Université de Neuchâtel

Recherche d'information distribuée : approches pour la sélection de collections et la fusion de listes de résultats.

par

Omega Yves Rasolofo

Institut interfacultaire d'informatique

Thèse présentée à la Faculté des sciences
en vue de l'obtention du grade de Docteur ès sciences
en informatique

Avril 2002

IMPRIMATUR POUR LA THESE

Recherche d'information distribuée: approches pour la sélection de collections et la fusion de listes de résultats

de M. Omega Yves Rasolofo

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. J. Savoy (directeur de thèse), P.-J. Erard,
J.-Y Nie (Montréal, Canada) et
D. Hawking (Canberra, Australie)

autorise l'impression de la présente thèse.

Neuchâtel, le 30 avril 2002

Le doyen:

F. Zwahlen

Remerciements

Je tiens à remercier très sincèrement le Professeur Jacques Savoy de l'Université de Neuchâtel qui a accepté de diriger ce travail et qui m'a fait découvrir le monde de la recherche. J'ai particulièrement apprécié la qualité de son encadrement, ses conseils, son attention, sa disponibilité et son humanité dans le travail.

Je remercie chaleureusement le Professeur David Hawking de CMIS-CSIRO, Canberra, Australie, pour m'avoir accueilli au sein de son laboratoire afin que j'y réalise une partie de mes recherches, ainsi que pour son étroite collaboration. Je lui suis aussi très reconnaissant d'avoir accepté l'expertise de cette thèse.

J'adresse également mes remerciements aux Professeurs Pierre-Jean Erard de Université de Neuchâtel et Jian-Yun Nie de Université de Montréal, qui ont accepté de consacrer du temps à l'expertise de la présente thèse.

Je souhaite exprimer ma gratitude à ma famille, et en particulier à mon épouse, pour leur soutien et leurs encouragements inépuisables.

Enfin, un grand merci aussi aux amis et aux collègues qui m'ont aidé de près ou de loin durant ces trois années de recherche.

Ce travail a été mené avec le soutien du Fonds National Suisse pour la Recherche Scientifique (Subside # 21-58 813.99).

Résumé

Une vue simplifiée d'un système de recherche d'information distribuée fait intervenir des serveurs de collections (ou des moteurs de recherche) et un courtier (métamoteur). Le courtier transmet une requête soumise par un utilisateur à un sous-ensemble de serveurs de collections. Les serveurs sélectionnés sont ceux qui ont une forte probabilité de contenir des documents pertinents à cette requête. Cette première étape est communément appelée « sélection de collections ».

Chaque collection ainsi sélectionnée traite la requête et retourne une liste de documents au courtier. Finalement, le courtier fusionne en une seule liste les différentes listes qu'il a reçues à l'utilisateur. On parle alors de « Fusion de résultats ».

Ce travail propose des nouvelles approches pour la « sélection de collections » et la « fusion de résultats ». Le détail de ces approches et des évaluations que nous avons effectuées est exposé dans les différentes publications incluses dans ce rapport. Ces publications ont été acceptées dans des conférences et des revues de renommée internationale.

Abstract

A simple distributed information retrieval system is made up of collection servers (or search engines) and a broker (or metasearcher). The broker forwards a request submitted by a user to a carefully selected subset of collections servers, those which are likely to contain relevant documents to the query ("collections selection").

Each selected collection server processes the query and returns a ranked list of documents to the broker. Finally, the broker merges the received results lists into a single list ("results merging") and forwards it to the user.

This work proposes new effective approaches to "collections selection" and "results merging". Details on those approaches and the evaluations we made are given in the enclosed papers. Those papers are published in well known conferences and journals.

TABLE DES MATIERES

Partie 1: Introduction

Les moteurs de recherche font désormais partie des outils incontournables pour la recherche d'information sur Internet (recherche de sites, de nouvelles, d'articles scientifiques, d'images, de morceau de musique). Le volume considérable d'information à indexer et les limites de la technologie actuelle poussent ces moteurs à filtrer et/ou à limiter les documents qu'ils vont inclure dans leur index. Une partie importante des informations stockées sur le Web est donc inaccessible au grand public. L'évolution rapide de la technologie et du matériel informatique promettent néanmoins des solutions. Des solutions logicielles commencent à voir le jour, notamment avec les métamoteurs de recherche qui consistent à interroger plusieurs moteurs (dont les contenus des index se chevauchent relativement peu [12]) via une seule interface. On parlera alors de la recherche d'information distribuée (RID). Le niveau de couverture (le nombre de document pouvant être atteint) d'un métamoteur est inversement proportionnel au taux de chevauchement des index des différents moteurs. L'utilisation d'un métamoteur sur des moteurs dont le taux de chevauchement est nul offre donc un niveau de couverture maximal à l'utilisateur.

L'émergence de plusieurs moteurs de recherche spécialisés dont certains se spécialisent dans une langue donnée (fr.yahoo), sur une région ou un pays (search.ch pour la Suisse), ou sur un domaine particulier (par exemple la médecine) peuvent faire l'objet de recherche d'information distribuée. Par exemple, un métamoteur utilisant des moteurs appartenant à des sites d'actualités (CNN, BBCNews, ABC, ...) [56], permet à l'utilisateur d'obtenir une seule liste contenant les différents articles de journaux publiés par différents sites sur un même sujet. Ceci est indéniablement un avantage pratique considérable.

Pour commencer, nous allons présenter quelques notions nécessaires à la lecture de ce travail. Un moteur de recherche est constitué d'un aspirateur (robot ou spider) d'un indexeur et d'un système de classement. L'aspirateur parcours le Web à la recherche des sites et des pages à indexer. L'indexeur résume chaque page Web en lui associant une liste de mots-clés. l'index lui même est créé sous forme de « fichier inversé » [15] dans lequel chaque terme est associé à une liste d'adresses URL des documents indexé par ce terme. Ceci est la forme la plus simple d'un fichier inversé. De plus en plus, les systèmes incluent d'autres informations comme, par exemple, la pondération et les positions du terme dans les documents. Le système de classement d'un moteur de recherche définit la manière avec laquelle le score d'un document donné est calculé. Les scores sont utilisés pour classer les documents à présenter à l'utilisateur. Un des systèmes les plus connus est le modèle probabilistique Okapi [13]. Ce système calcule le score d'un document en prenant en compte le nombre d'occurrence du terme de la requête dans le document, le nombre de documents contenant au moins une fois le terme et le nombre de termes du document analysé. D'autres méthodes de calcul de score sont proposées dans la littérature : par exemple, une autre approche probabilistique proposée par Amati *et al.* [1], , ou les travaux de Buckley *et al.* [2] qui proposent d'autres variantes plus efficaces du modèle vectoriel [15] et enfin, d'autres auteurs suggèrent d'inclure une mesure de proximité des termes [10].

Nos travaux ont pour cadre la RID (recherche d'information distribuée. Une vue simplifiée de la RID consiste à disposer plusieurs moteurs travaillant chacun sur sa propre fichier inversé sur un réseau (dans les articles présentés dans cette thèse, un moteur pourrait être tout simplement désigné par « une collection » sur laquelle le moteur travaille). Dans un

tel système distribué, l'utilisateur envoie sa requête vers un « courtier »[1] dont le rôle consiste à :

- éventuellement sélectionner automatiquement ou selon le choix de l'utilisateur (sélection manuelle), les moteurs susceptibles de contenir des documents pertinents à la requête (ce problème se nomme « sélection de serveurs » ou « sélection de collections »),
- traduire la requête vers les langages (ou format de requête) reconnus par les différents moteurs,
- soumettre les requêtes traduites aux moteurs correspondants,
- recevoir les différentes listes de résultats retournées par les moteurs et construire une seule liste qui sera affichée à l'utilisateur en suivant une certaine stratégie (problème nommé « fusion de résultats » ou « fusion de collections »).

Nos travaux se concentrent surtout sur deux problèmes, à savoir « la sélection de collections » et « la fusion de résultats ». Ces deux problèmes sont fondamentaux à la RID car on a constaté que les approches proposées pour résoudre ces deux problèmes affectent considérablement la qualité des résultats retournés. En effet, la plupart des tests effectués en laboratoire rapportent une diminution plus ou moins forte de la qualité des résultats [19], [45], [55], [60], [65] si l'on compare la RID avec une approche centralisée. Les articles qui sont inclus dans ce travail présentent les différentes expérimentations que nous avons faites afin d'évaluer les approches proposées dans la littérature, et proposent des nouvelles approches susceptibles d'améliorer la qualité des résultats.

[1] Ce terme est rarement utilisé dans nos articles, mais est défini ici pour une meilleure compréhension des concepts relatifs à la RID.

1.1. Présentations des publications

1.1.1. Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections.

Cet article décrit notre première évaluation de différentes approches en RID durant la campagne d'évaluation TREC[2]. L'utilisation de la RID a été motivée par le volume de la collection test fourni par TREC et notre volonté d'évaluer une telle approche dans le cadre d'une compétition. Nous nous sommes contentés d'utiliser les approches de fusion de résultats connues, notamment: « round robin »[3], « raw score »[4], « normalised raw score »[5] et « CORI » [22]. Pour chaque collection, nous avons utilisé le système Okapi qui propose une qualité de réponse intéressante. Nous avons soumis des résultats avec et sans reformulation de requête (« relevance feedback ») automatique utilisant la formule de Rocchio [14]. Nos résultats utilisant la méthode « raw score » (sans « relevance feedback ») avec les requêtes courtes (plus proche de la réalité sur le Web) s'avère être parmi les meilleurs.

1.1.2. Recherche d'informations dans un environnement distribué.

Ce travail fait suite à nos premières explorations et propose une méthode d'apprentissage automatique pour la RID [47]. Nous utilisons dans nos expérimentations des arbres de décision en vue de sélectionner quelles sources d'information (collections) devraient être interrogées ou non (sélection de collection). Les arbres de décision se présentent comme un outil adéquat car :

- contrairement à d'autres modèles d'apprentissage (par exemple : réseau de neurones), les arbres de décision sont faciles à interpréter. Cette facilité de lecture des règles permet de voir l'importance de chaque attribut pour la prise de décision ;
- les règles, sous forme d'arbres de décision peuvent être générées automatiquement. Cette caractéristique est important car dans un environnement réel, les règles utilisées pour traiter une requête envoyée par un utilisateur devraient se baser sur des données approchant le plus près possible des propriétés statistiques actuelles de la collection interrogée. En effet, les documents inclus dans une collection évoluent, leurs contenus changent, de nouveaux documents sont ajoutés et d'autres supprimés;
- il est possible de considérer plusieurs attributs : cette possibilité nous a permis de considérer simultanément plusieurs attributs sans tenir compte ni de leur type (continu ou discret) ni des transformations (par exemple : normalisation ou linéarisation) nécessaires à la plupart des méthodes proposées pour la RID.

Nous avons opté pour l'outil expérimental C4.5 qui se base sur la théorie d'information pour construire les arbres de décision. Nous avons obtenu des résultats encourageants dans ces expérimentations. L'avantage de cette approche est l'indépendance

---

[2] TREC (Text REtrieval Conference) : http://trec.nist.gov/

[3] Round robin (à chacun son tour): pour construire la liste de document finale, prendre à tour de rôle un document de chaque liste retourné par chaque collection.

[4] Raw score (par les scores) : mettre les documents venant de différentes collections dans une seule liste et les classer selon leurs scores.

[5] Normalised raw score (par les scores normalisés): idem que « raw score » mais pour chaque liste, les scores des documents sont d'abord divisés par le score maximal de la liste.

des décisions prises pour chaque collection par rapport aux autres collections. Chaque collection possède son propre arbre de décision construit à partir des attributs locaux.

1.1.3. Recherche d'informations dans des sources distribuées.

Nous avons introduit dans cet article une nouvelle méthode de fusion des résultats. Cette approche se base sur l'utilisation de la longueur des résultats (soit le nombre de documents inclus dans chaque liste retournée). Notre approche se démarque des autres en raison de sa simplicité et de la facilité qu'elle offre pour l'implémentation. De plus, cette démarche originale permet une amélioration de la qualité des réponses comparable à celle obtenue avec CORI [22], l'une des meilleures approches proposées dans la littérature. La méthode CORI se base sur des statistiques des collections (nombre de documents contenant chaque terme dans les différentes collections), lesquelles sont stockées au niveau du courtier. Le fait de devoir stocker ces informations au niveau du courtier présente un désavantage non négligeable du modèle CORI. En outre, l'approche CORI impose une mise à jour régulière afin d'actualiser ces méta-données pour qu'elles reflètent les contenus actuels des différentes collections. L'espace disque nécessaire pour stocker ces informations représente un deuxième problème. Notre approche en revanche, utilise des informations recueillies au moment de la réception des résultats venant des collections, ce qui supprime tout problème de mise à jour et d'espace de stockage. Nos expérimentations ont montré que cette approche est particulièrement efficace lorsque les collections utilisent le système Okapi.

