Méthodes pour la sélection de collections dans un environnement distribué

Faïza Abbaci[1], Jacques Savoy[2], Michel Beigbeder[3]

[1]*Ecole Nationale Supérieure des Mines de Saint-Etienne,*
158 Cours Fauriel, 42023 Saint Etienne CEDEX 2.
abbaci@emse.fr

[2]*Université de Neuchâtel, Pierre-à-Mazel 7, CH-2000 Neuchâtel*
Jacques.Savoy@unine.ch

[3]*Ecole Nationale Supérieure des Mines de Saint-Etienne,*
158 Cours Fauriel, 42023 Saint Etienne CEDEX 2.
mbeig@emse.fr

Résumé :

Nous explorons dans cet article trois approches de sélection de collections dans un environnement de recherche d'informations distribuée. Le processus de recherche se fait par l'intermédiaire d'un courtier qui pour une requête donnée sélectionne les collections à interroger et fusionne les résultats qu'elles retournent. Notre première approche de sélection consiste à classer les collections selon leur pertinence à la requête posée, les *n* premières collections sont alors interrogées. La seconde approche sélectionne les collections dont le score dépasse un certain seuil. Enfin, la troisième approche définit le nombre de documents à rechercher dans chaque collection. L'originalité de notre démarche est qu'elle utilise des données récoltées au moment de l'interrogation et ne repose pas sur des méta-données sauvegardées *a priori* au niveau du courtier comme c'est le cas de la plupart des méthodes connues dans la littérature. Afin d'évaluer nos approches et les comparer aux autres techniques notamment l'approche centralisée (à index unique) et CORI [CALL95] [XU98], nous avons conduit des expérimentations sur la collection de test WT10g, et les gains sont appréciables.

MOTS-CLES : Recherche d'informations distribuée, sélection de collections, évaluation, classement de collections.

Abstract :

We investigate in this paper three collection selection approaches within a distributed information retrieval environment. The first approach consists of sorting servers according to their usefulness and selecting the top N_ranked servers from which to obtain the final result list. The second selects those servers whose score exceeds a given threshold. Finally, the third determines the number of documents to be retrieved from each server. Our method differs from that of previous works in that most of these works were based on pre-stored metadata, our approach only requires information extracted while the request is being processed. Experiments were conducted to evaluate the effectiveness of our approaches and they were compared both with centralized indexing and various other DIR techniques (e.g., CORI [CALL95] [XU98]). Preliminary results from these experiments, conducted on the WT10g test collection, tend to demonstrate that our suggested methods can achieve appreciable retrieval effectiveness.

1. Introduction

Actuellement, d'innombrables systèmes de recherche d'informations ont vu le jour pour faire face au développement spectaculaire de l'Internet. Ces systèmes reposent tous sur le même principe : la construction d'un index centralisé qui sert de base pour le processus de recherche. La centralisation de l'index constitue la limite de ce type de systèmes, vu la croissance exponentielle de l'information disponible en ligne : ce problème est communément appelé *problème de scalabilité.*

Pour faire face à ce problème, distribuer le processus de stockage et de recherche constitue vraisemblablement le moyen le plus adapté. Un système de recherche d'informations distribuée est typiquement constitué d'un module qu'on appelle le courtier. A la réception d'une requête, le courtier se charge de sélectionner puis d'interroger un certain nombre de serveurs (ou collections) parmi ceux qu'il connaît. Chacun de ces serveurs effectue la tâche qui lui est demandée et retourne une liste de réponses. Le courtier se charge alors de les fusionner et présente à l'utilisateur une seule et unique liste de réponses.

Dans cet article nous présentons trois approches de sélection de serveurs. La première consiste à attribuer un score à chaque serveur qui servira à les classer, la deuxième sélectionne les serveurs dont le score dépasse un certain seuil, et enfin la dernière approche définit le nombre approprié de documents à rechercher de chacun des serveurs. Alors que Towell [TOWE95] considère cette dernière méthode comme étant une technique de fusion, nous la considérons comme une méthode à mi-chemin entre la sélection de serveurs et la fusion des résultats. De notre point de vue, c'est une méthode de sélection sauf qu'elle n'attribue pas un jugement binaire (pertinence, non pertinence) à un serveur donné contrairement aux autres techniques. Néanmoins, nous considérons aussi cette méthode comme une stratégie de fusion, car elle diminue le bruit en éliminant les documents non-pertinents des listes à fusionner.

