Recherche d'informations dans un environnement distribué

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique, Pierre-à-Mazel 7, 2000 Neuchâtel (Suisse)
{Jacques.Savoy, Yves.Rasolofo}@unine.ch
www.unine.ch/info/

In actes "Traitement Automatique de la Langue Naturelle, TALN 2000
Lausanne (Suisse), octobre 2000, pp. 317-326.

Résumé

Le Web ou les bibliothèques numériques offrent la possibilité d'interroger de nombreux serveurs d'information (collections ou moteurs de recherche) soulevant l'épineux problème de la sélection des meilleures sources de documents et de la fusion des résultats provenant de différents serveurs interrogés. Dans cet article, nous présentons un nouvelle approche pour la sélection des collections basée sur les arbres de décision. De plus, nous avons évalué différentes stratégies de fusion et de sélection permettant une meilleure vue d'ensemble des différentes solutions.

Abstract

The Web and digital libraries offer the possibility to send natural language queries to various information servers (corpora or search engines) raising the difficult problem of selecting the best document sources and merging the results provided by different servers. In this paper, a new approach for collections selection based on decision trees is described. Moreover, different merging and selection procedures have been evaluated leading to an overview of the suggested approaches.

1. Introduction

Les bibliothèques numériques et le Web connaissent un développement grandissant mais face au nombre considérable de pages, la navigation a elle seule ne peut plus être considérée comme l'outil adéquat pour dépister de l'information, même avec l'introduction de différentes listes thématiques. Un mécanisme basé sur des requêtes écrites en langue naturelle doit être fourni aux usagers afin de rechercher efficacement les documents souhaités. Les moteurs de recherche (Gordon & Pathak 1999) ont permis au Web de grandir dans les proportions que nous lui connaissons. Très souvent placés en tête de liste des sites les plus visités, ils sont utilisés par 85 % des utilisateurs comme premier outil de recherche d'informations (Lawrence & Giles 1999). Mais, dans le cadre du Web pour le moins, ces moteurs possèdent plusieurs inconvénients (Hawking & Thistlewaite 1999) et, pris individuellement, ils sont loin de couvrir toute l'information disponible sur la Toile. Ne pouvant pas représenter toute l'information disponible à l'aide d'un seul fichier inversé (par exemple, à cause de la taille maximale de 2 GB par fichier),

ces moteurs disposent de plusieurs fichiers inversés que l'on peut voir comme un ensemble de collections distribuées gérées par la même stratégie de recherche.

La présence d'un tel ensemble de collections distribuées pose des problèmes similaires aux bibliothèques numériques. Afin de les interroger, un dépistage efficace d'informations doit :

- découvrir les collections de documents disséminées à travers le réseau;
- sélectionner les "meilleurs" serveurs d'information auxquels la requête sera envoyée;
- convertir la requête soumise dans un format approprié pour chacun des serveurs sélectionnés (par exemple, en recourant au protocole Z90.50 ou au modèle STARTS (Gravano *et al.* 1997));
- sélectionner et trier les différentes listes de résultats provenant des serveurs interrogés afin de présenter une liste unique à l'usager (problème de fusion de collections).

Dans cet article, nous nous intéressons à proposer une réponse au deuxième et quatrième problème dans le contexte où les serveurs d'information utilisent la même stratégie d'indexation et de dépistage. De plus, les différents corpus utilisés forment chacune un ensemble logique (ensemble des documents provenant d'une source précise, d'un journal ou d'une agence gouvernementale). Cette situation reflète bien la situation d'une bibliothèque numérique disposant de plusieurs sources et utilisant le même logiciel pour gérer l'ensemble de ses fonds documentaires. Finalement, la résolution de ces deux problèmes nous permettra également de manier efficacement une collection très volumineuse de documents en la divisant en différents entités.

