RECITAL 2004, Fès, 21 avril 2004

Méthodes statistiques et apprentissage automatique pour l'évaluation de requêtes en recherche documentaire

Jens GRIVOLLA*†
encadré par Pierre Jourlin et Renato de Mori
Laboratoire Informatique d'Avignon (LIA)

Résumé - Abstract

Pour la recherche documentaire il est souvent intéressant d'avoir une bonne mesure de confiance dans les réponses trouvées par le moteur de recherche.

Une bonne estimation de pertinence peut permettre de faire un choix entre plusieurs réponses (venant éventuellement de différents systèmes), d'appliquer des méthodes d'enrichissement additionnelles selon les besoins, ou encore de permettre à l'utilisateur de prendre des décisions (comme d'approfondir la recherche à travers un dialogue).

Nous proposons une méthode permettant de faire une telle estimation, utilisant des connaissances extraites d'un ensemble de requêtes connues pour en déduire des prédictions sur d'autres requêtes posées au système de recherche documentaire.

In document retrieval applications it is often interesting to have a measure of confidence in the answers found by the retrieval system.

A good relevance estimation can allow us to make a choice between different answers (possibly provided by different sources), apply additional expansion techniques according to the specific needs, or enable the user to make decisions (such as to refine the search interactively).

We propose a method that allows us to make such estimations, using knowledge extracted from a known query corpus to deduce predictions on new queries presented to the document retrieval system.

Mots-clefs – Keywords

apprentissage/décision automatique, recherche documentaire, expansion de requêtes, évaluation de difficulté
automatic learning/decision, document retrieval, query expansion, difficulty evaluation

*jens.grivolla@lia.univ-avignon.fr
†avec le soutien de Digitech S.A. et de la Région PACA

1 Introduction

Un système de recherche documentaire doit trouver dans une base de documents (typiquement une collection de textes) les documents répondants à une requête posée (en langage naturel) par un utilisateur.

En général, un système de recherche documentaire répondra à une requête posée par une liste ordonnée de documents. Le but dans la conception d'un système de recherche est de maximiser le nombre de documents pertinents dans la liste, surtout en tête de liste, et de minimiser le nombre de documents non-pertinents.

Dans la plupart des applications, la performance d'un système sera jugé selon la précision, c'est-à-dire le pourcentage de documents pertinents parmi les documents trouvés par le système de recherche. Une autre mesure importante est le rappel, le pourcentage des documents pertinents contenus dans la collection qui ont été retournés par le système.

Nous avons utilisé la *précision moyenne*, qui tient compte aussi bien du rappel que de la précision, ainsi que du classement relatif des documents dans la liste.

Beaucoup de travail est fait par un grand nombre de chercheurs pour améliorer les performances, en particulier par des méthodes d'enrichissement de requêtes, c.à.d. des modifications des requêtes p. ex. en rajoutant des termes additionnels (voir section 3) afin de trouver des documents pertinent supplémentaires.

Malheureusement, ces méthodes ne sont efficaces que pour une partie des requêtes et peuvent même dégrader les résultats pour d'autres. Pour certaines requêtes aucun système n'arrive à fournir de réponses satisfaisantes.

De ceci se dégagent deux problématiques :
– déterminer le traitement optimal pour chaque requête
– identifier les requêtes pour lesquelles la recherche échoue afin de p. ex. poursuivre une approche interactive

2 L'environnement d'application

Toutes les expérimentations présentées dans cet article ont été effectuées sur la base du corpus de documents et des requêtes issues des campagnes TREC (*Text REtrieval Conference*). Pour la discipline appelée *adhoc*, la recherche non-interactive automatique à partir de requêtes écrites en langage naturel, ces campagnes fournissent chaque année un corpus de documents (environ 500 000) et 50 requêtes, ainsi que les moyens pour évaluer automatiquement les performances d'un système de recherche documentaire sur ces données.

Les requêtes sont disponibles sous différentes formes, pouvant être composées d'un *titre* (quelques mots clés), d'un *descriptif* (une ou deux phrases) et d'un *narratif* (explication détaillée de la thématique recherchée).