Nos trois approches reposent sur l'hypothèse suivante : la pertinence d'une collection est étroitement liée à la pertinence des premiers documents qu'elle retourne. Ainsi pour prédire la pertinence d'une collection nous examinons les *k* premiers documents qu'elle retourne et nous y analysons la proximité des termes de la requête. Ainsi la différence principale qui distingue nos approches de celles de la littérature est qu'elles ne nécessitent pas des connaissances *a priori* sur le contenu des serveurs. En outre, elles n'exigent aucune coopération de la part des serveurs sous-jacents.
Dans la section suivante nous présentons les techniques de sélection connues dans la littérature. Dans la section 3, nous proposons une méthode d'acquisition de données aidant à prédire la pertinence d'un serveur, et nous détaillons nos trois stratégies de sélection. Enfin, dans la section 4 nous évaluons nos approches et discutons les résultats.

2. Etat de l'Art

Un des challenges dans la recherche d'information distribuée est de réduire le coût du processus de la recherche sans pour autant diminuer l'efficacité du système. Ce but peut être partiellement atteint par la réduction de la quantité d'informations transférée via le réseau. Pour ce faire, deux solutions sont envisageables : réduire le nombre de documents à rechercher par serveur, ou réduire le nombre de serveurs à interroger. Nous classifions donc les techniques proposées dans la littérature en deux grandes catégories selon qu'elles choisissent l'une ou l'autre des deux solutions. Concernant la première catégorie, nous avons recensé une seule technique et c'est celle proposée par Towell *et al.* [TOWE95] qui ont développé une méthode d'apprentissage pour définir le nombre de documents avec lesquels un serveur donné devrait contribuer dans la construction de la liste finale.
La plupart des méthodes de la deuxième catégorie trient les collections selon leur pertinence à la requête posée et sélectionnent les *n_serv* serveurs les mieux placés. Le score d'une collection est calculé à partir des informations sur son contenu comme dans les travaux de Xu et Croft [XU99]. Callan *et al.* [CALL95] et Xu et Callan [XU98] proposent Collection Retrieval Inference network (CORI) qui consiste à considérer une collection comme un document gigantesque. Ainsi pour classer les collections il suffit d'appliquer un algorithme classique de classement des documents dans la recherche d'information. Dans GlOSS [GRAV99] le score d'une collection est la somme des scores estimés des documents jugés pertinents de cette même collection. Hawking et Thistlewaite [HAWK99] proposent d'envoyer des requêtes de sonde au moment de l'interrogation. Une requête de sonde est une requête de taille réduite, construite en gardant *p* termes de la requête initiale. Chaque serveur envoie les informations qui lui sont demandées et sur la base desquelles le score des collections sont calculés. La coopération des serveurs est nécessaire dans ce cas. Craswell *et al.* [CRAS00] enrichissent le score de CORI par des informations sur l'efficacité des systèmes de recherche locaux. Ogilvie et Callan [OGIL01] testent l'extension de requête dans la recherche distribuée mais leur résultats sont peu

concluants. Enfin, Yu *et al.* [YU01] suggèrent un classement basé sur les liens hypertextes entre les documents dans un environnement web.

3. Sélection de collections à base de données récoltées au moment de l'interrogation

La caractéristique principale de notre approche est la manière de collecter les informations nécessaires afin de prédire la pertinence d'une collection. Alors que la plupart des travaux antérieurs se basent sur des données globales pré-collectées (descriptions de serveurs, informations statistiques sur les serveurs), nous dégageons l'information requise à travers l'analyse des *n_doc* premiers documents que chaque serveur rend au moment de l'interrogation. Nous supposons donc que les *n_doc* premiers documents sont représentatifs de l'ensemble des documents qu'un serveur retourne à son interrogation.
Nous adoptons les notations suivantes dans la description de notre approche.