2. Sélection et fusion de collections

Afin d'évaluer expérimentalement nos propositions, nous avons choisi un corpus de documents rédigés en langue anglaise correspond à celui de la huitième conférence TREC, corpus nommé "TREC8". Cet ensemble comprend 528'155 documents (pour un volume de 1'904 MB) se divisant logiquement en quatre collections soit des articles du *Financial Times* (FT, 210'158 documents), des documents du *Federal Register* (FR, 55'630 documents), des dépêches du *Foreign Broadcast Information Service* (FBIS, 130'471 documents) et des articles du journal *Los Angeles Times* (LA Times, 131'896 documents). Des statistiques plus complètes sur ces collections sont décrites dans l'annexe 2. Avec ce corpus, nous disposons de 150 requêtes couvrant non pas un domaine limité mais présentant un éventail assez large de thèmes (par exemple "rabies", "journalist risks", "food/drug laws", "hydrogen energy", "piracy"). Comme l'ordinateur ne possède que le titre des besoins d'information, le texte disponible s'avère très bref, comportant en moyenne, 2.4 mots (erreur-type de la moyenne 0.65). De plus, les termes employés sont très souvent ambigus et très fréquents dans les documents disponibles.

Comme méthodologie d'évaluation, nous avons retenu la précision moyenne (Salton & McGill 1983) mesurant la qualité de la réponse fournie par l'ordinateur, mesure utilisée par la conférence TREC. Finalement, pour décider si un système de dépistage est meilleur qu'un autre, on admet comme règle d'usage qu'une différence de 5% dans la précision moyenne peut être considérée comme significative.

2.1. Performance des collections individuelles

Afin d'obtenir un panorama assez complet de la qualité des réponses fournies par différentes stratégies de dépistage avec nos quatre sous-collections et pour le corpus dans son ensemble, nous avons utilisé le logiciel SMART pour implanter diverses variantes du modèle vectoriel (Buckley *et al.* 1996) ainsi que le modèle probabiliste Okapi (Robertson *et al.* 1995).

Dans toutes nos expériences, le score de chaque document (ou son degré de pertinence jugé par la machine) est obtenu par le calcul du produit interne. Par exemple, pour l'approche "doc=bnn, requête=bnn" (indexation binaire notée synthétiquement "bnn-bnn"), ce score indiquera le nombre de termes communs entre le document et la requête. Pour l'approche "doc=nnn, requête=nnn", ce score tiendra compte de la fréquence d'occurrence des termes communs entre le document et la requête. Pour le modèle vectoriel classique "doc=ntc, requête=ntc", l'indexation tient compte à la fois de la fréquence d'occurrence dans le document et de l'inverse de la fréquence documentaire (nombre de documents dans lesquels le terme apparaît). De plus, dans cette stratégie "ntc-ntc", les poids sont normalisés selon la formulation du cosinus. D'autres stratégies tiendront compte également de la longueur du document ou de la requête (voir annexe 1).

La table 1 indique la précision moyenne des différentes stratégies retenues. Les résultats démontrent que la performance moyenne est très souvent plus élevée dans chacune des sous-collections que dans le corpus entier (colonne TREC8). On notera toutefois que d'une collection à l'autre le nombre de requêtes ainsi que le nombre de documents pertinents varient. Nous pensons que ces différences de performance laissent penser qu'un bon système de sélection peut améliorer la performance d'une interrogation en langue naturelle.

	Précision moyenne (% changement)				
collection	FT	FR	FBIS	LA TIMES	TREC8
# doc. pert.	4'903	844	4'410	3'535	13'692
modèle	144 requêtes	69 requêtes	116 requêtes	143 requêtes	150 requêtes
Okapi - npn	28.52	29.69	25.48	24.10	22.28
atn - ntc	25.57 (-10.3)	29.24 (-1.5)	25.41 (-0.3)	23.24 (-3.6)	20.99 (-5.8)
Lnu - ltc	25.54 (-10.5)	18.89 (-36.4)	21.48 (-15.7)	22.15 (-8.1)	19.49 (-12.5)
lnc - ltc	18.16 (-36.3)	14.28 (-51.9)	18.11 (-28.9)	16.00 (-33.6)	13.29 (-40.4)
ltc - ltc	18.11 (-36.5)	19.05 (-35.8)	19.40 (-23.9)	15.06 (-37.5)	13.20 (-40.8)
ntc - ntc	17.57 (-38.4)	14.39 (-51.5)	15.32 (-39.9)	14.94 (-38.0)	12.20 (-45.2)
anc - ltc	16.18 (-43.3)	16.61 (-44.1)	16.72 (-34.4)	14.84 (-38.4)	11.72 (-47.4)
bnn - bnn	14.12 (-50.5)	15.04 (-49.3)	14.15 (-44.5)	11.72 (-51.4)	11.02 (-50.5)
lnc - lnc	12.94 (-54.6)	8.44 (-71.6)	13.98 (-45.1)	13.12 (-45.6)	8.70 (-61.0)
nnn - nnn	9.51 (-66.7)	8.83 (-70.3)	10.28 (-59.7)	9.01 (-62.6)	5.63 (-74.7)