1.1.4. Approaches to Collection Selection and Results Merging for Distributed Information Retrieval.

Une nouvelle méthode de sélection de collections est présentée en combinaison avec notre approche de fusion des résultats présentée dans le précédent article. Les évaluations comparent nos approches de sélection de collection et de fusion de résultats avec d'autres approches connues. Une analyse plus approfondie est faite pour la vérification statistique des résultats obtenus. Cette vérification a été faite en utilisant le test du signe [15]. Sur cette base, nous arrivons à prouver que notre stratégie de fusion de résultats apporte des résultats aussi bon et avec moins d'inconvénient que les meilleures solutions proposées par d'autres auteurs.

1.1.5. Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching.

Dans le cadre de la campagne d'évaluation TREC-10, nos résultats utilisaient notre stratégie de fusion précédente. Notre performance a été quelque peu décevante dans la compétition « ad hoc ». Dans cette compétition, nous avons décidé de ne pas utiliser la reformulation automatique des requêtes en se basant sur les résultats négatifs obtenus par les différents groupes lors de TREC-9. Or pour TREC-10, les meilleurs groupes ont tous utilisé la reformulation de requête. Nos résultats ont pourtant montré que notre méthode de fusion permet d'augmenter la qualité de la recherche en terme de précision moyenne [15] dans le contexte de RID. Dans cet article, nous proposons également une nouvelle adaptation du système Okapi tenant en compte de la proximité des termes d'une requête dans les documents. Cette méthode s'avère être efficace pour améliorer le classement des premiers documents pertinents. Cette nouvelle pondération a contribué à notre bonne prestation dans la compétition sur la recherche des « homepages ».

1.1.6. Fusion de collections dans les métamoteurs.

Dans un environnement réel, les différents moteurs utilisent différents systèmes d'indexation et de recherche. Dans de telles conditions, les scores attribués aux documents

retournés par différents moteurs sont incomparables. Dans cet article nous proposons, une méthode de fusion légèrement différente de celle qui a été proposée dans nos précédentes publications. Une première évaluation consiste à oeuvrer dans une configuration idéale dans laquelle toutes les collections utilisent le même système. Les évaluations ont montré que l'approche proposée est meilleure que la méthode « round robin ». La seconde partie des expérimentations se voulant plus réaliste (les différentes collections n'utilisent pas le même système de recherche), ont encore montré une amélioration positive.

1.1.7. Result Merging Strategies for a Current News MetaSearcher.

Pour prendre en compte la réalité du Web, nous avons étendu notre étude sur une application réelle: la recherche de nouvelles à partir de quinze serveurs tels que CNN, BBCNews, ABC, etc, sites qui fournissent des moteurs de recherche en ligne. Cette étude a debouche sur la creation d'un métamoteur interrogeant ces quinze sites (abcnews.go.com, www.bbc.co.uk, news.cnet.com, cnnfn.cnn.com, www.ft.com, www.msnbc.com, www.news.com.au, www.thestar.ca, www.thetimes.co.uk, www.tokyo-weekly.ne.jp, allafrica.com, www.sptimes.ru, www.usatoday.com, www.dispatch.co.za, www.worldnews.com). Dans ce contexte, la plupart des informations utilisées par les approches suggérées en RID ne sont pas disponibles. Nous avons constaté que non seulement les données importantes comme le nombre de document contenant chaque terme de la requête (*df* ou document frequency) n'est jamais disponible mais que les informations fournies diffèrent d'un site à l'autre. Nous avons donc proposé une méthode simple de calcul de scores en se basant sur les informations les plus fréquemment disponibles (notamment le titre, la description et le rang des articles dans la liste des résultats retournée par les moteurs de recherches). Nous avons proposé une formule utilisant le titre et/ou la description d'un article. Notre approche tient compte du nombre de termes de la requête inclus dans le titre (et/ou la description) d'un article, la longueur du titre et celle de la requête. Les évaluations se basant sur la combinaison de plusieurs évidences permet d'atteindre des performances comparables à une approche qui consiste à collecter les documents, les indexer puis les classer. Cette dernière alternative exige un temps important de traitement et demande beaucoup de ressource, et de ce fait ne présente donc qu'un intérêt théorique.

1.2. Conclusion

Les travaux que nous avons mené ont démontré que la RID par l'intermédiaire des métamoteurs, offre une solution intéressante pour résoudre le problème lié à l'incapacité de stocker de gros volume de donnée par les moteurs de recherche disponible sur le Web. De plus, les métamoteurs reposent sur le même principe que les moteurs de recherche mais disposent d'un ensemble d'informations plus restreintes pour opérer les deux importantes tâches étudiées dans nos travaux: la sélection de collection et la fusion des résultats. Les approches que nous avons proposées tant pour la sélection de collection que pour fusion de résultats ont montré qu'il est possible d'atteindre une qualité de recherche comparable à celle obtenue par les moteurs de recherche basé sur un seul fichier inversé (approche centralisée).

1.3. Bibliographie complète

1.3.1. Recherche d'information

[1] Amati G., Carpineto C., Romano G.: FUB at TREC-10 Web Track: A Probabilistic Framework for Topic Relevance Term Weighting. Proceedings of TREC-10, 2002, (to appear).

[2] Buckley C., Singhal A., Mitra M., Salton G.: New Retrieval Approaches using SMART. Proceedings of TREC-4, #500-236, 1996, pp. 25-48.

[3] Buckley C.: Implementation of SMART Information Retrieval System. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, May 1985.

[4] Chakravarthy A. S., Haase K. B.: NetSerf: Using Semantic Knowledge to Find Internet Information Archives. Proceeding of the ACM-SIGIR'95, 1995, pp. 4-11.

[5] Clarke C. L. A., Cormack G. V., Burkowski F. J.: Shortest Substring Ranking (MultiText Experiments for TREC-4). Proceedings of TREC-4, 1995, pp. 295-304.

[6] Craswell N., Hawking D., Robertson S.: Effective Site Finding Using Link Anchor Information. Proceedings of the ACM-SIGIR'01, 2001. pp. 250-257.

[7] Croft W.B.: Combining Approaches to Information Retrieval. In W.B. Croft (Ed.), Advances in Information Retrieval, Kluwer Academic Publishers, pp. 1-36, 2000.

[8] Dumais S. T.: Latent Semantic Indexing (LSI) and TREC-2. Proceedings of TREC-2, 1994, pp. 105-115.

[9] Hawking D., Craswell N., Bailey P., Griffiths K.: Measuring Search Engine Quality. Information Retrieval 4(1), 2001, pp. 33-59.

[10] Hawking D., Thistlewaite P.: Proximity Operators - So Near And Yet So Far. Proceedings of TREC-4, #500-236, pp. 131-144, 1996.

[11] Kwok K. L., Grunfeld L., Lewis D. D.: TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. Proceedings of TREC-3, 1995, pp. 247-255.

[12] Lawrence S., Giles C.L.: Searching the World Wide Web. Science 280(5360): 98-100, 1998.

[13] Robertson S. E., Walker, S., Beaulieu M.: Experimentation as a Way of Life: Okapi at TREC. Information Processing & Management, 36(1), 2000, pp. 95-108.

[14] Rocchio J.J.: Relevance Feedback in Information Retrieval. In The SMART Retrieval System, G. Salton (Ed), Prentice Hall, Inc, 1971, pp. 313-323.

[15] Salton G., McGill M.J.: Introduction to Modern Information Retrieval. New-York: McGraw-Hill, 1983.

1.3.2. Recherche d'information distribuée

[16] Aslam J.A., Montague M.H.: Models for Metasearch. Proceedings of the ACM-SIGIR'01, 2001, pp. 275-284.

[17] Aslam J.A., Montague M.H.: Bayes optimal metasearch: a probabilistic model for combining the results. Proceedings of the ACM-SIGIR'00, 2000, pp. 379-381.

[18] Baldonado M., Chang C.-C. K., Gravano L., Paepcke A.: The Stanford Digital Library Metadata Architecture, , in International Journal of Digital Libraries, 1(2), 1997, pp. 108-121.

[19] Baumgarten C.: A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. Proceeding of the ACM-SIGIR'99, 1999, pp. 246-253.

[20] Baumgarten C.: Probabilistic Information Retrieval in a Distributed Heterogeneous Environment. Dresden, Technische Universität, Fakultät Informatik, Diss., 1999.

[21] Baumgarten C.: A Probabilistic Model for Distributed Information Retrieval. Proceeding of the ACM-SIGIR'97, 1997, pp. 258-266.

[22] Callan J.: Distributed Information Retrieval. In W. B. Croft (Ed.), Advances in Information Retrieval. Kluwer Academic Publishers, 2000 pp. 127-150.

[23] Callan J.P., Lu Z. and Croft W.B.: Searching distributed collections with inference networks. Proceedings of the ACM-SIGIR'95, 1995, pp. 21-28.

[24] Craswell N., Bailey P., Hawking D.: Server Selection in the World Wide Web. Proceedings of The Fifth ACM Conference on Digital Libraries, 2000, pp. 37-46.

[25] Craswell N., Hawking D., Thistlewaite P.B.: Merging Results From Isolated Search Engines. Australasian Database Conference, 1999, pp. 189-200.

[26] Danzig P.B., Ahn J., Noll J., Obraczka K.: Distributed Indexing: A Scalable Mechanism for Distributed Information Retrieval. Proceedings of the ACM-SIGIR'91, 1991, pp. 220-229.

[27] De Kretser O., Moffat A., Shimmin T., Zobel J.: Methodologies for Distributed Information Retrieval. Proceedings of the 18th International Conference on Distributed Computing Systems, 1998, pp. 66-73.

[28] Dolin R., Agrawal D., El Abbadi A.: Scalable Collection Summarization and Selection. Proceedings of the ACM-DL, 1999, pp. 49-58.

[29] Dolin, R.: Pharos: A Scalable Distributed Architecture for Locating Heterogeneous Information Sources. Ph.D. Dissertation, 1998.

[30] Dreilinger D., Howe A.E.: Experiences with Selecting Search Engines Using Metasearch. ACM Transactions on Information Systems, 15(3), 1997, pp. 195-222.

[31] Dushay N., French J.C., Lagoze C.: Using Query Mediators for Distributed Searching in Federated Digital Libraries. Proceedings of the ACM-DL, 1999, pp. 171-178.

[32] Fan Y., Gauch S.: Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources. Proceedings of the AAAI Symposium on Intelligent Agents in Cyberspace, 1999, pp. 40-46.

[33] French J.C., Powell A.L., Gey F.C., Perelman N.: Exploiting A Controlled Vocabulary to Improve Collection Selection and Retrieval Effectiveness. Proceedings of the ACM-CIKM 2001, pp. 199-206.

[34] French J.C., Powell A.L.: Metrics for evaluating database selection techniques. Technical Report CS-99-19, 1999.

[35] French J.C., Powell A.L., Callan J., Viles C.L., Emmitt T., Prey K.J. and Mou Y.: Comparing the performance of database selection algorithms. Proceedings of the ACM-SIGIR'99, 1999, pp. 238-245.

[36] French J.C., Powell A.L., Creighton III W.R.: Efficient Searching in Distributed Digital Libraries. Proceedings of the ACM-DL, 1998, pp. 283-284.

[37] Gauch S., Wang G., Gomez M.: ProFusion*: Intelligent Fusion from Multiple, Distributed Search Engines. Journal of Universal Computer Science 2(9), 1996, pp. 637-649.

[38] Gauch S., Wang G.: Information Fusion With ProFusion. Proceedings of WebNet '96, 1996, pp. 174-179.

[39] Glover E.J., Lawrence S., Birmingham W.P., Giles C.L.: Architecture of a Metasearch Engine That Supports User Information Needs. Proceedings of the ACM-CIKM'99, 1999, pp. 210-216.