C représente l'ensemble $\{S_1, S_2, .., S_{|C|}\}$ des serveurs disponibles.
N_i est l'ensemble des documents à analyser pour un serveur S_i (et dans notre cas $|N\_i| = n\_doc$).
N_C est l'union des N_i pour tout les serveurs.
N_premiers est l'ensemble des documents en tête de N_C trié par l'une de nos approches.
N_C_i est l'ensemble des documents appartenant à N_C et provenant de S_i ($|N_i|$=n).
N_premier_i est l'ensemble des documents dans N_premiers provenant de S_i.

3.1. Calcul du score d'un document

Dans nos approches de sélection, nous commençons d'abord par attribuer un score à chacun des documents de N_C afin de déterminer la pertinence des serveurs. Pour ce faire, nous supposons que le nombre de mots clés de la requête appartenant à un document, la distance entre ces mots clés dans le document et leurs fréquences d'occurrences sont de bons indicateurs de pertinence de ce même document. De cette supposition et en s'inspirant de Lawrence [LAW98] nous calculons le score d'un document comme suit :

$$score(d,q)=(c_1 \cdot nb_d)+(c_2 \cdot ind\_dis(d,q))+\frac{nb\_occ}{c_3}$$

où, pour tout document *d* :

nb_d est le nombre de mots clés de la requête contenus dans d,
nb_occ est le nombre total d'occurrences des mots clès dans d,
ind_dis est un indicateur de distance entre deux termes de la requête dans d, cette fonction retourne une valeur supérieure ou égale à zéro,

c_1, c_2, c_3 sont des constantes positionnées à $c_1 = 100$, $c_2 = 1000$, $c_3 = 1000$ dans nos expérimentations.

Selon la formule proposée par Clarke *et al.* [CLAR95] et en supposons que les deux premiers termes de la requête sont les plus importants mots clés, nous calculons ind_dis, seulement pour ces deux termes, comme suit :

$$ind\_dis(d,q) = \sum_i dis_i$$

où :
dis_i est la longueur (nombre de termes) du i^{eme} bloc de termes de *d* de taille minimale délimité par les deux termes considérés de la requête.
Dans le cas des requêtes à terme unique et selon [LAWR98], ind_dis est l'inverse de la distance (nombre de termes) entre le début du document jusqu'à la première occurrence du mot clé dans le document.
Enfin, la valeur zéro est attribuée aux documents ne contenant aucun terme de la requête.

3.2. Nos trois approches de sélection de serveurs

Une fois que les documents de N_C sont récupérés, nous calculons leurs scores selon la méthode décrite ci-dessus. Les documents sont alors classés par ordre décroissant de leurs scores. La liste des documents résultante de ce tri sera utilisée dans l'étape de la sélection de collections.

Classement des serveurs (CS)

Notre première méthode de sélection consiste à définir d'abord un degré de pertinence (score) pour les |C| serveurs. Ces derniers sont alors classés par ordre décroissant de leurs scores et les *n_serv* serveurs les mieux placés sont sélectionnés. Les listes des serveurs sélectionnés seront fusionnées pour construire la réponse finale à la requête. Nous posons ici l'hypothèse que les documents de N_premiers sont pertinents, et nous définissons le score s_i d'un serveur S_i comme étant sa contribution dans N_premiers. La contribution de S_i est définie ici comme étant la somme des scores de ses documents dans N_premiers. Plus formellement :

$$s_i = \sum_{d \in N\_premiers\_i} score(d,q)$$

Sélection des serveurs selon un seuil (SS) [ABBA02]

Un serveur S_i est sélectionné si sa contribution, qui ici est définie comme le nombre des documents de S_i, dans N_first est effective (|N_premiers_i|> 1).