Nos méthodes ont été appliquées sur la base d'un système de recherche documentaire utilisant le modèle probabiliste. Le système implémenté par Pierre Jourlin est relativement classique, proche du système OKAPI, et avait préalablement servi dans le contexte du *«spoken document retrieval» (SDR)* (Jones *et al.*, 2001).

3 L'enrichissement des requêtes

Une méthode pour enrichir des requêtes sans interaction avec l'utilisateur est le *blind relevance feedback (BRF)*(Walker & de Vere, 1990).

Partant de l'hypothèse que les premiers documents dans la liste retournée par le système de recherche ont une forte probabilité d'être pertinents, un nombre t de termes caractérisant les d premiers documents est ajouté à la requête initiale. La requête ainsi enrichie peut permettre de retrouver des documents pertinents qui ne contiennent pas les termes employés dans la requête d'origine, et ainsi augmenter le rappel. Diverses études sur le *BRF* ont montré que la précision est (en moyenne) également meilleure en utilisant des requêtes enrichie de cette manière.

Nous avons constaté que (quel que soit le paramétrage ou le type de requête) l'application de BRF dégrade les résultats obtenus pour au minimum environ 30% des requêtes, malgré une amélioration globale des performances. Pour les requêtes courtes consistant uniquement du titre ou du descriptif, le taux d'échec peut dépasser les 50%. Il est donc intéressant d'éviter cet effet négatif pour les requêtes concernées.

3.1 La difficulté des requêtes et l'effet de l'enrichissement

Nos travaux portant sur les deux aspects de la difficulté des requêtes posées et de l'effet qu'ont des techniques d'enrichissement selon la requête, nous avons également analysé le lien entre la précision moyenne obtenue (sans BRF) et le gain apporté par l'enrichissement de la requête.

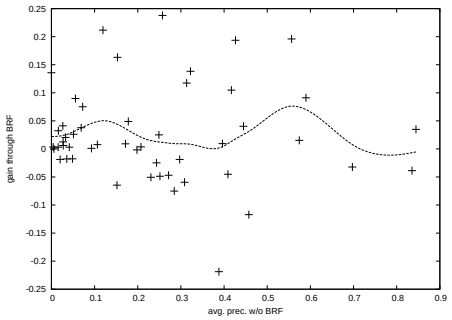

FIG. 1 – précision moyenne et gain par BRF : TREC 8 ad-hoc

On constate dans la figure 1 que l'expansion de requête fonctionne surtout sur des requêtes pour lesquelles on obtient des performances moyennes sans BRF, alors que pour les requêtes qui donnaient des résultats très bon ou très mauvais l'enrichissement a plutôt tendance à dégrader le résultat.

On peut supposer que ceci s'explique principalement par le fait que l'enrichissement utilise des mots extraits des premiers documents dans la liste. Si ceux-ci ne sont pas pertinents, il n'est pas possible de trouver des termes représentatif de la thématique recherchée, l'expansion de requête ne peut donc pas améliorer les performances. Dans le cas contraire, les documents trouvés étant bons, une nouvelle recherche avec un classement différent des documents risque de déclasser les documents pertinents initialement trouvés et ainsi de détériorer le résultat.

4 Notre approche

L'approche poursuivie consiste à déterminer un ensemble d'attributs calculables sur la base de la requête même et (selon les informations disponibles) des réponses ainsi que du processus ayant conduit à la réponse (scores internes du système de recherche documentaire, documents dont la réponse est extraite dans les applications *QA*, etc.)

Chaque requête ou chaque couple requête-réponse sera ainsi représenté par un vecteur d'attributs d'une dimension variant selon l'ensemble d'attributs choisi. En utilisant un corpus de requêtes pour lesquelles la performance du système est connue, on peut ensuite appliquer des méthodes de classification automatique afin d'aboutir à des prédictions par rapport au critère de classification (p. ex. la difficulté de la requête en termes de précision moyenne obtenue) pour des nouvelles requêtes non comprises dans ce corpus d'apprentissage.