[40] Gravano L., Garcia-Molina H., Tomasic A.: GlOSS: Text-Source Discovery Over the Internet. ACM Transactions on Database Systems, 24(2), 1999, pp. 229-264.

[41] Green N., Ipeirotis P., Gravano L.: SDLIP + STARTS = SDARTS: A Protocol and Toolkit for Metasearching. Proceedings of the First ACM+IEEE Joint Conference on Digital Libraries (JCDL 2001), 2001, pp. 207-214.

[42] Hawking D., Thistlewaite P.: Methods for Information Server Selection. ACM Transactions on Information Systems, 17(1), 1999, pp. 40-76.

[43] Hill L.L., Janee G., Dolin R., Frew J., Larsgaard M.: Collection Metadata Solutions for Digital Library Applications. JASIS 50(13), 1999, pp. 1169-1181.

[44] Howe A.E., Dreilinger D.: SAVVYSEARCH: A Metasearch Engine That Learns Which Search Engines to Query. AI Magazine 18(2), 1997, 19-25.

[45] Larkey L. S., Connell M. E., Callan J.: Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. Proceedings of CIKM'2000, 2000, pp. 282-289.

[46] Lawrence S., Giles C. L.: Inquirus, the NECI Meta Search Engine. Proceedings of The Seventh International World Wide Web Conference, 1998, pp. 95-105.

[47] Le Calvé A., Savoy J. : Database Merging Strategy Based on Logistic Regression. Information Processing & Management, 36(3), 2000, pp. 341-359.

[48] Ipeirotis P., Barry T., Gravano L.: Extending SDARTS: Extracting Metadata from Web Databases and Interfacing with the Open Archives Initiative, to appear in Procedings of the Second ACM+IEEE Joint Conference on Digital Libraries (JCDL 2002), 2002.

[49] Manmatha R., Rath T., Feng F.: Modeling Score Distributions for Combining the Outputs of Search Engines. Proceedings of the ACM-SIGIR'01, 2001, pp. 267-275.

[50] Meng W., Yu C., Liu K.: Building Efficient and Effective Metasearch Engines . ACM Computing Surveys , 34(1), 2002, pp.48-84.

[51] Moffat A. , Zobel J.: Information Retrieval Systems for Large Document Collections. Proceedings of TREC-3, 1995, pp. 85-94.

[52] Montague M.H., Aslam J.A.: Relevance Score Normalization for Metasearch. Proceedings of the ACM-CIKM'01, 2001, pp. 427-433.

[53] Ogilvie P. and Callan. J.: The effectiveness of query expansion for distributed information retrieval. Proceedings of the 10th ACM-CIKM, 2001, pp. 183-190.

[54] Pinkerton B.: Finding What People Want: Experiences with the WebCrawler. Proceedings of WWW'94, 1994, http://www.thinkpink.com/bp/WebCrawler/WWW94.html.

[55] Powell A.L., French J.C., Callan J., Connell M. and Viles C.L.: The impact of database selection on distributed searching. Proceedings of ACM-SIGIR'00, 2000, pp. 232-239.

[56] Rasolofo Y., Hawking D., Savoy J.: Result Merging Strategies for a Current News MetaSearcher, Information Processing & Management. January 2002. (to appear).

[57] Rasolofo Y., Abbaci F., Savoy J.: Approaches to Collection Selection and Results Merging for Distributed Information Retrieval. ACM-CIKM'2001, 2001, pp. 191-198.

[58] Savoy J., Rasolofo Y., Abbaci F.: Fusion de collections dans les métamoteurs. Actes JADT 2002, mars 2002, (à paraître) .

[59] Savoy J., Rasolofo Y., Abbaci F.: Recherche d'informations dans des sources distribuées. INFORSID 2001, 2001, pp. 237-252.

[60] Savoy J., Rasolofo Y.: Report on TREC-9 Experiment: Linked-based Retrieval and Distributed Collections. Proceedings of TREC9, 2000.

[61] Savoy J., Rasolofo Y.: Recherche d'informations dans un environnement distribué. Actes TALN 2000, Octobre 2000, pp. 317-326.

[62] Selberg E.W.: Towards Comprehensive Web Search. Ph.D. Thesis, University of Washington, 1999.

[63] Shaw J.A., Fox E.A.: Combination of Multiple Searches. Proceedings of TREC-3, 1994, pp. 105-108.

[64] Towell G., Voorhees E. M., Narendra K. G., Johnson-Laird B.: Learning Collection Fusion Strategies for Information Retrieval. Proceedings of The Twelfth Annual Machine Learning Conference, 1995, pp. 540-548.

[65] Voorhees E. M., Gupta N. K., Johnson-Laird B.: Learning Collection Fusion Strategies. Proceedings of the ACM-SIGIR'95, 1995, pp. 172-179.

[66] Viles C.L., French J.C.: Content Locality in Distributed Digital Libraries. Information Processing and Management, 35(3), 1999, pp. 317-336.

[67] Viles C.L. : Maintaining retrieval effectiveness in distributed, dynamic information retrieval systems. Ph.D. thesis, University of Virginia, 1996.

[68] Viles, French J.C.: Dissemination of Collection Wide Information in a Distributed Information Retrieval System. Proceedings of the ACM-SIGIR'95, 1995, pp. 12-20.

[69] Wu Z., Meng W., Yu C., Li Z.: Towards a Highly-Scalable and Effective Metasearch Engine . Proceedings of Tenth World Wide Web Conference (WWW10), 2001, pp.386-395.

[70] Xu J., Callan J. P.: Effective Retrieval with Distributed Collections. Proceedings of the ACM-SIGIR'98, 1998, pp. 112-120.

[71] Xu J., Croft W. B.: Cluster-based Language Models for Distributed Retrieval. Proceedings of ACM-SIGIR'99, 1999, pp. 254-261.

[72] Yu C., Meng W., Wu W., Liu K.: Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents. Proceedings of ACM-SIGMOD Conference, 2001, pp.187-198.

[73] Yuwono B., Lee D.L.: Server Ranking for Distributed Text Retrieval Systems on the Internet. Proceedings of DASFAA'97, 1997, pp. 41-50.

[74] Yuwono B., Lee D.L.: Search and Ranking Algorithms for Locating Resources on the World Wide Web. Proceedings of ICDE'96, 1996, pp. 164-171.

[75] Zhu X., Gauch S.: Incorporating quality metrics in centralized/distributed information retrieval on the World Wide Web. Proceedings of the ACM-SIGIR, 2000, pp. 288-295.

[76] Zobel J.: Collection Selection via Lexicon Inspection. Proceedings of The Second Australian Document Computing Symposium, 1997.

Partie 2: Nouvelles approches pour la sélection de collections et la fusion de résultats

Cette partie est consacréé à la présentation détaillée des différentes méthodes par le moyen des articles que nous avons publiés au cours de nos recherches. Les articles concernés sont les suivants :

- Savoy J., Rasolofo Y.: Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections. Proceedings TREC-9, NIST publication #500-249, Gaithersburg (MD), 2001, 579-588
- Savoy J., Rasolofo Y.: Recherche d'informations dans un environnement distribué. Actes TALN 2000, Octobre 2000, 317-326.
- Savoy J., Rasolofo Y., Abbaci F.: Recherche d'informations dans des sources distribuées. *INFORSID 2001*, Genève (Suisse), Mai 2001, 237-252.
- Rasolofo Y., Abbaci F., Savoy J.: Approaches to Collection Selection and Results Merging for Distributed Information Retrieval. ACM-CIKM'2001, Atlanta, November 2001, 191-198.
- Savoy J., Rasolofo Y.: Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching. Proceedings TREC-10, NIST, Gaithersburg (MD), 2002, (to appear).
- Savoy J., Rasolofo Y., Abbaci F.: Fusion de collections dans les métamoteurs. Actes JADT 2002, mars 2002, (à paraître) .
- Rasolofo Y., Hawking D., Savoy J.: Result Merging Strategies for a Current News MetaSearcher. Information Processing & Management (to appear), 2002.

Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections

Jacques Savoy, Yves Rasolofo

Proceedings TREC-9, NIST, Washigton D.C., November 2000.

Institut interfacultaire d'informatique
Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch
Web page: www.unine.ch/info/

Summary

The web and its search engines have resulted in a new paradigm, generating new challenges for the IR community which are in turn attracting a growing interest from around the world. The decision by NIST to build a new and larger test collection based on web pages represents a very attractive initiative. This motivated us at TREC-9 to support and participate in the creation of this new corpus, to address the underlying problems of managing large text collections and to evaluate the retrieval effectiveness of hyperlinks.

In this paper, we will describe the results of our investigations, which demonstrate that simple raw score merging may show interesting retrieval performances while the hyperlinks used in different search strategies were not able to improve retrieval effectiveness.

Introduction

Due to the huge number of pages and links, browsing cannot be viewed as an adequate searching process, even with the introduction of tables of contents or other classifying lists (e.g., Yahoo!). As a result, effective query-based mechanisms for accessing information will always be needed. Search engines currently available on the web are not able to adequately access all available information [Lawrence 99], as they are inhibited by many drawbacks [Hawking 99].

In the first chapter, we will describe our experiments on the web track in which a large web text collection is divided into four sub-collections in order to keep inverted file size below the 2 GB limit. The second chapter will verify whether or not hyperlinks improved retrieval effectiveness based on four different link-based search models.

To evaluate our hypothesis, we used the SMART system as a test bed for implementing the OKAPI probabilistic model [Robertson 95]. This year our experiments were conducted on an Intel Pentium III/600 (memory: 1 GB, swap: 2 GB, disk: 6 x 35 GB) and all experiments were fully automated.

1. Distributed collections

To evaluate the retrieval effectiveness of various merging strategies, we formed four separate sub-collections (see Appendix 1). In this study, we assumed *that each sub-collection used the same* indexing schemes and retrieval procedures. A distributed cootext such as this more closely reflects local area networks or search engines available on the Internet than the meta search engines, where different search engines may collaborate to respond to a given user request [Le Calvé 00], [Selberg 99].

The following characteristics would more precisely identify our approach. A query was sent to all four text databases (no selection procedure were applied) and according to the four ranked lists of items produced, our search system merged them into a single result list presented to the user (Section 1.2). Before we describe the collection merging approaches, Section 1.1 will identify retrieval effectiveness measures achieved by various search models with the whole collection and with each of our four sub-collections.

1.1. Performance of sub-collections

From the original web pages, we retained only the following logical sections: <TITLE>, <H1>, <CENTER>, <BIG>, with the most common tags <P> (or , together with </P>,) being removed. Text delimited by the tags <DOCHDR>, </DOCHDR> were also removed. For long requests, various insignificant keywords were also removed (such as "Pertinent documents should include ..."). Moreover, search keywords appearing in the Title part of the topics were considered to have a term frequency of 3 (this feature has no impact on short requests).

For the web track, we conducted different experiments usiog the OKAPI probabilistic model in which the weight w_{ij} assigned to a given term t_j in a document D_i was computed according to the following formula:

$$w_{ij} = \frac{(k_1 + 1)\ tf_{ij}}{K + tf_{ij}}$$

$$\text{with } K = k_1 \cdot \left((1-b) + b \cdot \frac{l_i}{avdl} \right)$$

where tf_{ij} indicates the within-document term frequency, and b, k_1 are parameters. K represents the ratio between the length of D_i measured by l_i (sum of tf_{ij}) and the collection mean denoted by advl.

To index a request, the following formula was used:

$$w_{qj} = \frac{tf_{qj}}{k_3 + tf_{qj}} \cdot \ln[(N - df_j) / df_j]$$

where tf_{qj} indicates the search term frequency, df_j the collection-wide term frequency, N the number of documents in the collection, and k_3 is a parameter.

To adjust the underlying parameters of the OKAPI search model, we used advl = 900, b = 0.7625, k_1 = 1.5, and k_3 = 1000. These parameter values were set according to the best performance achieved on the WT2g (TREC-8). A slightly different parameter setting was suggested by Walker *et al.* [98] whereby advl = 900, b = 0.75, k_1 = 1.2, and k_3 = 1000. When using our parameter values, the corresponding label will be "OKAPI" while the second setting will be identified by adding an "R".

Two different query formulations were considered: (1) using only the Title section (T), or (2) all three logical sections (Title, Descriptive and Narrative, noted T-D-N). The data in Table 1 shows that retrieval effectiveness is significantly enhanced when topics include more search terms.

In order to build a single collection, we selected the first 500 retrieved items of 13 search strategies (corresponding to OKAPI and different vector-space approaches) and we added all relevant documents not retrieved by our various search models.