Table 1 : Précision moyenne des moteurs de recherche sur les différentes collections

En résumé, on notera que le modèle probabiliste Okapi présente la meilleure précision moyenne. En deuxième position, on trouve l'approche "atn-ntc" tandis que le modèle vectoriel classique "ntc-ntc" occupe seulement le sixième rang. Nous avons été étonné de trouver la stratégie binaire "bnn-bnn" dans un meilleure position que les approches "nnn-nnn" ou "lnc-lnc".

2.2. Stratégie de sélection de collections

Face à plusieurs sources de documents et sur la base d'une requête, la machine doit d'abord décider quelles collections elle doit interroger. Dans ce but, plusieurs études antérieures suggèrent de classer les différents corpus selon le nombre potentiel de documents pertinents qu'ils devraient contenir. Pour atteindre cet objectif, Voorhees *et al.* (1995) proposent de calculer la similarité entre la requête courante et les requêtes passées et ainsi déduire le nombre de documents à extraire de la liste proposée par chaque corpus. Pour Callan *et al.* (1995) et Xu et Callan (1998), le classement des différentes collections en fonction d'une requête correspond au travail habituel d'un moteur de recherche. En effet, au lieu de classer des documents selon leur

degré de similarité avec la requête, le système doit classer des collections en fonction de la requête. Dans ce but, la formule suivante est utilisée :

$$\text{score}(t_j \mid db_i) = \text{defB} + (1 - \text{defB}) \cdot \frac{df_i}{df_i + K} \cdot \frac{\log\left(\frac{db + 0.5}{cf_j}\right)}{\log(db + 1)} \quad \text{et} \quad K = k \cdot \left((1 - b) + b \cdot \frac{ldb_i}{avldb}\right)$$

dans laquelle t_j un terme de la requête, db_i désigne la i^e collection, df_i le nombre de documents dans la i^e collection contenant le terme t_j, DB le nombre de collections, ldb_i le nombre de termes contenus dans le i^e corpus, avldb la moyenne des ldb_i, defB, b et k étant des constantes. Comme valeurs possibles à ces constantes, Xu et Callan (1998) proposent defB=0.4, k=200 et b=0.75. Comme l'expression précédente s'applique pour un seul terme, le score obtenue pour une collection désignée est simplement la moyenne de ces valeurs pour tous les termes de la requête.

Hawking et Thistlewaite (1999) suggèrent d'interroger toutes les collections en leurs soumettant une requête courte composée de deux termes sélectionnés et reflétant au mieux le centre d'intérêt de la requête courante. Ces auteurs proposent de combiner de manière linéaire différentes statistiques obtenus en réponse à ces requêtes courtes afin de classer les différents serveurs en fonction du nombre estimé d'articles pertinents contenus dans chaque serveur. Finalement Fuhr (2000) présente un modèle théorique dans lequel le nombre d'articles à extraire de chaque corpus peut se calculer en fonction de coût de traitement de la requête, du coût associé au dépistage de documents pertinents ou non, de la qualité des moteurs de recherche (basé sur une estimation de leur courbe de précision-rappel) et du nombre estimé de documents pertinents contenu dans chaque collection.

Les pistes énoncées ne répondent pas vraiment à une sélection mais présentent souvent un classement des différentes sources en fonction de la requête courante. Sur ce résultat, on peut dès lors interroger les m meilleurs corpus ou ceux dont le score dépasse un seuil fixé arbitrairement. Rejetant ces deux premières solutions, Callan *et al.* (1995) suggèrent de classifier les collections en fonction de leur score en utilisant l'algorithme du "single-pass" et de sélectionner le ou les deux meilleurs groupes ainsi obtenus.