Nous avons développé une liste d'attributs plus ou moins fortement corrélés avec la difficulté de la requête (en terme de précision obtenue par le moteur de recherche). Sur la représentation vectorielle de la requête, nous avons appliqué différentes méthodes de classification et décision automatique (arbres de décision, SVM, etc.) afin d'obtenir des prédictions.

Les attributs choisis peuvent se diviser en deux types :
– des attributs basés purement sur la requête même
– des attributs utilisant les résultats d'une première phase de recherche
La première catégorie comprend p. ex. la longueur de la requête, des mesures de spécificité des termes de la requête (hyponymie, IDF, ...) ou encore des mesures d'ambiguïté (le nombre de sens des termes de la requête, ...) ainsi que des scores divers calculés sur la base de telles propriétés.

Parmi les attributs du deuxième type ce trouvent en particulier des mesures de similarité ou distance entre les documents trouvés, les scores des documents d'après la formule Okapi-BM25[1] (Robertson *et al.*, 1996) et des scores dérivés.

Nous avons utilisé différents classifieurs génériques sur différentes représentations vectorielles des informations disponibles : arbres de décision, CAL5, Dipol, SVM, NaiveBayes, réseaux de neurones, etc. Le système WEKA (Witten & Frank, 1999) permet l'utilisation d'un grand nombre de classifieurs différents, mais nous avons également évalué d'autres implémentations.

Les algorithmes à base d'arbres de décisions présentent l'avantage de construire des règles de classification qui peuvent être lues et interprétées. Ceci peut livrer des informations intéressantes et exploitables sur le processus de classification. Par contre, sur le nombre limité de données disponible d'autres classifieurs (en particulier les SVM) permettent d'obtenir de meilleurs résultats de classification.

Nous avons également utilisé un classifieur spécial pour la catégorisation de textes, les *«Semantic Classification Trees» (SCT)* (Kuhn & De Mori, 1995), qui construit à partir d'un ensemble de textes (dans notre cas des requêtes) un arbre basé sur des expressions régulières (extraites automatiquement) qui servent à diviser le corpus sur les différentes branches de l'arbre.

[1] $\sum_t cw_{t,d} = \sum_t \mathrm{qtf}_t \frac{(K+1)\times tf_{t,d}}{K\times(1-b+b\times L_d)+tf_{t,d}} \log \frac{N}{N_t}$

5 Résultats

5.1 Estimation de difficulté

Sur l'estimation de difficulté des requêtes nous avons séparé les requêtes en deux classes : «faciles» et «difficiles», selon la précision moyenne obtenue par rapport à un seuil fixé.

Nous avons ensuite utilisé la méthode *«leave one out»* afin de découper le corpus de requêtes disponibles en *entraînement* et *test*. Pour chaque requête, un classifieur a ainsi été construit sur la base du reste du corpus et appliqué sur la requête restante pour évaluer la qualité des prédiction. La précision de classification devait ensuite être comparée au taux obtenable par la connaissance de la distribution des classes (en faisant systématiquement la prédiction de la classe plus grande).

Le seuil devait être fixé de manière relativement arbitraire (il dépendrait d'une application pratique éventuelle), nous avons choisi de prendre pour l'instant la valeur médiane ainsi que la moyenne pour évaluer notre système.

Actuellement, en utilisant un classifieur à base de SVM et un jeu d'environ 20 caractéristiques (plus un grand nombre de variations) nous arrivons à un taux de classification correcte de 70% avec la valeur médiane comme seuil (par rapport à 50% basé sur la distribution des classes). Avec le seuil fixé à la valeur moyenne nous obtenons une précision de 73% (contre 62%).

Ces valeurs ont été obtenues sur les requêtes de TREC 8 et semblent être significativement au-dessus des taux de référence. Il reste à vérifier si ces performances sont généralisables à d'autres corpus, mais cela semble probable, aucune «optimisation» n'ayant été faite sur le corpus donné. Il reste également à étudier quel taux de classification peut être obtenu avec des seuils fixés différemment.