Table 1 provides a summary of the results of our various experiments. In this case, we reported the non-interpolated average precision at eleven recall values, based on 1,000 retrieved items per request. From this data we can see that the parameter setting used by Walker's *et al.* results in better performance (e.g., in the WEB9.1 sub-collection, the average precision increases from 19.47 to 20.30 (+4.3%)).

It is recognized that pseudo-relevance feedback (blind expansion) is a useful technique for enhancing retrieval effectiveness. In this study, we evaluated the OKAPI search model with and without query expansion in order to verify whether or not this technique might improve retrieval performance when faced with different query formulations.

In this study, we adopted Rocchio's approach [Buckley 96] where the parameter settings were chosen according to experiments done with the WT2g from the TREC data (TREC-8).

For a short request the values α=0.75, β=0.25 were assigned and the system was allowed to add to the original query those 50 search terms extracted from the 12-best ranked documents. For long queries, the parameters were set as follows: α=0.7, β=0.3 and the search engine was allowed to add to the original query those 40 search terms extracted from the 15 best-ranked documents. The resulting retrieval effectiveness is depicted in Table 1 under the label "XQ".

After examining sub-collections WEB9.1 and WEB9.3, there was some improvement in results, as depicted in Table 1. For example, based on our parameter setting and examining the WEB9.1 sub-collection, the average precision increased from 19.47 (label "OKAPI") to 21.44 (label "OKAPIXQ") (+10.1%). However, for the other two sub-collections, the average precision decreased (e.g. in WEB9.4, the average precision decreases from 19.26 to 18.24 (-5.3%)).

1.2. Merging procedure

Recent works have suggested solutions in which answer lists were merged in order to produce a unique ranked list of retrieved records. As a first approach, we might assume that each sub-collection contains approximately the same number of pertinent items and that the distribution of the relevant documents is the same across the answer lists. Based only on a ranking of the retrieved records, we might interleave the results in a round-robin fashion. According to previous studies [Voorhees 95], [Callan 95], the retrieval effectiveness of such interleaving schemes is around 40% below the performance achieved by a single retrieval scheme technique, with a single huge collection representing the entire set of documents. The third column of Table 2 confirms this finding but to a lesser extent (around -26.1% when dealing with short queries or -17.0% when examining Title, Descriptive and Narrative fields in the topics).

	Average Precision				
Query Title only Model	WEB9.1 46 queries 749 rel	WEB9.2 44 queries 600 rel	WEB9.3 43 queries 608 rel	WEB9.4 46 queries 660 rel	WEB9 50 queries 2,617 rel.
Okapi	19.47	20.85	16.09	19.26	19.55
OkapiR	20.30	21.32	16.52	19.51	19.86
OkapiXQ	21.44	20.89	17.73	18.24	19.43
OkapiNRXQ	21.70	20.67	18.98	18.33	19.31
Query T-D-N					
Okapi	32.61	30.26	28.09	28.44	27.25
OkapiNR	33.25	30.19	29.01	28.49	27.52
OkapiXQ					28.10
OkapiNRXQ	34.41	28.25	31.18	26.69	28.30

Table 1: Average precision of isolated sub-collections and the whole test collection

In order to account for the score achieved by the retrieved document, we might formulate the hypothesis that each sub-collection is managed by the same search strategy and that the similarity values are therefore directly comparable [Kwok 95]. Such a strategy, called raw-score merging, produces a final list, sorted by the retrieval status value computed by each separate search engine.

However, as demonstrated by Dumais [94], collection-dependent statistics in document or query weights may vary widely among sub-collections; and therefore, this phenomenon may invalidate the raw-score merging hypothesis.

The fourth column of Table 2 indicates the retrieval effectiveness of such merging approaches, depicting a relatively interesting performances in our case (degradation of around -5.3% for long requests or -14.9% for short queries). Thus, the raw-score merging seems to be a simple and valid approach when a huge collection is distributed across a local-area network and operating within the same retrieval scheme.

As a third merging strategy, we may normalize each sub-collection's similarity value ($SIM(D_i, Q)$) by dividing it by the maximum value in each result list. The fifth column in Table 2 shows its average precision, depicting surprisingly poor retrieval effectiveness (average reduction of -19.6% for short queries and -16.2% for long requests).

As a fourth merging strategy, Callan *et al.* [95] suggest using the CORI approach, which will first compute a score s_i for each sub-collection as follows:

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

where t_j indicates a search keyword, db_i the ith collection, df_i the number of documents in the ith collection containing term t_j, cf_j the number of collections containing term t_j, db the total (number of collections equals to four in our case), ldb_i the number of indexing terms included in the ith corpus, avldb the mean value of ldb_i, where defB, b and k are three parameters. Xu & Callan [98] suggest assigning values to these constants (defB=0.4, k=200, and b=0.75, values used in this study). The previous equation is defined for one search term, and the score for a given collection is simply the sum over all keywords included in the current request.

The sub-collection score (noted s_i) is the first component used to merge the retrieved items. To obtain the score of a given retrieved item of the ith collection, the similarity between the request and the document is multiplied by a coefficient w_i computed as follows:

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

	Average Precision (% change)				
Query Title Model	50 queries 2617 rel one coll	merge 50 queries 2617 rel round-robin	merge 50 queries 2617 rel raw-score	merge 50 queries 2617 rel norm. score	merge 50 queries 2617 rel CORI
Okapi	19.55	13.88 (-29.0%)	17.59 (-10.0%)	15.94 (-18.5%)	15.83 (-19.0%)
OkapiR	19.86	14.44 (-27.3%)	17.81 (-10.3%)	16.37 (-17.6%)	15.99 (-19.5%)
OkapiXQ	19.43	14.54 (-25.2%)	15.96 (-17.9%)	15.07 (-22.4%)	15.31 (-21.2%)
OkapiNRXQ	19.31	14.89 (-22.9%)	15.87 (-17.8%)	15.44 (-20.0%)	15.28 (-20.9%)
		-26.1%	-14.9%	-19.6%	-20.2%
Query T-D-N					
Okapi	27.25	22.82 (-16.3%)	26.56 (-2.5%)	23.39 (-14.2%)	26.81 (-1.6%)
OkapiNR	27.52	23.19 (-15.7%)	26.75 (-2.8%)	23.94 (-13.0%)	26.87 (-2.4%)
OkapiXQ	28.10	23.09 (-17.8%)	25.99 (-7.5%)	23.28 (-17.2%)	25.94 (-7.7%)
OkaNRXQ	28.30	23.22 (-18.0%)	25.93 (-8.4%)	22.57 (-20.2%)	25.84 (-8.7%)
		-17.0%	-5.3%	-16.2%	-5.1%

Table 2: Average precision of different merging procedures

where dbs indicates the number of the selected collections (all in our case), s_i the score achieved by the ith collection and S_m the mean score over all collections. According to our evaluation, the mean average precision results in a degradation of around 20.2% for short queries and 5.1% for long requests. It is interesting to note that both the raw-score merging and the CORI approach result in good performances when dealing with long requests yet a decrease in performance when using short requests.

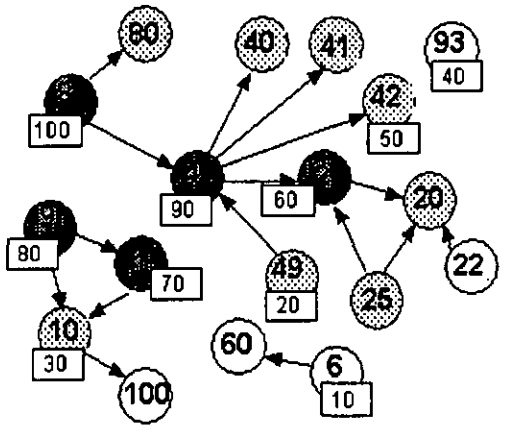

Figure 1: Starting situation for our link-based approaches

2. Link-based retrieval

Various retrieval strategies have been suggested io order to take account of hyperlinks, based on the assumption that links between documents indicate useful semantic relationships between related web pages [Kleinberg 98], [Brin 98], [Chakrabarti 99]. For example, Chakrabarti et al. [99] stated:

> "Citations signify deliberate judgment by the page author. Although some fractions of citations are noisy, most citations are to semantically related material. Thus the relevance of a page is a reasonable indicator of the relevance of its neighbors, although the reliability of this rule falls off rapidly with increasing radius on average." [Chakrabarti 99, p. 550-551]

With small variations, similar hypotheses are also cited by other authors [Kleinberg 98]. In order to verify the retrieval effectiveness of such assumptions, we have evaluated four different search strategies, namely our spreading activation approach in Section 2.1, our PAS search model in Section 2.2, Kleinberg's algorithm in Section 2.3 and the PageRank approach in Section 2.4. These search strategies will be described briefly using a small example.

As a first step, the search strategy computes the *similarity between the given query and the documents*, with values noted as $SIM(D_i, Q)$. These

values are depicted inside a rectangle in Figure 1. In this case, we can see that the first five retrieved documents are D_8, D_4, D_9, D_1 and D_2. At this point various retrieval schemes will take note of the hyperlinks so that the retrieval effectiveness might hopefully be improved.

2.1. Spreading activation

In a first link-based strategy, we chose the spreading activation (SA) approach [Crestani 00]. In that method, the degree of match between a web page D_i and a query, as initially computed by the IR system (denoted $SIM(D_i, Q)$), is propagated to the linked documents through a certain number of cycles using a propagation factor. We used a simplified version with only one cycle and a fixed propagation factor λ for all links. In that case, the final retrieval status value of a document D_i linked to m documents is computed according to the following equation:

$$RSV(D_i) = SIM(D_i, Q) + \lambda \cdot \sum_{j=1}^{k} SIM(D_j, Q)$$

Using all the incoming and outgoing links, and for different values of the parameter λ, in most cases did not result in retrieval improvement within the WT2g corpus [Savoy 01]. In order to be more selective in the spreading phase, we only consider in this study the best outgoing and the best incoming link for each of the k best-ranked documents (the constant k was fixed to 15 in this paper and the parameter λ to 0.05). But, what do we mean by the best link?

Instead of considering the m web pages linked to a given document, we only consider the incoming link coming from the best ranked document. For the outgoing links, we adopt a similar point of view, taking into account only the link starting from the given document to the best rank web page.

For example, based on Figure 1, we do not follow all outgoing from D_4 but we activate only the hyperlink to D_2 (the rank of this document is better than for the others). Similarly, the best incoming link is the link between D_8 to D_4. Fixing the parameter λ to 0.1 and k to 5, the final retrieval status value of D_4, noted $RSV(D_4)$, will be :

$$RSV(D_4) = SIM(D_4, Q) + \lambda \cdot SIM(D_2, Q) + \lambda \cdot SIM(D_8, Q) =$$
$$90 + 0.1 \cdot 60 + 0.1 \cdot 100 = 106$$

The similarity value of non-retrieved documents (e.g., D_{20} in our example) will be set according the similarity achieved by the last retrieved item (10 in our example, 1,000 in the evaluation). The evaluation of other web pages included in our example is given in Table 3.

2.2. Probabilistic argumentation system

In a second set of experiments, we used our probabilistic argumentation systems (PAS) [Picard 98], in which we used a simplified version of our approach, whereby the final retrieval status value of a document (or its degree of support, denoted $DSP(D_i)$) might only be affected by its direct neighbors. In this case we do not need to keep track of inferences, and can derive a simple formula which might be considered to be a more refined spreading activation method. Instead of propagating a document's similarity value, we propagated its probability of being relevant.

In this approach, we must therefore first compute the relevance probability of a document D_i. To achieve this, we suggest using logistic regression methodology [Bookstein 92] and the natural logarithm of its rank as an explanatory variable. Such a computation will be noted $p(D_i \mid rank)$ [Le Calvé 00] and in accordance with the following formula:

$$P[D_i \mid rank] = \frac{e^{\alpha+\beta \cdot \ln(rank)}}{1+e^{\alpha+\beta \cdot \ln(rank)}}$$

in which α et β are parameters set to 0.7 and -0.8 respectively.

In a second step, this probability of relevance will be modified according to the neighbors of a given document. The individual contribution of a linked document D_j to D_i is given by $[p(D_j \mid rank) \cdot p(link)]$, instead of the $[SIM(D_j, Q) \cdot \lambda]$ used with the spreading activation technique.