2.3. Fusion de collections

Afin de fusionner les différentes listes de résultats provenant des sources sélectionnées pour en présenter une seule à l'usager, différentes propositions ont été avancées. Comme première approche, nous pouvons admettre que chaque corpus retenu contient un nombre approximativement égal de documents pertinents et que ceux-ci se retrouvent distribués de manière identique dans les réponses obtenues des serveurs (Voorhees *et al.* 1995). Selon ces hypothèses, nous pouvons construire la liste finale en prenant un élément dans chaque liste puis en recommençant. Cette stratégie "à chacun son tour" se rencontre souvent dans des meta-moteurs (Selberg 1999) et permet d'obtenir une précision moyenne d'environ 40 % inférieure à l'interrogation d'un collection unique regroupant tous les documents (Voorhees *et al.* 1995; Callan *et al.* 1995). La dernière ligne de la table 2 confirme en partie cette conclusion en indiquant une perte moyenne de 21.72 % par rapport à un système interrogeant la collection dans son ensemble.

Cependant, en plus du rang de chaque article dépisté, les moteurs de recherche fournissent parfois un score ou un degré de similarité entre la requête et le document. On peut admettre que cette mesure obtenue par la même stratégie d'indexation et le même moteur de recherche présente des valeurs comparables entre les différents corpus (Kwok *et al.* 1995). Le tri des articles

provenant des différents serveurs peut donc s'opérer sur la base de ce score et nous nommerons cette stratégie "fusion par le score". Cependant, Dumais (1994) a indiqué que plusieurs statistiques dépendent des collections (par exemple, la valeur idf retenue dans la pondération des documents ou de la requête) et ces valeurs peuvent varier fortement d'un corpus à l'autre. Ce phénomène risque d'invalider cette approche. Pourtant les résultats indiqués dans la table 2 démontrent qu'une stratégie de fusion par le score ne détériore que peu la performance moyenne (perte moyenne de 4.61 %). Cette expérience indique qu'une telle approche semble être valide lorsqu'une collection très volumineuse est distribuée sur un réseau et qu'elle est interrogée par le même moteur de recherche.

	Précision moyenne					
stratégie modèle	collection unique	à chacun son tour	score	score normalisé	CORI k=200,b=.75 defB=.4	régression logistique
Okapi - npn	22.28	16.21	21.40	16.73	20.19	20.45
atn - ntc	20.99	15.64	20.05	15.63	18.50	19.97
Lnu - ltc	19.49	14.35	17.72	13.51	16.26	17.96
lnc - ltc	13.29	10.05	12.35	9.83	9.46	12.04
ltc - ltc	13.20	10.50	12.23	10.41	11.37	12.78
ntc - ntc	12.20	9.35	11.46	9.38	10.59	11.48
anc - ltc	11.72	9.48	10.81	9.27	10.01	10.89
bnn - bnn	11.02	7.93	10.99	9.78	9.85	10.94
lnc - lnc	8.70	7.03	8.59	6.93	8.06	8.80
nnn - nnn	5.63	5.43	5.70	5.84	5.75	6.01
perte moyenne en %		-21.72 %	-4.61 %	-20.02 %	-12.42 %	-3.93 %

Table 2 : Précision moyenne de différentes stratégies de fusion

Comme troisième approche, nous pouvons normaliser le score obtenue par chaque document dans chaque collection en le divisant par le score maximum obtenue pour le corpus considéré (stratégie du score normalisé). Les expériences menées indiquent une baisse de la précision moyenne d'environ 20 %. Comme quatrième stratégie, Callan *et al.* (1995) suggèrent dans leur système CORI de tenir compte du score s_i obtenue par chaque collection lors du processus de sélection (voir section 2.2). Ainsi, le score de chaque document extrait de la i^e collection est multiplié par un coefficient w_i calculé comme suit :

$$w_i = 1 + DBs \cdot [(s_i - S_m) / S_m]$$

dans laquelle DBs indique le nombre de corpus sélectionnés, s_i le score obtenue par la i^e collection considérée et S_m le score moyen des collections. Selon nos expériences, la performance moyenne présente une baisse moyenne de 12.42 %.