Utilisant un corpus très limité pour l'apprentissage du classifieur, les SVMs se sont avérés les plus performants. Dans un contexte avec un plus grand nombre d'exemples disponibles, il est pensable que d'autres méthodes obtiennent de meilleurs résultats, cela n'a pas encore pu être vérifié.

Nous avons fait quelques tests avec des classifieurs capables de prédire des valeurs continues afin d'avoir une estimation directe de la précision (du moteur de recherche) attendue, mais nous n'avons pas obtenu de résultats satisfaisants, probablement dû à la faible quantité de données dont nous disposons.

5.2 Décision sur l'enrichissement des requêtes

Au-delà de l'estimation de la qualité attendue du résultat, il est possible d'utiliser la classification afin de décider de l'utilisation de méthodes d'expansion des requêtes (ou d'autres traitements spécifiques).

Une première étude à permis de déterminer le potentiel de notre approche en supposant qu'on soit capable d'une décision parfaite sur l'application d'enrichissement pour chaque requête.

En particulier, sur les requêtes de taille moyenne (titre et descriptif) de TREC 8, notre système arrive à une moyenne (sur toutes les requêtes) des «précisions moyennes» de 24% sans expansion et 27% avec le meilleur paramétrage choisi globalement pour toutes les requêtes. Par

contre, en choisissant les meilleurs paramètres pour chaque requête on arrive à 33%, avec un simple choix binaire d'application de BRF sans variation des paramètres 29% sont possibles.

Il est donc possible (en principe) d'obtenir des performances considérablement supérieures aux meilleures performances n'utilisant pas de décision pour chaque requête individuelle.

La décision sur le paramétrage précis de l'expansion à appliquer demande l'utilisation de classifieurs permettant une prédiction numérique des paramètres et suppose que l'estimation des différents paramètres est indépendante (qu'on peut donc estimer les valeurs optimales de d et t séparément).

Nous nous sommes pour l'instant contentés d'une décision binaire d'application ou non de l'expansion (avec un paramétrage fixé) selon les probabilités d'une amélioration ou dégradation des performances. Un grand choix de méthodes de classification peut être utilisé pour cela.

En appliquant les prédictions issues d'un classifieur dans la pratique nous n'obtenons actuellement pas de performances significativement meilleures que sans décision. De plus, nous avons constaté que les résultats varient fortement selon le corpus de requêtes traité.

6 Conclusions et perspectives

Les résultats actuels pour l'estimation de difficulté sont prometteurs. L'effet de l'enrichissement étant lié à la précision obtenue, il semble probable qu'on puisse arriver à des prédiction exploitables, ce qui se traduirait directement en une meilleure performance du système utilisant ces informations.

Le domaine «questions/réponses» semble se prêter particulièrement à notre approche, car au-delà des attributs que nous avons utilisés pour la tâche «adhoc», une grande quantité d'informations linguistiques pourraient être exploitées.

À plus long terme, il semble intéressant d'exploiter les informations extraites pour diriger un dialogue avec l'utilisateur pour les requêtes qui ne peuvent pas être traitées correctement de manière automatique.

Références

JONES K. S., JOURLIN P., JOHNSON S. & WOODLAND P. (2001). The cambridge multimedia document retrieval (mdr) project : Summary of experiments.

KUHN R. & DE MORI R. (1995). *The Application of Semantic Classification Trees to Natural Language Understanding*, volume 17, chapter 5, p. 449–460. IEEE Transactions on Pattern Analysis and Machine Intelligence.

ROBERTSON S., WALKER S., JONES S., HANCOCK-BEAULIEU M. & GATFORD M. (1996). Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*.

WALKER S. & DE VERE R. (1990). Improving subject retrieval in online catalogues : 2. relevance feedback and query expansion. *British Library Research Paper 72*.

WITTEN I. H. & FRANK E. (1999). *Data Mining : Practical machine learning tools with Java implementations*. San Francisco : Morgan Kaufmann.