur l'ensemble des requêtes. La qualité d'un pool peut être mesurée également par la similarité entre ses paires significativement différentes et celles obtenues à partir du pool de référence. Nous avons utilisé un *t*-test pour chaque paire de moteurs avec un niveau de confiance de 95%. Le rappel et le taux de faux positifs obtenus par les pools de petite taille fournis par RBoost offrent systématiquement des résultats nettement supérieurs à ceux obtenus par la méthode TREC et de MTF aux pools de petite taille, voir la figure 4.

5.3.2. *Groupement de Tukey*

Ce test de multi-comparaison[2] vise à grouper des moteurs selon leur similarité statistique. Nous nous focaliserons particulièrement sur le premier groupe, dit groupe A, contenant les moteurs similaires au sens statistique au moteur le meilleur de la compétition. Plus petite la cardinalité du groupe A, plus fiable le pool. On voit de la figure 5 que les pools RBoost fonctionnent beaucoup plus stablement que ceux de MTF et de TREC.

2. disponible dans le package IR-STAT-PAK [TAG 95]

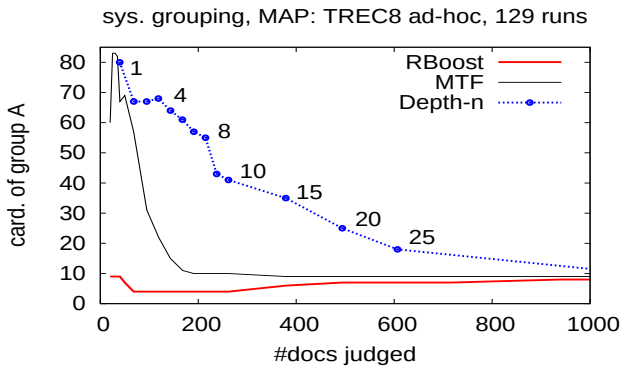


Figure 5 – Cardinalité du groupe A selon le test de Tukey.

6. Conclusions et Discussions

Notre étude a bien montré que des algorithmes d'apprentissage entraînés selon des critères d'ordonnancement permettent de créer des corpus d'évaluation d'excellente qualité. Le corpus final est non seulement suffisamment petit pour permettre l'emploi de jugements humains mais est également plus fiable que le corpus de même taille créé par la stratégie conventionnelle de TREC ainsi que cela par MTF.

Du point de vue empirique, il est nécessaire de vérifier l'efficacité de la méthode proposée sur de nouveaux scénarios de recherche de TREC comme celui en très grande taille "Terabyte", ou de tâche "Topic Distillation" en environement de page web où le fait que le nombre de réponses pertinentes varie largement parmi les requêtes rend l'apprentissage statistique plus difficile. Du point de vue théorique, au-delà de cette première exploration, il serait important de définir quelles sont les classes d'algorithmes permettant la construction de corpus d'évaluation. Nous explorons actuellement d'autres alternatives basées sur des algorithmes à noyaux de type SVM qui ont déjà été employés dans le cas de la classification et de la méta-recherche [JOA 02b].

Remerciements M.-R. Amini, B. Piwowarski, J. Zobel et les trois rapporteurs anonymes ont fait des critiques constructives. E. Voorhees nous a permi l'accès des données et des détails de pooling de TREC. G.V. Cormack nous a fait parvenir des détails de MTF. Le travail a été soutenu partiellement par le programme IST de la communauté Européenne, sous le réseau d'excellence de PASCAL, IST-2002-506778. La publication ne reflète que des vues des auteurs.

7. Bibliographie

[ASL 03] A J. A., P V., S R., « A unified model for metasearch, pooling, and system evaluation », *Proc. CIKM'03*, 2003, p. 484–491.

[ASL 05] A J. A., P V., Y E., « A Sampling Technique for Efficiently Estimating Measures of Query Retrieval Performance Using Incomplete Judgments », *Proc. Wkshop*

Learning with Partially Classified Training Data, ICML'05, 2005.

[BUC 00] B C., V E. M., « Evaluating evaluation measure stability », *Proc. SIGIR'00*, 2000, p. 33–40.

[BUC 04] B C., V E. M., « Retrieval evaluation with incomplete information », *Proc. SIGIR'04*, 2004, p. 25–32.

[CAR 05] C B., A J., « Incremental Test Collections », *Proc. CIKM'05*, 2005, p. 680–687.

[CLé 05] C ´ ¸ S., L G., V N., « Ranking and scoring using empirical risk minimization », *Proc. COLT 2005*, 2005.

[COR 98] C G. V., P C. R., C C. L. A., « Efficient construction of large test collections », *Proc. SIGIR'98*, 1998, p. 282–289.

[COR 99] C G., L O., P C. R., « Estimating precision by random sampling », *Proc. SIGIR'99*, 1999, p. 273–274.

[CRE 03] C F., W S., « Methods for Ranking Information Retrieval Systems Without Relevance Judgements », *Proc. SAC 2003*, 2003, p. 811–816.

[FRE 97] F Y., S R. E., « A decision-theoretic generalization of on-line learning and an application to boosting », *J. Comput. Sys. Sci.*, vol. 55, nº 1, 1997, p. 119-139.

[FRE 03] F Y., I R., S R. E., S Y., « An efficient boosting algorithm for combining preferences », *J. Mach. Learning Res.*, vol. 4, 2003, p. 933–969.

[HAW 03] H D., R S., « On Collection Size and Retrieval Effectiveness », *Information Retrieval*, vol. 6, nº 1, 2003, p. 99–105.

[JOA 02a] J T., « Evaluating Retrieval Performance Using Clickthrough Data », *Proc. SIGIR Wkshop on Mathematical/Formal Methods in IR*, 2002.

[JOA 02b] J T., « Optimizing search engines using clickthrough data », *Proc. KDD'02*, 2002, p. 133–142.

[NUR 06] N R., C F., « Automatic ranking of information retrieval systems using data fusion », *Info. Process. Management*, vol. 42, nº 3, 2006, p. 595–614.

[SAN 04] S M., J H., « Forming test collections with no system pooling », *Proc. SIGIR'04*, 2004, p. 33-40.

[SOB 01] S I., N C., C P., « Ranking Retrieval Systems without Relevance Judgments », *Proc. SIGIR'01*, 2001, p. 66–73.

[SPA 75] S J K., V R C. J., « Report on the need for and provision of an ideal information retrieval test collection », rapport nº 5266, 1975, Computer Lab., Univ. Cambridge.

[TAG 95] T -S J., B J., « A statistical analysis of the TREC-3 data », *Proc. TREC-3*, 1995, p. 385–398.

[VOO 99] V E. M., H D., « Overview of the Eighth Text REtrieval Conference (TREC-8) », *Proc. TREC 8*, 1999, p. 1–24.

[VU 05] V H.-T., G P., « Using RankBoost to Compare Retrieval Systems », *Proc. CIKM'05*, 2005, p. 309–310.

[ZOB 98] Z J., « How reliable are the results of large-scale information retrieval experiments ? », *Proc. SIGIR'98*, 1998, p. 307–314.