Finalement, la fusion peut s'effectuer selon la probabilité que le document soit pertinent, probabilité estimée selon la méthode de la régression logistique (Bookstein *et al.* 1992) sur la base du rang et du score obtenue par ce document (Le Calvé & Savoy 2000). Dans nos expériences, la performance obtenue est légèrement inférieure à celle obtenue en construisant un seul corpus avec l'ensemble des documents (perte moyenne de 3.93 %).

En résumé, on constate que la fusion par le score présente une performance intéressante et similaire à la meilleure approche étudiée soit la régression logistique. Le modèle CORI calculé sans sa procédure de sélection occupe la troisième place et les stratégies "à chacun son tour" ou du score normalisé doivent être abandonnée.

3. Arbre de décision

Afin de définir l'ensemble des corpus à interroger, la question que doit répondre chaque serveur est la suivante: "Est-ce que cette collection (avec son propre moteur de recherche) peut répondre de manière satisfaisante (avec un ou plusieurs documents pertinents) à la requête courante". A notre point de vue, le nombre précis d'articles à extraire des différents serveurs sélectionnés est du ressort de la procédure de fusion.

Pour permettre à la machine de répondre automatiquement à cette question, nous avons opté pour les arbres de décision, et plus précisément pour l'algorithme C4.5 (Quinlan 1993; Mitchell 1997). Cette approche possède l'avantage de tenir compte de tous les exemples d'apprentissage présentés afin de dériver un arbre de décision ou un ensemble de règles (SI (*conditions*) ALORS *décision*) pour chaque serveur d'information potentiel. De plus, avec cette méthode d'apprentissage automatique, le processus de décision peut aisément être compris par un être humain. Approche robuste face à des données bruitées, elle évite grâce à une procédure d'élagage a posteriori ("post-pruning") de générer un ensemble de règles trop proches des caractéristiques sous-jacentes de l'ensemble d'apprentissage. Finalement, cette dernière opération possède l'avantage de réduire la taille de l'arbre de décision et de générer les règles moins spécifiques.

Afin d'utiliser les arbres de décision, nous devons fournir à l'ordinateur un ensemble d'exemples, chacun possédant une série de couples attribut - valeur. Dans notre contexte, nous avons choisi plusieurs attributs potentiellement utiles comme :

- le nombre de termes inclus dans la requête, noté longR;
- le nombre de documents dans la collection (pour les trois termes de la requête les moins fréquents), nombre noté df1, df2 et df3;
- la moyenne des valeurs df des termes de la requête, notée moyenne;
- le degré de similarité des trois premiers documents extraits (RSV1, RSV2, et RSV3);
- l'écart normalisé du degré de similarité pour les trois premiers rangs sachant que la collection possède plusieurs documents pertinents, soit (RSV - x) / écart-type, noté dr1, dr2 et dr3.

De plus, nous avons considéré que l'approche des k plus proches voisins ("k-NN") présente un attribut intéressant pour la construction de nos arbres de décision car cette approche a déjà prouvée sa bonne performance (Michie *et al.* 1994) et son utilité dans notre contexte (Voorhees *et al.* 1995). Dans cette méthode, le système calcule la similarité entre la nouvelle requête et toutes les autres requêtes (149 dans notre cas). La machine retient les trois plus proches voisins (k = 3). L'attribut noté nn peut donc prendre des valeurs comprises entre 0 et 3 avec 0 indiquant que, pour les trois plus proches requêtes, aucune ne retournait des documents pertinents. Une valeur de deux indiquent que pour deux des trois requêtes les plus proches, il existait des documents pertinents.