Just as with the spreading activation experiments, using all incoming or outgoing links did not demonstrate any improvement, except in some cases when using the WT2g test collection [Savoy 01]. We then decided to include only the most important sources of evidence, the same way as for spreading activation. For example, we considered the initial rank of document D_i, the best incoming document D_{in} and the best outgoing document D_{out}.

This link-based retrieval approach will thus multiply the probability of linked document relevance by the probability of the link, denoted $p(link_{in})$ for incoming hyperlinks or $p(link_{out})$ for outgoing links. The final degree of support corresponding to document D_i is computed as follow:

$$DSP(D_i) = 1 - (1 - p(D_i \mid rank)) \cdot [1 - p(D_{in} \mid rank) \cdot p(link_{in})] \cdot [1 - p(D_{out} \mid rank) \cdot p(link_{out})]$$

Fixing $p(link_{in})=0.1$ and $p(link_{out})=0.2$, and based on the situation depicted in Figure 1, computation of degree of support for Document 1 as follows:

$$DSP(D_1) = 1 - (1 - p(D_1 \mid rank)) \cdot [1 - p(D_9 \mid rank) \cdot p(link_{in})] \cdot [1 - p(D_{10} \mid rank) \cdot p(link_{out})] =$$
$$1 - (1 - 0.3991) \cdot [1 - 0.4554 \cdot 0.1] \cdot [1 - 0.2762 \cdot 0.2] =$$
$$1 - (0.6009) \cdot [0.95446] \cdot [0.94476] =$$
$$1 - 0.5418 = 0.4582$$

Table 4 lists other results pertaining to the best ten retrieved items of Figure 1. For the results based on the web test collection, link probabilities are fixed as $p(link_{in}) = 0.062$, $p(link_{out}) = 0.051$, probability estimates are defined in [Savoy 01]. Finally, documents not belonging to the top 1000 have a similarity value equal to the similarity value obtained for the 1000th retrieved item.

2.3. Kleinberg's algorithm

As a third link-based approach, we have applied Kleinberg's algorithm [Kleinberg 98]. In this scheme, a web page pointing to many other information sources must be viewed as a "good" hub while a document with many web pages pointing to it is a "good" authority. Likewise, a document that points to many "good" authorities is an even better hub while a web page pointed to by many "good" hubs is an even better authority.

For document D_i after c+1 iterations, the updated formulas for the hub and authority scores $H^{c+1}(D_i)$ and $A^{c+1}(D_i)$ are:

$$A^{c+1}(D_i) = \sum_{D_j=parent(D_i)} H^c(D_j)$$

$$H^{c+1}(D_i) = \sum_{D_j=child(D_i)} A^c(D_j)$$

Rank	D_i	$SIM(D_i, Q)$	D_i	$RSV(D_i)$
1	8	100	8	109
2	4	90	4	106
3	9	80	9	87
4	1	70	1	78
5	2	60	2	69
6	42	50	42	50
7	93	40	93	40
8	10	30	10	37
9	49	20	49	20
10	6	10	20	16
			6	10

Table 3: Retrieval status value obtained by the spreading activation

Rank	D_i	$SIM(D_i,Q)$	$p(D_i \mid rank)$	D_i	$DSP(D_i)$
1	8	100	0.6682	8	0.7038
2	4	90	0.5363	4	0.5982
3	9	80	0.4554	9	0.4989
4	1	70	0.3991	1	0.4582
5	2	60	0.3572	2	0.4211
6	42	50	0.3244	42	0.3244
7	93	40	0.2980	10	0.3051
8	10	30	0.2762	93	0.2980
9	49	20	0.2577	20	0.2690
10	6	10	0.2419	49	0.2577
11			0.2419	6	0.2419

Table 4: Computation of the degree of support of our PAS search model

which is computed for the k best-ranked documents (defined as the root set) retrieved by a classical search model, together with their children and parents (which defined the base set). The hub and authority scores were updated for five iterations (while the ranking did not change after this point), and a normalization procedure (dividing each score by the sum of all square values) was applied after each step.

As an example, we will refer to the initial situation shown in Figure 1. We fixed k = 5 and our root set was {D_8, D_4, D_9, D_1, D_2}, leading to the following base set {D_8, D_4, D_9, D_1, D_2, D_{80}, D_{40}, D_{41}, D_{42}, D_{20}, D_{25}, D_{49}, D_{10}}. Initially, the hub and authority score for each document is set to 1. In the first iteration, the hub score for D_4 corresponds to the sum of the authority values for its children (D_{40}, D_{41}, D_{42}, D_2) while its authority score is the sum of the hub scores of its parents (D_8, D_{49}). For other items belonging to the basic set, computation of these scores is depicted in Table 5.

After five iterations and using the normalization procedure, we obtained the ranked list depicted in Table 6. Taking the five best-ranked documents obtained by the traditional search engine into account and the top five documents retrieved according to the authority scores, we note that the intersection included only one item, namely D_2.

2.4. PageRank algorithm

Brin & Page [98] suggest a link-based search model called PageRank that first evaluated the importance of each web page based on its citation pattern. As for the spreading activation approach, the PageRank algorithm reranked the retrieved pages of a traditional search schemes according to the PageRank values assigned to the retrieved items.

In this approach, a web page will have a higher score if many web pages point to it. This value increases if there are highly scoring documents pointing to it. The PageRank value of a given web page D_i, value noted as $PR(D_i)$, having D_1, D_2, ... D_m pages pointing to D_i, is computed according to the following formula:

$$PR(D_i) = (1 - d) + d \cdot [(PR(D_1) / C(D_1)) + \ldots + (PR(D_m) / C(D_m))]$$

where d is a parameter (set to 0.85 as suggested by [Brin 98] and $C(D_j)$ are the number of outgoing links for web page D_j.

The computation of the PageRank value can be done using an iterative procedure (five iterations were computed in our case). After each iteration, each PageRank value was divided by the sum of all PageRank values. Finally, as initial values, $PR(D_i)$ were set to 1/N where N indicates the number of documents in the collection.

Based on our example, the result list achieved by using the PageRank algorithm is depicted in Table 8.

2.5. Evaluation

The retrieval effectiveness of the four link-based search model are shown in Table 9. From this table, it seems clear that links do not seem an appropriate source of information about document contents, and they seem to provide less information than do the bibliographic references or co-citation schemes used in our previous studies [Savoy 96]. The poor results depicted by Kleinberg's approach or PageRank algorithm raise some questions: Is our implementation without bugs? Can other teams confirm these findings? Have the underlying parameters the good values?

Our official runs were produced using the raw-score merging, where three were based only on the Title portion of the requests (NEtm, NENRtm, NENRtmLpas) and three were based on all logical sections of the queries (NEnm, NEnmLpas, NEnmLsa). Three of them were link-based retrievals (ending by Lpas or Lsa indicating the PAS or spreading activation approach).

For the two types of requests, our official runs included a spelling check performed automatically by the Smalltalk-80 system. This feature has a positive effect for short queries (e.g., 15.96 vs. 17.54 (+9.9%)) but not for long ones (25.99 vs. 24.99 (-3.8%)).

Conclusion

The various experiments carried out within the web track demonstrated that:

- Hyperlinks do not result in any significant improvement (at least as implemented in this study). Link information seems to be marginally useful when the retrieval system produces relatively high retrieval effectiveness;

- Pseudo-relevant feedback techniques (blind query expansion) usually result in significant improvement but setting the underlying parameters based on another test collection may lead to a decrease in retrieval effectiveness;
- Longer topic descriptions (Title, Description and Narrative) improve the retrieval performance significantly over short queries built only from the Title section;
- It seems that the raw-score approach might be a valid first attempt for merging result lists provided by the same retrieval model.

Acknowledgments

The authors would like to thank C. Buckley from SabIR for allowing us the opportunity to use the SMART system. This research was supported by the SNSF (Swiss National Science Foundation) under grant 21-58813.99.

D_i	$H^0(D_i)$	Author comput	$A^1(D_i)$	$A^0(D_i)$	Hub comput	$H^1(D_i)$
8	1		0	1	1 + 1	2
4	1	1 + 1	2	1	1 + 1 + 1 + 1	4
9	1		0	1	1 + 1	2
1	1	1	1	1	1	1
2	1	1 + 1	2	1	1	1
40	1	1	1	1		0
41	1	1	1	1		0
42	1	1	1	1		0
49	1		0	1	1	1
80	1	1	1	1		0
20	1	1 + 1	2	1		0
25	1		0	1	1 + 1	2
10	1	1 + 1	2	1		0

Table 5: Computation of the hub and authority scores for our example

Rank	D_i	$SIM(D_i, Q)$	D_i	$A^5(D_i)$	D_i	$H^5(D_i)$
1	8	100	2	0.1239	4	0.1501
2	4	90	42	0.0762	25	0.0723
3	9	80	41	0.0762	9	0.0241
4	1	70	40	0.0762	8	0.0241
5	2	60	20	0.0667	2	0.0222
6	42	50	4	0.0413	49	0.0148
7	93	40	10	0.0413	1	0.0148
8	10	30	80	0.0254	80	0
9	49	20	1	0.0254	42	0
10	6	10	9	0	41	0

Table 6: Computation of the hub and authority scores after five iterations

Rank	D_i	$SIM(D_i, Q)$	Rank	D_i	$PR(D_i)$
1	8	100	1	10	0.2710
2	4	90	2	4	0.2548
3	9	80	3	2	0.2146
4	1	70	4	1	0.1849
5	2	60	5	42	0.1797
6	42	50	6	93	0.15
7	93	40	7	49	0.15
8	10	30	8	9	0.15
9	49	20	9	8	0.15
10	6	10	10	6	0.15

Table 8: Ranked list obtained in our example by the traditional and the PageRank approach

	without normalizat.		with normalization	
D_i	$PR^1(D_i)$	$PR^5(D_i)$	$PR^1(D_i)$	$PR^5(D_i)$
1	0.1736	0.2138	0.1925	0.1849
2	0.1854	0.2863	0.2138	0.2146
4	0.2208	0.3413	0.2775	0.2548
6	0.15	0.15	0.15	0.15
8	0.15	0.15	0.15	0.15
9	0.15	0.15	0.15	0.15
10	0.2208	0.3954	0.2775	0.2710
20	0.2681	0.5846	0.3625	0.3547
22	0.15	0.15	0.15	0.15
25	0.15	0.15	0.15	0.15
40	0.1618	0.2225	0.1713	0.1797
41	0.1618	0.2225	0.1713	0.1797
42	0.1618	0.2225	0.1713	0.1797
49	0.15	0.15	0.15	0.15
60	0.1972	0.2775	0.235	0.2198
80	0.1736	0.2138	0.1925	0.1849
93	0.15	0.15	0.15	0.15
100	0.1972	0.4861	0.235	0.2762

Table 7: Computation of the PageRank values with and without normalization

	Average Precision				
Query Title Model	merge raw-score	SA	PAS	Kleinberg	PageRank
Okapi	17.59	14.64	17.57	0.18	2.82
OkapiR	17.81	14.59	17.76	0.19	2.79
OkapiXQ	15.96	13.43	15.91	0.17	2.37
OkapiNRXQ	15.87	13.48	15.85	0.17	2.69
Query T-D-N					
Okapi	26.56	23.80	26.43	0.36	3.09
OkapiNR	26.75	24.10	26.65	0.25	3.14
OkapiXQ	25.99	22.27	25.87	0.31	3.11
OkaNRXQ	25.93	22.57	25.82	0.25	3.13

Table 9: Average precision of different link-based approaches

Official run name	Corresponding run name	Average Pre.	# ≥ Median	# Best
NEtm	OKAPIXQ	17.54	41	3
NENRtm	OKAPIRXQ	17.43	41	2
NENRtmLpas	OKAPIRXQ + PAS	17.36	40	1
NEnm	OKAPIXQ	24.99	45	4
NEnmLpas	OKAPIRXQ + PAS	24.88	43	0
NEnmLsa	OKAPIRXQ + SA	21.85	41	0

Table 10: Summary of our official runs for the web track

References

[Bookstein 92] A. Bookstein, E. O'Neil, M. Dillon, D. Stephens: Applications of loglinear models for informetric phenomena. Information Processing & Management, 28(1), 1992, 75-88.

[Brin 98] S. Brin, L. Page: The anatomy of a large-scale hypertextual web search engine. WWW8, 1998, 107-117.