Sélection non-binaire(SNB)

Dans cette approche nous écartons le jugement binaire et nous considérons qu'un serveur peut être « très pertinent », « pertinent », « peu pertinent » ou « non

pertinent ». Ainsi, le nombre de documents à extraire de la réponse d'un serveur sera proportionnel à son score, tout en conservant l'idée que le score d'un serveur est proportionnel à sa contribution dans N_premiers. Si L est le nombre de documents demandés par l'utilisateur, nous obtenons alors le nombre Nb_i de documents à rechercher dans S_i comme suit :

$$Nbi = \frac{L \cdot N\_premiers\_i}{|N\_premiers|}$$

4. Etudes expérimentales

4.1. Notre collection de test

Pour mener nos expérimentations, nous avons utilisé la collection de pages Web constituée pour la neuvième conférence TREC, corpus nommé WT10g. Cet ensemble comprend 1 692 096 pages Web écrites en anglais pour un volume de 11033 mégaoctets. Nous disposons de 50 requêtes couvrant des domaines variés (par exemple "parkinson's disease", "hunger", "baltimore", "how e-mail bennefits businesses", "mexican food culture"). Les requêtes correspondent à des exemples réels soumis au moteur de recherche Excite. Elles comportent en moyenne 2.4 mots (écart type de 0.6). De plus, les termes employés sont très souvent ambigus et très fréquents dans les pages Web disponibles. Afin de simuler un environnement distribué, nous avons divisé WT10g en 8 collections équilibrées en taille. La table 1 donne pour chaque collection sa taille en mégaoctets, le nombre de documents qu'elle contient et le nombre de requêtes pour lesquelles elle détient des documents pertinents. Les huit collections ont été indexées par le système SMART avec le modèle probabiliste Okapi [ROBE00].

Table 1. Informations statistiques pour les huit collections.

Collection	Taille (MO)	# docs	# Q
TREC9.1	1 325	207 485	28
TREC9.2	1 474	207 429	44
TREC9.3	1 438	221 916	38
TREC9.4	1 316	202 049	21
TREC9.5	1 309	203 073	22
TREC9.6	1 311	215 451	24
TREC9.7	1 336	200 146	25
TREC9.8	1 524	234 547	43

4.2. Les approches utilisées pour la comparaison

Dans nos expérimentations nous avons comparé nos approches avec les techniques suivantes :

1) *CORI* [CALL95] où le score s_i d'un serveur S_i est calculé comme suit :

$$s_i = \frac{1}{m} \sum_{j=1}^{m} s(t_j|C_i)$$

tel que :

$$s(t_j|C_i) = defB + (1 - defB) \cdot \frac{df_i}{df_i + K} \cdot \frac{log\left(\frac{|C| + 0.5}{cf_j}\right)}{log(|C| + 1.0)}$$

où :

$$K = k \cdot \left[(1-b) + b \cdot \frac{lc_i}{avlc}\right]$$

m est le nombre de termes de la requête q,

|C| est le nombre de serveurs,

df_i est le nombre de documents dans la collection C_i qui contiennent le j^{eme} terme de la requête.

cf_j est le nombre de serveurs contenant le terme t_j,

lc_i le nombre de termes que contient C_i,

avlc la moyenne de lc_i ,

defB, b et k sont des constantes dont le valeurs sont celles suggérées par Xu et Callan [XU98] : defB=0.4, b=0.75 et k=200.

Après le tri des serveurs, une des possibilités de sélection est de choisir pour interrogation les n_serv premiers serveurs, où la valeur de n_serv peut être déterminée par l'utilisateur. Une autre possibilité est l'utilisation d'un algorithme de clusterisation pour regrouper les collections selon leur scores et de sélectionner les collections du premier cluster. Dans nos expérimentations nous considérons la première possibilité.

2) *Sélection Optimale* (Sel-OPT): En se basant sur les données fournies par TREC concernant la totalité de la collection, Sel-OPT sélectionne toutes les collections qui contiennent au moins un document pertinent.

3) *Tri optimal* (Tri-OPT): Les serveurs sont triés selon le nombre de documents pertinents qu'ils contiennent.

4) *L'approche centralisée* (CENTR): Tous les documents sont réunis dans une seule base de données.

5) *L'approche sans sélection* (SANS-Sel): Tous les serveurs sont sélectionnés.

6) *Tri constant* (Tri-CST): Les serveurs sont triés selon le nombre de documents qu'ils contiennent et dans notre cas les serveurs sont triés comme suit : S_8, S_3, S_6, S_1, S_2, S_5, S_4, S_7.