Pour les collections FT et LA Times, l'arbre de décision obtenue indique qu'il faut toujours les interroger. En effet, pour ces deux corpus, le nombre de requêtes ne possédant pas de documents pertinents est faible (6 requêtes pour FT et 7 pour LA Times). Pour les deux autres serveurs, les arbres de décision varient un peu selon le modèle de recherche. A titre d'exemple, pour la collection FR et le moteur Lnu-ltc, nous avons obtenu l'arbre suivant:

SI (dr1 ≤ -0.6177 & nn ≤ 1) ALORS ne pas interroger;
SI (dr1 ≤ -0.6177 & nn > 1) ALORS interroger la collection;

SI (dr1 > -0.6177 & longR ≤ 1 & moyenne > 222) ALORS ne pas interroger;
SI (dr1 > -0.6177 & longR ≤ 1 & moyenne ≤ 222 & RSV3 ≤ 9) ALORS ne pas interroger;
SI (dr1 > -0.6177 & longR ≤ 1 & moyenne ≤ 222 & RSV3 > 9) ALORS interroger;

SI (dr1 > -0.6177 & longR > 1 & nn ≤ 0) ALORS ne pas interroger;
SI (dr1 > -0.6177 & longR > 1 & nn > 0) ALORS interroger;

Dans le cas présent, l'écart normalisé du degré de similarité du premier document extrait (dr1), la valeur de l'attribut nn et la longueur de la requête (longR) jouent les premiers rôles dans la décision tandis que la moyenne des valeurs df (moyenne) et la similarité du troisième article dépisté (RSV3) jouent les seconds rôles. Ainsi, par exemple, la dernière règle indique que le système doit prendre en compte ce corpus si l'écart normalisé est plus grand que -0.6177, que la requête possède plus qu'un terme et qu'il existe au moins une requête antérieure proche de la requête actuelle (similarité calculée selon les mots apparaissant dans les requêtes).

	Précision moyenne (% changement)				
stratégie modèle	collection unique	pas de sélection	sélection optimale	CORI k=200, b=0.75 defB = 0.4	arbre de décision
Okapi - npn	22.28	21.40 (-3.95)	22.58 (+1.35)	20.06 (-9.96)	21.76 (-2.33)
atn - ntc	20.99	20.05 (-4.48)	21.39 (+1.91)	18.49 (-11.91)	20.57 (-2.00)
Lnu - ltc	19.49	17.72 (-9.08)	19.23 (-1.33)	16.21 (-16.83)	18.60 (-4.57)
lnc - ltc	13.29	12.35 (-7.07)	13.65 (+2.71)	9.41 (-29.19)	12.84 (-3.39)
ltc - ltc	13.20	12.23 (-7.35)	13.33 (+0.98)	11.33 (-14.17)	12.53 (-5.08)
ntc - ntc	12.20	11.46 (-6.07)	12.59 (+3.20)	10.52 (-13.77)	11.99 (-1.72)
anc - ltc	11.72	10.81 (-7.76)	12.20 (+4.10)	10.03 (-14.42)	11.52 (-1.71)
bnn - bnn	11.02	10.99 (-0.27)	11.15 (+1.18)	9.71 (-11.89)	10.99 (-0.27)
lnc - lnc	8.70	8.59 (-1.26)	9.26 (+6.44)	8.05 (-7.47)	8.73 (+0.34)
nnn - nnn	5.63	5.70 (+1.24)	6.46 (+14.74)	5.68 (+0.89)	6.10 (+8.35)
gain / perte moyen en %		-4.61 %	+3.53 %	-12.87 %	-1.24 %

Table 3 : Précision moyenne de différentes stratégies de sélection

Les résultats présentés dans la table 3 indique que l'absence de sélection (fusion par la score) présente une légère dégradation de la précision moyenne comparée à la gestion d'une collection unique comprenant tous les documents. Le recours à une sélection optimale (le système connaissant toujours toutes les serveurs possédant au moins un document pertinent) permet d'accroître légèrement la performance moyenne. Le modèle CORI avec sa procédure de sélection possède une précision moyenne significativement inférieure à l'absence de sélection. Ce résultat est plutôt décevant sachant que la sélection appliquée n'a pas éliminé beaucoup de corpus (nombre moyen de corpus interrogés 3.78 sur un maximum de 4). Notre approche recourant aux arbres de décision (fusion par le score) propose une alternative très intéressante au niveau de la performance moyenne dont la différence n'est pas significative avec celle obtenue par l'interrogation d'une collection comprenant tous les articles.