[Buckley 96] C. Buckley, A. Singhal, M. Mitra, G. Salton: New retrieval approaches using SMART. TREC-4, 1996, 25-48.

[Callan 95] J. P. Callan, Z. Lu, W. B. Croft: Searching distributed collections with inference networks. ACM-SIGIR'95, 21-28.

[Chakrabarti 99] S. Chakrabarti, M. Van den Berg, B. Dom: Focused Crawling: A new approach to topic-specific web resource discovery. WWW8, 1999, 545-562.

[Crestani 00] F. Crestani, P. L. Lee: Searching the web by constrained spreading activation. Information Processing & Management, 36(4), 2000, 585-605.

[Dumais 94] S. T. Dumais: Latent semantic indexing (LSI) and TREC-2. TREC'2, 1994, 105-115.

[Hawking 99] D. Hawking, N. Craswell, P. Thistlewaite, D. Harman: Results and challenges in web search evaluation. WWW8, 1999, 243-252.

[Kleinberg 98] J. Kleinberg: Authoritative sources in a hyperlinked environment. Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, 668-677.

[Kwok 95] K. L. Kwok, L. Grunfeld, D. D. Lewis: TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. TREC-3, 1995, 247-255.

[Lawrence 99] S. Lawrence, C. Lee Giles: Accessibility of information on the web. Nature 400 (6740), 1999, 107-110.

[Le Calvé 00] A. Le Calvé, J. Savoy: Database merging strategy based on logistic regression. Information Processing & Management, 36(3), 2000, 341-359.

[Picard 98] J. Picard: Modeling and combining evidence provided by document relationships using PAS systems. ACM-SIGIR'98, 182-189.

[Robertson 95] S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu: Large test collection experiments on an operational, interactive system: OKAPI at TREC. Information Processing & Management, 31(3), 1995, 345-360.

[Savoy 96] J. Savoy: Citation schemes in hypertext information retrieval. In Information retrieval and hypertext, M. Agosti, A. Smeaton (Eds), Kluwer, 1996, 99-120.

[Savoy 01] J. Savoy, J. Picard: Retrieval effectiveness on the web. Information Processing & Management, 2001, to appear.

[Selberg 99] E. W. Selberg: Towards comprehensive web search. Ph.D. Thesis, University of Washington, 1999.

[Voorhees 95] E. M. Voorhees, N. K. Gupta, B. Johnson-Laird: Learning collection fusion strategies. ACM-SIGIR'95, 172-179.

[Walker 98] S. Walker, S. E. Robertson, M. Boughamen, G. J. F. Jones, K. Sparck Jones: Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. TREC-6, 1998, 125-136.

[Xu 98] J. Xu, J. P. Callan: Effective retrieval with distributed collections. ACM-SIGIR'98, 112-120.

Appendix 1: Statistics describing our various sub-collections

Collection	WEB9.1	WEB9.2	WEB9.3	WEB9.4	WEB9
Size (in MB)	2,799 MB	2,754 MB	2,790 MB	2,690 MB	11,032 MB
# of documents	414,914	423,965	442,711	410,506	1,692,096
# of relevant doc.	749	600	608	660	2617
# of queries	46	44	43	46	50
mean	16.2826	13.6364	14.1395	14.3478	52.34
standard error	25.0986	21.826	21.3637	22.1873	84.1405
maximum	157	105	133	124	519
for # query	(#q:495)	(#q:495)	(#q:495)	(#q:495)	(#q:495)
minimum	1	1	1	1	1
for # query	(#q:461)	(#q:461)	(#q:464)	(#q:456)	(#q:473)
size invert. file doc.nnn	674.2 MB	642.1 MB	655.6 MB	635.4 MB	
# indexing terms	3,428,795	2,827,067	3,607,359	3,537,393	
max df	189,386	207,892	228,922	191,208	
Indexing time (real)	1:05:17	1:00:28	1:00:18	1:00:49	

Table A.1: Some statistics about the four sub-collections of the Web corpora

Conférence TALN 2000, Lausanne, 16-18 octobre 2000

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 metamoteurs (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 $\leq$ -0.6177 & nn $\leq$ 1) ALORS ne pas interroger;
SI (dr1 $\leq$ -0.6177 & nn > 1) ALORS interroger la collection;

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

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

Dans le cas présent, l'écart normalisé du degré de similarité du premier document extrait (drl), 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 CALVE 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:

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

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 · (tf_{ij} / max tf dans d_i)
l	new_tf = ln(tf_{ij}) + 1.0
L	new_tf = [ln(tf_{ij}) + 1.0] / [1.0 + ln(moyenne (tf dans d_i))]
n	new_wt = new_tf (pas de modification)
t	new_wt = new_tf · ln[n / df_j]
p	new_wt = new_tf · 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) · pivot + slope · 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

Recherche d'informations dans des sources distribuées

Jacques Savoy[1], Yves Rasolofo[1], Faïza Abbaci[2]

[1] Institut interfacultaire d'informatique
Pierre-à-Mazel 7, 2000 Neuchâtel (Suisse)

{Jacques.Savoy, Yves.Rasolofo}@unine.ch, site: www.unine.ch/info/

[2] Ecole nationale supérieure des Mines de Saint-Etienne
158 cours Fauriel, 42023 St-Etienne (France)

Abbaci@emse.fr, site : www.emse.fr

In Actes du XIXe congrès INFORSID, Genève (Suisse), mai 2001, pp. 237-252

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 fusion des résultats provenant des différents serveurs interrogés. Dans cet article, nous proposons une nouvelle stratégie de fusion et évaluons différentes approches basées sur un corpus de documents, des articles de quotidiens (2 GB) pour l'essentiel, et une seconde collection de pages Web de 10 Gb. Nos expériences démontrent que notre stratégie apporte une bonne performance comparée aux autres approches suggérées.

Abstract

The Web and digital libraries give us the opportunity to search among various information servers (collections or search engines) which lead us to the difficult problem of merging ordered lists provided by different information servers. In this paper, we suggest a new simple merging strategy and evaluate it using different search models and based on two test-collections, the first one containing news extracted from four different sources, and the second storing 10 GB of Web pages. Our evaluations demonstrate that the suggested merging approach shows interesting performance compared to other merging strategies.

Mots-clés / Keywords

Recherche d'informations distribuée, moteurs de recherche, fusion de collections, évaluation

Distributed information retrieval, search engines, merging procedures, evaluation

1 Introduction

Par rapport à la recherche classique d'informations œuvrant sur un ensemble statique de documents [SALT 1989], le Web pose de nouveaux défis aux concepteurs de moteurs de recherche. D'abord, le volume d'informations disponible sur Internet augmente sensiblement de mois en mois. De plus, cette information se présente sous différentes formes telles que texte (écrit dans plus d'une centaine de langues), image, graphique, animation auxquelles s'ajoutent le son et la vidéo. Ensuite, la qualité des renseignements présentés se révèle très variable et le contenu est parfois éphémère, variant rapidement au cours du temps. Enfin, les sites sont distribués à travers la planète et des robots doivent parcourir le réseau afin de les découvrir et permettre aux moteurs de recherche de les indexer.

En plus de ces difficultés inhérentes à Internet, les moteurs de recherche disponibles connaissent plusieurs lacunes [HAWK 1999]. Ainsi, même des systèmes disposant d'une capacité en disque très importante n'indexent qu'une fraction de toute l'information disponible et la couverture de ces moteurs n'augmente pas aussi rapidement que la taille du Web. Par exemple, Northern Light, moteur disposant selon les dernières estimations de la plus grande couverture du Web, n'indexe que 16 % des pages tandis que Lycos couvre environ 2,5 % [LAWR 1999]. Deuxièmement, cette difficulté s'accroît si l'on tient compte du fait que les index doivent être dupliqués pour des raisons diverses (temps d'accès, temps de réponse, largeur de bande passante disponible, pannes). Troisièmement, les pages indexées sont souvent obsolètes ou ont disparu (ce qui représente environ 5 % des réponses). Quatrièmement, la page utilisée lors de l'indexation ne correspond pas à la page actuelle car l'auteur a modifié, parfois radicalement, son contenu sémantique. Cinquièmement, une distinction claire entre les sources dignes de foi et les autres n'est actuellement pas possible. Sixièmement, plusieurs sources ou collections de documents ne sont pas librement accessibles (*copyrights*, collections privées). Finalement, la qualité des réponses obtenues par ces moteurs se révèle quelque peu décevante [GORD 1999].

Le propos de cet article est d'analyser la performance de diverses stratégies d'indexation et de dépistage sur un corpus d'articles de presse (environ 2 GB) et sur un ensemble assez important de pages extraites du Web (un peu plus de 10 GB). Plus particulièrement, nous nous intéresserons à différentes approches pouvant gérer une partition de l'ensemble de nos sources documentaires partition nécessaire si l'on désire indexer un nombre important de pages tout en respectant la taille maximale de 2 GB par fichier inversé. Ce problème de collaboration entre différents moteurs de recherche se retrouve également dans les bibliothèques numériques ou les métamoteurs de recherche. Dans ces derniers cas, chaque fond documentaire ou serveur d'information peut être interrogé individuellement mais une réponse adéquate nécessite la fusion des résultats provenant de toutes les sources. Ce problème, nommé problème de fusion de collections, constitue le sujet principal de cet article.

Dans un premier temps, nous décrirons les deux collections de documents utilisées et nous présenterons une première évaluation en optant pour une approche centralisée dans laquelle tous les documents disponibles sont regroupés pour former une seule banque de documents. Ensuite, nous distribuerons nos sources selon différents critères et nous analyserons la performance obtenue en interrogeant les différentes collections ainsi créées. Dans cette troisième partie, nous présenterons également une nouvelle stratégie pour résoudre efficacement et simplement le problème de fusion de collections.

2 Approche centralisée

La recherche d'informations s'appuie sur une longue tradition empirique encourageant les chercheurs à évaluer leur système de dépistage sur la base de collections-tests. Ces corpus disposent d'un nombre plus ou moins important de documents et un ensemble de requêtes. De plus, pour chaque requête, on dispose de la liste des documents jugés pertinents, jugement posé par un être humain et non une machine. Ces renseignements constituent la partie la plus onéreuse et délicate dans la mise au point d'une collection-test.

Dans cet article, nous évaluerons différents modèles de recherche à l'aide de deux collections afin d'obtenir une plus grande validité pour nos résultats. En effet, les caractéristiques, souvent inconnues, d'un corpus peuvent favoriser une approche au détriment d'une autre. De plus, nos investigations ne seront pas limitées à un seul modèle de dépistage de l'information, mais nous évaluerons plusieurs stratégies de recherche permettant ainsi d'obtenir une meilleure compréhension de l'efficacité de divers systèmes.

2.1. Les collections-tests

Afin d'évaluer expérimentalement nos propositions, nous avons choisi un corpus de documents rédigés en langue anglaise correspondant 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). Avec ce corpus, nous disposons de 50 requêtes couvrant non pas un domaine limité mais présentant un éventail assez large de thèmes (par exemple «Estonia, economy», «suicides», «airport security», «osteoporosis», «cosmic events»). Comme l'ordinateur ne possède que le titre des besoins d'information, le texte disponible se révèle très bref, comportant en moyenne, deux mots (écart type de 0,96). De plus, les termes employés sont très souvent ambigus et très fréquents dans les documents disponibles.

Afin de compléter cette première évaluation, nous avons également repris la collection de pages Web utilisée lors de la neuvième conférence TREC, corpus nommé TREC9. Cet ensemble comprend 1 692 096 pages Web écrites en anglais pour un volume de 11 033 MB. Cette seconde collection possède donc un volume approximativement six fois supérieur au corpus précédent et présente des documents de nature différente. Dans ce cas, il n'y pas de vrai contrôle éditorial sur le contenu et le nombre de fautes d'orthographe est plus important.

Avec ce second corpus, nous disposons aussi de 50 requêtes couvrant des domaines variés (par exemple «parkinson's disease», «hunger», «baltimore», «how e-mail bennefits businesses», «mexican food culture»). Des exemples plus complets sont repris dans l'annexe 1. Dans ce cas aussi, les requêtes correspondent à des exemples réels soumis au moteur de recherche Excite. Comme l'ordinateur ne possède que le titre des besoins d'information, le texte disponible se révèle très bref comportant 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. Des statistiques plus complètes sur ces deux collections sont reprises dans l'annexe 2.