4.3. Evaluation

Pour évaluer nos approches de sélection, nous choisissons d'étudier la précision moyenne atteinte par ces approches. Nous utilisons aussi des mesures d'évaluation spéciales pour évaluer notre approche de tri (CS).

Evaluation générale

Notre méthode d'expérimentation a consisté à combiner chacune des trois approches de sélection proposées ainsi que les approches SANS-Sel et CORI avec la méthode de fusion LMS ("Length Merging Score") [RASO01] qui se base sur la taille des listes de résultats pour effectuer la fusion. Comme les stratégies de sélection sont toutes combinées avec la même méthode de fusion, nous aurons une meilleure idée de l'efficacité de chacune d'elles. Nous avons aussi calculé la précision moyenne atteinte par l'approche centralisée (voir table 2).

La deuxième colonne de la table 2 montre les paramètres de chaque stratégie. Etant donné que, dans notre environnement de test, il y a en moyenne 4.9 collections pertinentes pour chaque requête, nous avons donc choisi de sélectionner les 5 premiers serveurs pour chacune des stratégies de tri à savoir CS et CORI.

Table 2. Comparaison des précisions moyennes obtenues par les différentes approches de sélection.

Strategy	***Param.***	***Prec. Moy***	***Précision obtenu à***					
			5 docs	***10 docs***	***15 docs***	***20 docs***	***30 docs***	***100 docs***
CENTR		19.60	26.67	22.29	19.17	17.40	15.97	10.98
SANS-Sel		19.32	27.92	22.08	19.44	17.60	15.76	10.75
CORI	defB = 0.4 k = 200, b = 0.75, n_serv = 5	*13.66*	*23.33*	*19.58*	*17.78*	16.98	15.42	9.77
CS	n=3, |n_premiers| = 24, n_serv = 5	*17.63*	**30.00**	**24.17**	**20.83**	**18.85**	16.11	*9.00*
SS	n=3, |n_premiers| = 9	19.00	**28.33**	**23.54**	**20.56**	**18.96**	*16.32*	*9.90*
SNB	n=3, |n_premiers| = 9	*17.92*	*26.67*	22.71	**20.28**	**18.96**	15.97	*9.81*

La table 2 montre dans la troisième colonne la précision moyenne obtenue par l'application de chacune des 5 stratégies de sélection. Et dans les colonnes suivantes sont présentées les précisions obtenues à 5, 10, 15, 20, 30 et 100 documents retrouvés. Ces mesures de performances sont utiles car l'utilisateur typique s'intéresse plus aux premiers résultats qu'on lui propose. Avant de poursuivre, nous précisons qu'une différence de précision n'est significative que si elle dépasse 5%.

Nous avons utilisé des attributs typographiques dans la table 2 pour mettre en évidence les améliorations ou les dégradations de performance par rapport à l'approche centralisée. Ainsi, les valeurs en gras déterminent une amélioration significative de performance. Les valeurs en italique montrent une dégradation significative et enfin le reste montre une différence de performance non significative.

Evaluation du classement de serveurs

Concernant l'évaluation des techniques de classement de serveurs : CS, CORI, Tri-CST et Tri-OPT, nous avons utilisé les mesures P_x et R_x [LU96] qui sont assimilées aux mesures conventionnelles de précision et de Rappel, x indique le nombre de serveurs sélectionnés. La différence entre R_x et la mesure de rappel classique est que R_x considère le nombre de documents dans chaque serveurs. Ces mesures sont définies comme suit :

$$P_x = \frac{|\Gamma|}{x} \qquad R_x = \frac{\sum_{i=1}^{x} r_i}{Rel}$$

où : x indique le nombre de serveurs triés.

Γ est le nombre de serveurs parmi les x serveurs classés qui contiennent au moins un document pertinent.

r_i indique le nombre de documents pertinents dans S_i,

Rel est le nombre total des documents pertinents dans tous les serveurs.

Les figures 1 et 2 montrent le moyennes respectives de P_x et R_x sur les 50 requêtes de la collection de test.