4. Conclusion

Sur la base de nos expériences, les conclusions suivantes peuvent être tirées ;

1. le modèle probabiliste Okapi présente une performance très attractive;
2. le modèle vectoriel classique "ntc-ntc" ne propose pas une stratégie d'indexation et de dépistage efficace;
3. la fusion par la score offre une alternative intéressante lorsque l'on doit fusionner plusieurs listes de résultats obtenues par le processus d'indexation et le même moteur de recherche;
4. les arbres de décision proposent une alternative intéressante pour la sélection de collections.

De plus, le contexte de nos expériences se distinguent des travaux antérieures par le fait que la division de notre corpus de départ s'effectue sur une base logique (ensemble de documents provenant de la même source), par le nombre plus restreint de sous-collections prises en compte, et la longueur extrêmement restreinte des requêtes. Notre approche propose également de fournir une explication à l'usager des raisons qui ont conduit le système à sélectionner tel ou tel serveur plutôt que tel autre.

On peut également élargir le modèle proposée en considérant une hiérarchie de collections dans laquelle, à chaque niveau, on rencontre les problèmes de sélection et de fusion ouvrant ainsi la voie à la présence d'un très grand nombre de corpus ou serveurs distribués sur un réseau.

Nos travaux en cours tendent de résoudre les mêmes problèmes de sélection et de fusion en admettant que les différentes collections sont gérées par des stratégies d'indexation et de recherche différentes, situation classique des meta-moteurs de recherche, et en supposant que nous n'avons pas de requêtes antérieures pour paramétrer le modèle; dans le cas contraire, voir (Le Calvé & Savoy 2000).

Remerciements

Cette recherche a été subventionnée en partie par le FNS avec le subside 21-58 813.99.

Références

BOOKSTEIN A., O'NEIL E., DILLON M., STEPHEN D. (1992). Applications of Loglinear Models for Informetric Phenomena. *Information Processing & Management*, Vol. 28, pp. 75-88.

BUCKLEY C., SINGHAL A., MITRA M., SALTON G. (1996). New Retrieval Approaches using SMART. In D. Harmann, Ed., *Proceedings of the TREC'4*, pp. 25-48.

CALLAN J.P., LU Z., CROFT W.B. (1995). Searching Distributed Collections with Inference Networks. In E. Fox, P. Irgwersen, R. Fidel, Ed., *Proceedings of the ACM-SIGIR'95*, pp. 21-28.

DUMAIS S.T. (1994). Latent Semantic Indexing (LSI) and TREC-2. In D. Harmann, Ed., *Proceedings of TREC'2*, pp. 105-115.

FUHR N. (2000). A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems*, 2000, to appear.

GORDON M., PATHAK P. (1999). Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines. *Information Processing & Management*, Vol. 35, pp. 141-180.

GRAVANO L., CHANG K., GARCÍA-MOLINA H., LAGOZE C., PAEPCKE A. (1997). *STARTS - Stanford Protocol Proposal for Internet Retrieval and Search*. Computer Systems Laboratory, Stanford University, Stanford (CA), (voir http://www-db.stanford.edu/~gravano/starts_home.html).

HAWKING D., THISTLEWAITE P. (1999). Methods for Information Server Selection. *ACM Transactions on Information Systems*, Vol. 17, pp. 40-76.

KWOK K.L., GRUNFELD L., LEWIS D.D. (1995). TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. In D. Harmann, Ed., *Proceedings of TREC'3*, pp. 247-255.

LAWRENCE S., GILES C. L. (1999). Accessibility of Information on the Web. *Nature*, Vol. 400, pp. 107-110.

LE CALVÉ A., SAVOY J. (2000). Database Merging Strategy Based on Logistic Regression. *Information Processing & Management*, Vol. 36, pp. 341-359.

POWELL A.L., FRENCH J.C., CALLAN J., CONNELL M., VILES C.L. (2000). The Impact of Database Selection on Distributed Searching. In *Proceedings of the ACM-SIGIR'2000*, to appear.

MICHIE D., SPIEGELHALTER D.J., TAYLOR C.C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood: New York (NY).

MITCHELL T.M. (1997). *Machine Learning*. McGraw-Hill: New York (NY).

QUINLAN J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann: Los Angeles (CA).

ROBERTSON S.E., WALKER S., HANCOCK-BEAULIEU M.M. (1995). Large Test Collection Experiments on an Operational, Interactive System: OKAPI at TREC. *Information Processing & Management*, Vol. 31, pp. 345-360.