2.2. Performance de l'approche centralisée

Décrire un modèle de dépistage de l'information correspond à définir comment sont représentés les documents d'une part et, d'autre part, les requêtes. Dans les différentes stratégies étudiées dans cet article, cette représentation se limite à un ensemble de termes simples pondérés. Finalement, un modèle spécifie la manière dont l'appariement entre la requête et les documents se réalise. Pour ce dernier aspect, et dans toutes nos expériences, le degré de similarité de chaque document avec la requête (ou son degré de pertinence jugé par la machine) est obtenu par le calcul du produit interne [SALT 1989, p. 318]. Ainsi si nous nommons w_{ij} le poids reflétant l'importance du terme d'indexation t_j, j = 1, 2, ..., t, dans un document D_i, et par w_{qj} le poids de ce même terme d'indexation t_j dans la requête Q, le degré de similarité entre le document D_i et la requête Q se mesure selon le produit interne à l'aide de la formule suivante :

$$\text{Sim}(D_i, Q) = \sum_{j=t}^{t} w_{ij} \cdot w_{qj}$$

Pour indexer un document ou une requête, différentes stratégies ont été proposées. La manière la plus simple est d'effectuer une indexation binaire (approche notée «bnn»). Les deux valeurs possibles pour w_{ij} et w_{qj} seront le 1 (si le terme décrit bien le contenu sémantique du document ou de la requête) et 0. Si nous représentons les documents et les requêtes avec cette forme («doc: bnn, requête: bnn»), le degré de similarité de chaque document (calculé selon le produit interne) indiquera le nombre de termes communs entre celui-ci et la requête.

Evidemment, d'autres stratégies d'indexation existent et on peut tenir compte de la fréquence d'occurrence des termes d'indexation dans le document ou la requête (approche «nnn»). Ainsi, si un mot se répète trois fois dans une page, la valeur w_{ij} associée possédera la valeur 3. L'importance de ce terme sera donc plus grande qu'un mot apparaissant une seule fois. Dans le modèle «doc: nnn, requête: nnn», le calcul du degré de similarité 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 du terme dans le document et de l'inverse de sa fréquence documentaire (valeur notée idf, inversement proportionnel au nombre de documents dans lesquels ce terme apparaît). Ainsi, si un mot apparaît dans de très nombreux documents, son poids lors de l'indexation diminuera. En effet, un tel terme ne permettra pas à l'ordinateur de distinguer les documents pertinents des autres. Finalement, dans cette stratégie «doc: ntc, requête: ntc», les poids sont normalisés selon la formulation du cosinus (voir annexe 3 pour les détails de ces formules d'indexation) afin que chaque valeur attribuée à w_{ij} ou w_{qj} soit comprise entre 0 et 1.

Ces dernières années, et en particulier sous l'impulsion des conférences TREC, d'autres modèles de dépistage de l'information ont vu le jour, entre autres, diverses variantes du modèle vectoriel et en particulier les stratégies «doc: Lnu, requête: ltc» [BUCK 1996], «doc: dnu, requête: dtc» [SING 1999] ou l'approche «doc: atn, requête: ntc». Ces stratégies tendent à attribuer une importance de moins en moins forte à la fréquence d'occurrence par le recours au logarithme. Ainsi, si la fréquence d'apparition d'un terme passe de 2 à 3, ces méthodes considéreront que cet accroissement unitaire doit être valorisé de manière plus importante que le passage de 10 à 11.

Depuis quelques années, l'approche probabiliste Okapi [ROBE 1995] rencontre également un grand succès au niveau de la qualité de la performance obtenue. Ce modèle sera donc aussi repris dans nos évaluations et la formulation précise de ces différentes formules d'indexation est

indiquée dans l'annexe 3. Comme précédemment, ce dernier modèle tient compte de la fréquence d'occurrence, de l'inverse de la fréquence documentaire et de la longueur des documents. Pour ce dernier aspect, ce modèle de dépistage privilégie les documents courts au détriment des documents plus longs. En effet, ces derniers, possédant plus de termes, ont plus de chance d'être dépistés mais présentent aussi plusieurs thèmes dont un seul a peut-être un rapport avec la requête.

Comme méthodologie d'évaluation, nous avons retenu la précision moyenne [SALT 1983, Section 5.2] mesurant la qualité de la réponse fournie par l'ordinateur, mesure utilisée par la conférence TREC. L'évaluation des différentes stratégies de dépistage retenues au regard de nos deux collections-tests est indiquée dans la table 1. 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.

	Précision moyenne (% changement)	
collection	TREC8	TREC9
nombre doc. pertinents	4 728	2 617
modèle	50 requêtes	50 requêtes
doc:Okapi, requête: npn	25,66	19,86
doc: dnu, requête: dtn	24,03 (-6,4 %)	15,42 (-22,4 %)
doc: atn, requête: ntc	22,75 (-11,3 %)	14,59 (-26,5 %)
doc: Lnu, requête: ltc	21,63 (-15,7 %)	16,81 (-15,4 %)
doc: lnc, requête: ltc	14,44 (-43,7 %)	3,79 (-80,9 %)
doc: ltc, requête: ltc	13,60 (-47,0 %)	5,20 (-73,8 %)
doc: ntc, requête: ntc	12,21 (-52,4 %)	8,00 (-59,7 %)
doc: bnn, requête: bnn	11,51 (-55,1 %)	5,13 (-74,2 %)
doc: nnn, requête: nnn	4,47 (-82,6 %)	4,06 (-79,6 %)

Table 1 : Précision moyenne des moteurs de recherche sur les deux corpus (approche centralisée)

Sur la base de la table 1, on notera que le modèle probabiliste Okapi présente la meilleure précision moyenne et que cette performance se révèle significativement supérieure aux autres approches (différence de précision moyenne supérieure à 5 %). En deuxième position, on trouve l'approche «doc: dnu, requête: dtn» pour le corpus TREC8 ou «doc: Lnu, requête: ltc» pour la collection TREC9 extraite du Web. Ces résultats indiquent également que le modèle vectoriel classique «doc: ntc, requête: ntc» occupe seulement le septième rang pour le corpus TREC8 et le cinquième dans le cadre de la collection TREC9. Nous avons également été étonné de trouver la stratégie binaire «doc: bnn, requête: bnn» dans une meilleure position que le modèle «doc: nnn, requête: nnn».

Les requêtes étant différentes d'un corpus à l'autre, une comparaison directe entre les performances sur les collections TREC8 et TREC9 n'est donc pas possible. Nous remarquons que les modèles vectoriels «doc: lnc, requête: ltc» ou «doc: ltc, requête: ltc» [BUCK 1995] ont plutôt été conçus pour des collections de documents possédant peu de fautes d'orthographe. Pour ces approches, la précision moyenne sur des pages Web se dégradent fortement. De plus, l'approche basée sur une indexation binaire («doc: bnn, requête: bnn») semble aussi connaître une baisse sensible de performance lorsque cette stratégie est utilisée sur le Web.

3 Fusion de collections

Dans un premier temps, l'ensemble des documents était réuni pour former une seule banque de documents. Cette approche constituait un axiome dans la recherche d'informations classique [SALT 1989]. Or, le corpus TREC8 peut se diviser logiquement en différentes sources correspondant aux différents quotidiens dont le volume est assez variable (voir annexe 2). Cette situation reflète bien le cas d'une bibliothèque numérique possédant plusieurs sources d'information qui peuvent être interrogées en même temps par le même moteur. Ainsi, en réponse à une requête d'un usager, le système d'interrogation envoie cette demande à l'ensemble des moteurs de recherche. Ces derniers calculent leur réponse et la retournent au système d'interrogation. Ce dernier peut soit présenter la liste des réponses de chaque source séparément, soit fusionner ces listes pour n'en visualiser qu'une seule. La manière de générer une seule liste en ordonnant les résultats en fonction de leur pertinence par rapport à la requête soumise correspond au problème de fusion des collections. Ce phénomène se rencontre également dans les métamoteurs de recherche dont la qualité de réponse dépend fortement de la stratégie de fusion de collections utilisée.

Dans le cadre de la collection TREC9, le problème est quelque peu différent car une partition logique ne se justifie pas vraiment. Mais ce corpus possède un volume très important de texte, de l'ordre de 10 GB et face à ce volume considérable d'information, il nous est impossible de créer un seul fichier inversé. En effet, la taille de ce dernier dépasse la limite de 2 GB. Nous avons donc décidé de former huit collections possédant chacune environ 1,2 GB d'information (voir l'annexe 2 pour plus de détails). En présence d'une requête, il faut interroger ces huit serveurs utilisant la même stratégie d'indexation et de dépistage, obtenir huit listes de résultats et les fusionner afin de retourner une seule liste ordonnée à l'usager.

3.1. Stratégies de fusion

Dans cette partie, nous admettons qu'un ensemble de serveurs doivent collaborer via un système d'interrogation (ou courtier) afin de répondre à une requête de l'usager. Les serveurs mémorisent un fond documentaire particulier (TREC8) ou une partie d'un ensemble très volumineux d'information (TREC9). Le système d'interrogation reçoit la requête de l'utilisateur et, sur la base des informations contenues dans cette demande et celles qu'il possède des différents serveurs, il sélectionne les serveurs qu'il doit interroger. Dans le cadre de cet article, nous n'avons pas introduit de processus de sélection et notre courtier interroge toujours tous les serveurs possédant la même stratégie d'indexation et de dépistage. Le courtier transforme au besoin la requête reçue dans un format lisible par chacun des serveur, par exemple en recourant au protocole Z90.50 [KAHL 1993] ou en utilisant le modèle STARTS [GRAV 1997].

serveur 1			serveur 2			serveur 3		
rang	document	sim.	rang	document	sim.	rang	document	sim.
1	LA123	1.2	1	FR453	0.8	1	FT567	1.6
2	LA673	1.0	2	FR012	0.75	2	FT195	1.3
3	LA946	0.72	3	FR673	0.65	3	FT548	0.9
4	LA765	0.6				4	FT649	0.7
...	...	...				...	...	...
8	LA546	0.2				...	...	...
						12	FT940	0.1

Figure 1 : Exemple de listes obtenues par le courtier

Chaque serveur interrogé répond par une liste ordonnée de documents répondant au mieux à la requête selon ses connaissances propres (la figure 1 présente un tel exemple dans lequel le premier serveur retrouve 8 documents, le deuxième 3 et le dernier 12). Ayant reçu ces listes, le courtier doit sélectionner et fusionner les différentes références contenues dans ces listes afin de ne présenter à l'usager qu'une seule liste de documents répondant à sa requête. Certes, il existe quelques exemples de métamoteurs qui ne procèdent pas à cette fusion comme, par exemple, www.all4one.com qui retourne dans quatre fenêtres distinctes les résultats obtenus par les moteurs AltaVista, Lycos, HotBot et Excite. Mais dans la très grande majorité des cas, une fusion des listes obtenues est effectuée pour permettre à l'utilisateur de consulter une seule liste de documents.

Afin de résoudre ce problème de fusion, différentes propositions ont été avancées. Comme première approche, nous pouvons admettre que chaque serveur retournera un nombre approximativement égal de documents pertinents et que ceux-ci se retrouvent distribués de manière identique dans les réponses obtenues [VOOR 1995]. Selon ces hypothèses, nous pouvons construire la liste finale en prenant un élément dans chaque liste puis en recommençant. Cette stratégie, nommée «à chacun son tour», se rencontre souvent dans certains métamoteurs [SELB 1999], en particulier par les premiers outils de ce type disponibles sur Internet. Sur la base des listes indiquées dans la figure 1, un exemple de résultat obtenu par cette stratégie de fusion est présenté dans la figure 2.

rang	document
1	LA123
2	FR453
3	FT567
4	LA673
5	FR012
6	FT195
7	...

Figure 2 : Exemple de fusion «à chacun son tour»

rang	document	sim.
1	FT567	1.6
2	FT195	1.3
3	LA123	1.2
4	LA673	1.0
5	FT548	0.9
6	FR453	0.8
7	...	

Figure 3 : Exemple de fusion par le score

Cependant, en plus du rang de chaque article dépisté, les moteurs de recherche fournissent parfois le degré de similarité entre la requête et le document (ou un degré estimé de la pertinence du document retourné). 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 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». En reprenant les données de la figure 1, cette stratégie de fusion donne le résultat présenté dans la figure 3. 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 un autre. Ce phénomène risque d'invalider cette approche.