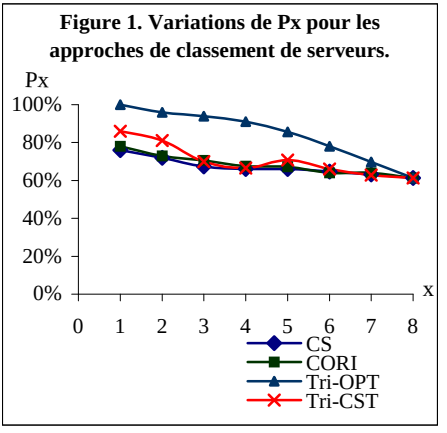

Figure 1. Variations de Px pour les approches de classement de serveurs.

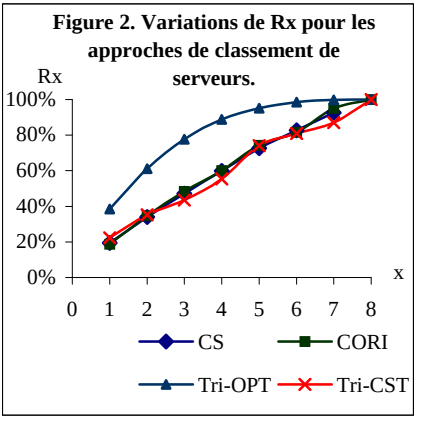

Figure 2. Variations de Rx pour les approches de classement de serveurs.

4.4. Discussion

En analysant les données de la table 2, nous notons que la précision moyenne de SS (la 3eme colonne) ne présente pas une différence significative comparée à l'approche centralisée. Même si les deux autres approches (CS et SNB) présentent une dégradation de performance, toutes les méthodes que nous avons proposées montrent une nette amélioration dans la précision à 15 ou 30 documents (ou à 5, 10, 15, 20 documents retrouvés pour CS et SS). Dans ces cas une amélioration significative de nos approches est aussi détectée par rapport à CORI et le modèle SANS-Sel. Nous constatons neanmoins que CS, SS et SNB entrainent une petite dégradation de performance après 100 documents retrouvés, cependant, peu d'utilisateurs du web consultent au-delà de 20 premiers documents. En examinant les graphes des figures 1 et 2, nous remarquons que les performances de CORI et CS sont très similaires. En observant les valeurs de P_x, la mesure qui représente la proportion de serveurs contenant au moins un document pertinent parmi les x serveurs sélectionnés, nous constatons que Tri-CST est meilleure que CS et CORI si l'on sélectionne seulement 1 ou 2 serveurs. Cependant les valeurs correspondantes de R_x, qui mesure la proportion des documents pertinents retrouvés lorsqu'on sélectionne x serveurs, nous montrent que Tri-CST retrouve pratiquement le même nombre de documents pertinents. En outre, toujours par rapport aux valeurs de R_x nous constatons que les trois approches de classement de serveurs (CS, CORI, et Tri_CST) atteignent la même performance et retrouvent approximativement 70% des documents pertinents lorsque 5 serveurs sont interrogés.

5. Conclusion

Dans un environnement distribué de recherche d'informations, où la coopération n'est pas évidente et où les informations sur les contenus des collections ne sont pas disponibles, nous avons proposé une stratégie d'acquisition de données servant à estimer la pertinence d'un serveur à une requête donnée. Ces données sont extraites par analyse d'un nombre limité des documents qu'un serveur retourne. Une fois la pertinence d'un serveur calculée, nous avons testé trois alternatives afin de résoudre le problème de sélection de serveurs. Premièrement, nous avons trié les serveurs selon leurs pertinences et avons sélectionné les *n_serv* premiers. Deuxièmement, nous avons sélectionné les serveurs dont le score dépasse un certain seuil. Et enfin, nous avons utilisé les informations récoltées pour définir la contribution, en nombre de documents, de chaque serveur dans la liste finale des réponses. Nous avons examiné les performances de ces trois approches en utilisant la collection de test WT10g, et les résultats sont satisfaisants. Bien que nos approches nécessitent plus de trafic de transfert pour télécharger *n_docs* documents par serveur, elles ont néanmoins les avantages suivants : primo, elles ne nécessitent pas de données *a priori*, et de ce fait ne nécessitent aucune mise à jour pour refléter les changements qui peuvent survenir sur les contenus des collections. Secundo, nos approches ne requièrent aucune coopération des serveurs sous-jacents. Enfin nos approches ont le mérite de détecter les serveurs médiocres. En effet, si un serveur contient des