SALTON G., MCGILL M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill: New York (NY).

SELBERG E.W. (1999). *Towards Comprehensive Web Search*. Ph.D. Thesis, University of Washington.

VOORHEES E.M., GUPTA N.K., JOHNSON-LAIRD B. (1995). Learning Collection Fusion Strategies. In E. Fox, P. Irgwersen, R. Fidel, Ed., *Proceedings of the ACM-SIGIR'95*, pp. 172-179.

XU J., CALLAN J.P. (1998). Effective Retrieval with Distributed Collections. In W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, J. Zobel, Ed., *Proceedings of the ACM-SIGIR'98*, pp. 112-120.

Annexe 1: Formules de pondération

Afin d'attribuer un poids w_{ij} reflétant l'importance du terme t_j dans la description du document d_i, trois facteurs sont pris en compte, chacun étant représenté par une lettre, à savoir:

1. la fréquence d'occurrence du terme t_j dans le document d_i, désignée par tf_{ij} (première lettre);
2. la fréquence documentaire (nombre de documents dans lesquels le terme t_j apparaît), désignée par df_j (deuxième lettre);
3. la normalisation des poids (troisième lettre).

Dans la table A.1, n indique le nombre de documents dans la collection, la longueur d'un document d_i (mesurée par le nombre de termes d'indexation) est notée par nt_i, le facteur pivot = 150 et la constante slope = 0.2. Finalement, le modèle probabiliste Okapi repose sur la pondération suivante:

$$w_{ij} = \frac{(k_1 + 1) \cdot tf_{ij}}{K + tf_{ij}} \qquad \text{avec } K = k \cdot \left((1 - b) + b \cdot \frac{l_i}{avdl}\right)$$

dans laquelle K représente le rapport entre la longueur du document d_i mesurée par l_i (la somme de ses tf_{ij}) et la longueur moyenne des documents d'un corpus, moyenne notée par advl (dans cette étude, advl = 750, b = 0.9, k = 2).

n	new_tf = tf_{ij} (nombre d'occurrences de t_j dans d_i)
b	new_tf = pondération binaire (0 ou 1)
a	new_tf = $0.5 + 0.5 \cdot (tf_{ij} / \max tf \text{ dans } d_i)$
l	new_tf = $\ln(tf_{ij}) + 1.0$
L	new_tf = $[\ln(tf_{ij}) + 1.0] / [1.0 + \ln(\text{moyenne (tf dans } d_i))]$
n	new_wt = new_tf (pas de modification)
t	new_wt = new_tf $\cdot \ln[n / df_j]$
p	new_wt = new_tf $\cdot \ln[(n - df_j) / df_j]$
n	w_{ij} = new_wt (pas de normalisation)
c	diviser chaque new_wt par sqrt (somme (new_wts ^2))
u	w_{ij} = new_wt / [(1-slope) $\cdot$ pivot + slope $\cdot$ nt_i]

Table A.1: *Formules de pondération*

Annexe 2: Statistiques sur les collections

Collection	FT	FR	FBIS	LA Times	TREC8
Taille (en MB)	564 MB	395 MB	470 MB	475 MB	1'904 MB
# de documents	210'158	55'630	130'471	131'896	528'155
# de formes	375'499	196'220	502'099	337'492	1'008'463
# formes / doc.					
moyenne μ	124.4	131.16	141.56	158.46	136.84
erreur-type de μ	93.26	127.95	125.04	124.11	114.54
médiane	101	128	107	122	108
maximum	3'050	23'517	5'677	5'040	23'515
minimum	6	2	6	10	2
# d'occurr. / doc.					
moyenne μ	195.62	320.11	267.2	262.86	240.89
erreur-type de μ	172.66	1,128.3	598.82	248.5	501.35
médiane	151	289	168	184	171
maximum	13'761	211'944	61'300	16'100	211'934
minimum	6	2	10	10	2
requête 301 à 450					
# doc. pertin.	4'903	844	4'410	3'535	13'692
# de requêtes	144	69	116	143	150

Table A.2: Quelques statistiques sur les collections utilisées