documents pertinents et que son système local n'est pas capable de les retrouver alors il n'est pas intéressant de le sélectionner. Nos approches, en analysant les *n_docs* premiers documents retournés par un serveur, prennent en compte l'efficacité de son système de recherche d'informations local. L'une des perspectives que nous envisageons consiste à utiliser le score attribués aux collections sélectionnées pour pondérer les scores des documents qu'elles retournent afin d'améliorer la fusion des résultats.

6. Bibliographie

[ABBA02] Abbaci F., Savoy J., Beigbeder M., «A Methodology for Collection Selection in Heterogeneous Contexts », conférence IEEE-International Conference on Information Technology : Coding and Computing, 2002.

[CALL95] Callan, J.P., Lu, Z., et Croft, W.B., « Searching distributed collections with inference networks », ACM-SIGIR'95, p. 21-28, 1995.

[CLAR95] Clarke, C.L.A., Cormack, G.V., et Burkowski, F.J., « Shortest substring ranking (MultiText experiments for TREC-4) ». TREC'4, NIST Publication p. 295-304, 1995.

[CRAS00] Craswell, N., Bailey, P., et Hawking, D., « Server selection in the worldwide web », ACM-DL'2000, p. 37-46, 2000.

[GRAV95] Gravano, L., et Garcia-Molina, H., « Generalizing Gloss to Vector-Space Databases and Broker Hierarchies », VLDB'95, 1995.

[GRAV99] Gravano, L., Garcia-Molina, H., et Tomasic, A., « GlOSS: Text-source discovery over the Internet », ACM Transactions on Database Systems, 24(2), p. 229-264, 1999.

[HAWK99] Hawking, D., et Thistlewaite, P., « Methods for information server sélection », ACM Transactions on Information Systems, 17(1), p. 40-76, 1999.

[LAWR98] Lawrence, S., et Lee Giles, C., « Inquirus, the NECI meta search engine », WWW'7, p. 95-105. 1998.

[LU96] Lu Z., Callan J. et Croft W., « Measures in collection ranking evaluation », Rapport technique TR96 -39, Computer Science Department, University of Massachusetts, url: citeseer.nj.nec.com/66442.html, 1996.

[OGIL01] Ogilvie, P., et Callan, J., « The effectiveness of query expansion for distributed information retrieval », ACM-CIKM'2001, p. 183-190, 2001.

[RASO01] Rasolofo, Y., Abbaci, F., et Savoy, J. , « Approaches to collection selection and results merging for distributed information retrieval », ACM-CIKM'2001, p. 191-198, 2001.

[ROBE00] Robertson, S.E., Walker, S., et Beaulieu, M., « Experimentation as a way of life: Okapi at TREC ». Information Processing & Management, 36(1), p. 95-108, 2000.

[TOWE95] Towell, G., Voorhees, E.M., Narendra, K.G., et Johnson-Laird, B., « Learning collection fusion strategies for information retrieval ». The Twelfth Annual Machine Learning Conference, p. 540-548, 1995.

[XU99] Xu, J., et Croft, W.B., « Cluster-based language models for distributed retrieval », ACM-SIGIR'99, p. 254-261, 1999.

[XU98] Xu, J., et Callan, J.P., « Effective retrieval with distributed collections ». ACM-SIGIR'98, p. 112-120, 1998.

[YU01] Yu, C., Meng, W., Wu, W., et Liu, K.-L., « Efficient and effective metasearch for text databases incorporating linkages among documents ». SIGMOD'2001, ACM Press, p. 187-198, 2001.

[ZOBE97] Zobel, J., « Collection selection via lexicon inspection », The Second Australian Document Computing Symposium, p.74-80, 1997.