Comme variante à cette démarche, nous pouvons considérer que le système d'interrogation ne devrait pas comparer directement le degré de similarité absolu mais calculer une valeur relative. Nous évitons ainsi de comparer directement des pommes avec des oranges car les serveurs calculent les degrés de similarité indépendamment les uns des autres. Nous pouvons normaliser le degré de similarité obtenue par chaque document dans chaque collection en le divisant par le degré de similarité maximum obtenu pour le corpus considéré (stratégie du degré de similarité normalisé). Cette stratégie s'avère bien adaptée en présence de serveurs recourant à des stratégies d'indexation et de dépistage différentes. En reprenant les données de notre exemple, nous obtenons une liste unique présentée dans la figure 4.

rang	document	sim.
1	LA123	100
2	FR453	100
3	FT567	100
4	FR012	93.75
5	LA673	83.33
6	FR673	81.25
7	FT195	81.25
8	...	

Figure 4 : Exemple de fusion par le degré de similarité normalisé

Comme quatrième stratégie de fusion, on peut considérer le système CORI [LARK 2000], [XU 1998], [CALL 1995]. Dans ce cas, au lieu de classer des documents selon leur degré de similarité avec la requête, le système doit d'abord classer des collections sur la base de la requête soumise. Dans ce cas, au lieu de rechercher des documents dans une collection en fonction d'une requête, on doit dépister des collections sur la base d'une requête de l'utilisateur. Vu sous cet angle, nous pouvons utiliser les principes de la recherche d'informations décrits à la section 2.2 en les adaptant au contexte de la recherche de collections.

Ainsi, au lieu de considérer la fréquence d'occurrence d'un terme t_j dans un document D_i, nous allons tenir compte du nombre de documents dans la i^e collection contenant le terme t_j (valeur notée df_i). De même, nous remplacerons la fréquence documentaire (nombre de documents dans le terme t_j apparaît) par la variable cf_j indiquant le nombre de collections dans lesquelles le terme t_j apparaît. Dans le système CORI, l'importance du terme t_j dans le score de la i^e collection db_i s'évalue par la formule suivante :

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

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, cf_j le nombre de collections contenant le terme t_j, ldb_i le nombre de termes contenus dans la i^e collection, 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 obtenu pour une collection est simplement la moyenne de ces valeurs pour tous les termes contenus dans la requête (valeur notée S_i pour la i^e collection).

Sur la base du score S_i de la i^e collection, le système CORI calcule un facteur de pondération, noté w_i, comme suit :

$$w_i = 1 + db \cdot [(S_i - Sm) / Sm]$$

où db indique le nombre de corpus sélectionnés, S_i le score de la i^e collection considérée et Sm le score moyen des collections. Lors de la fusion, le degré de similarité de chaque document extrait de la i^e collection est multiplié par ce coefficient w_i. En reprenant l'exemple de la figure 1 et en attribuant un facteur de pondération de 0.9 pour la première collection, 0.5 pour la deuxième et 1.2 pour la troisième, nous obtenons la fusion décrite dans la figure 5.

rang	document	sim.
1	FT567	1.92
2	FT195	1.56
3	LA123	1.08
4	FT548	1.08
5	LA673	0.9
6	FT649	0.84
7	...	

Figure 5 : Exemple de fusion obtenu par le système CORI

3.2. Notre stratégie de fusion

Notre stratégie de fusion fonctionne également en deux temps. En premier lieu, nous attribuons un score à chaque collection interrogée. Ce score est proportionnel au nombre de documents extraits (ou longueur de la liste L_i retournée) par ce corpus par rapport à l'ensemble des documents dépistés par tous les corpus. Ainsi, si une collection retourne peu de documents, comme c'est le cas du deuxième serveur dans la figure 1, nous attribuerons une importance faible aux documents provenant de cette collection. Par contre, si un serveur extrait un nombre conséquent d'articles, nous estimons que cette collection contient très certainement des informations pertinentes par rapport à la requête. Nous pondérerons donc plus fortement les documents provenant de ce serveur. Afin de définir une formule correspondant à ces souhaits, le score S_i de la i^e collection parmi les db serveurs interrogés s'évalue selon la formule suivante :

$$S_i = \ln [1 + ((L_i \cdot k) / \Sigma_{j=1 \text{ à } db} L_j)]$$

dans laquelle k est une constante (fixée à 600 dans nos évaluations), L_j le nombre de documents retournés par la j^e collection. Dans notre modèle, le recours au logarithme naturel (ln) se justifie par notre volonté d'attribuer une pondération de moins en moins forte par rapport au nombre de documents retournés. Ainsi, si ce nombre passe de 10 à 11, nous considérons que cet accroissement unitaire doit être valorisé de manière plus importante que le passage de 210 à 211.

Sur la base de ce score S_i défini pour chaque serveur, notre algorithme de fusion calcule un facteur de pondération, noté w_i, pour chaque collection selon l'équation suivante :

$$w_i = 1 + [(S_i - Sm) / Sm]$$

dans laquelle S_i indique le score obtenu par la i^e collection considérée et Sm le score moyen des collections. Durant la phase de fusion des listes retournées par les différents corpus, le degré de similarité de chaque document est multiplié par ce coefficient w_i indiquant l'importance relative à attribuer à chaque collection.

En reprenant notre exemple de la figure 1, nous pouvons calculer le score S_i de chacun des serveurs, puis le facteur de pondération w_i attachée à chaque collection (voir figure 6). Ce dernier va nous servir à multiplier les différents degrés de similarité au sein de chacune des collections. Finalement, nous pouvons effectuer le tri selon ce nouveau degré de similarité.

serveur 1			serveur 2			serveur 3		
$L_1 = 8$ $S_1 = \ln [1 + ((8 \cdot 600) / 23)]$ $= 5.346$ $Sm = 5.156$ $w_1 =$ $1 + [(5.346 - 5.156) / 5.156]$ $= 1.037$			$L_2 = 3$ $S_2 = \ln [1 + ((3 \cdot 600) / 23)]$ $= 4.373$ $w_2 =$ $1 + [(4.373 - 5.156) / 5.156]$ $= 0.848$			$L_3 = 12$ $S_3 = \ln [1 + ((12 \cdot 600) / 23)]$ $= 5.750$ $w_3 =$ $1 + [(5.750 - 5.156) / 5.156]$ $= 1.115$		
rang	document	sim.	rang	document	sim.	rang	document	sim.
1	LA123	1.244	1	FR453	0.678	1	FT567	1.784
2	LA673	1.037	2	FR012	0.636	2	FT195	1.450
3	LA946	0.747	3	FR673	0.551	3	FT548	1.004
4	LA765	0.622				4	FT649	0.781
...	...	...				...	...	...
8	LA546	0.207				...	...	...
						12	FT940	0.112

rang	document	sim.
1	FT567	1.784
2	FT195	1.450
3	LA123	1.244
4	LA673	1.037
5	FT548	1.004
6	FT649	0.781
7	...	

Figure 6 : Exemple de fusion obtenu par notre stratégie

Comparé au système CORI, notre approche possède l'avantage d'une plus grande simplicité. Elle ne requiert pas de connaissance préalable au niveau du courtier. Ainsi, notre approche ne nécessite pas d'analyser, pour chaque terme de la requête, le nombre de documents indexés par celui-ci dans chaque corpus (df_i). Cette information n'étant pas disponible dans la très grande majorité des moteurs de recherche, le système CORI exige soit que le courtier gère des fichiers inversés, soit que l'on ajoute cette possibilité d'interrogation au moteur interrogeant un corpus. De plus, des renseignements supplémentaires doivent être connus comme la longueur de chaque collection (ldb_i) ou la longueur moyenne des collections à interroger. Dans un environnent dynamique, le volume des divers corpus augmentera certainement de manière différente nécessitant la mise à jour de ces valeurs au niveau du système d'interrogation. Finalement, le nombre de constantes entrant dans notre formule est limité à l'unité contrairement au système CORI. Si la simplicité plaide en faveur de notre approche, est-ce que sa performance ne sera pas plus faible, voire significativement plus basse que l'approche du système CORI ? Cette question sera abordée dans la prochaine section.

3.3. Evaluation

A priori, il se révèle difficile de sélectionner la meilleure stratégie de fusion de collections, chacune présente un intérêt, une justification théorique et un temps de calcul raisonnable. Afin d'obtenir une meilleure vue d'ensemble de ces différentes approches, nous les avons évaluées avec nos deux collections-tests. Si pour le corpus TREC8, nous avons opté pour une partition logique selon les sources de documents (FT, FR, FBIS, LA Times), cette approche ne se révèle pas vraiment pertinente pour la collection TREC9. Dans ce cas, nous avons généré huit collections possédant un nombre approximativement égal de documents. Selon les statistiques disponibles dans l'annexe 2, on remarquera que, pour nos deux collections-tests, chacune des collections ainsi générée ne possède pas des réponses pertinentes pour toutes les requêtes. Ainsi, si le corpus du *Financial Times* (FT) possède au moins un article pertinent pour 49 des 50 requêtes, le *Federal Register* (FR) ne dispose de réponses adéquates que pour 18 requêtes. La situation se présente de manière assez similaire pour le corpus TREC9.

Lors de l'évaluation, la requête soumise est traitée par chacune des collections et la liste des documents extraits est transmise au système d'interrogation. Selon les informations fournies dans l'annexe 2, seulement deux requêtes n'arrivent pas à extraire de pages dans le corpus TREC9. Ces requêtes possèdent une faute d'orthographe et le système de dépistage n'arrive pas à faire un appariement (requête 464 «nativityscenes» et requête 487 «angioplast7»).

Selon les résultats reportés dans les tables 2a et 2b, la stratégie «à chacun son tour» permet d'obtenir une précision moyenne d'environ 26 % à 29 % inférieure à l'interrogation d'une collection unique regroupant tous les documents. Des travaux antérieurs [VOOR 1995], [CALL 1995] basés sur une collection et une seule stratégie de dépistage, annonçaient une détérioration de l'ordre de 40 %, soit un peu plus élevée que dans nos deux collections. Nous pouvons donc affirmer que chaque collection contient un nombre variable de documents pertinents et que ceux-ci ne se retrouvent pas distribués de manière identique dans les réponses obtenues des serveurs.

	Précision moyenne					
stratégie fusion modèle	collection centralisée TREC8	«à chacun son tour»	degré de similarité	degré de similarité normalisé	CORI k=200,b=,75 defB=0,4	notre approche
Okapi, req:npn	25,66	18,84	23,50	20,74	24,16	24,62
doc:dnu, req:dtn	24,03	17,40	22,20	17,69	22,68	23,53
doc:atn, req:ntc	22,75	16,18	20,43	16,59	21,38	20,47
doc:Lnu, req:ltc	21,63	15,31	19,35	15,05	21,44	19,59
doc:lnc, req:ltc	14,44	10,23	12,35	10,45	14,41	12,51
doc:ltc, req:ltc	13,60	10,24	12,21	10,36	13,23	12,71
doc:ntc, req:ntc	12,21	8,17	10,73	8,33	11,85	11,12
doc:bnn,req:bnn	11,51	7,58	11,36	9,38	11,02	12,47
doc:nnn,req:nnn	4,47	4,37	4,47	5,11	4,60	4,62
perte moyenne en %		-26,18 %	-8,32 %	-21,16 %	-2,84 %	-4,75 %

Table 2a : Précision moyenne de différentes stratégies de fusion (TREC8)

La fusion par le degré de similarité dont l'efficacité est indiquée dans la quatrième colonne des tables 2a et 2b démontrent que cette approche ne détériore que peu la performance moyenne (perte moyenne de 4,5 % à 8 %). 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. On remarque également qu'il n'y a pas de différence notable

entre nos deux corpus tests, ce qui valide cette approche. En revanche, l'approche du degré de similarité normalisé présente une performance peu attrayante (perte de 21 % à 31 %).

Selon nos expériences, l'efficacité moyenne du système CORI présente une baisse moyenne de 3 à 4 % et la perte de précision moyenne de notre approche se situe également vers les 4 %. Si nous nous limitons au meilleur modèle de dépistage, soit le modèle probabiliste Okapi, l'approche CORI fournit une baisse de précision moyenne de 5,8 % sur le corpus TREC8 (de 25,66 à 24,16) et de -7 % (de 19,86 à 18,47) dans le cadre de la collection TREC9. Dans ces deux cas, notre approche pour la fusion redonne de meilleurs résultats; sur le corpus TREC8 on passe d'une précision moyenne de 25,66 à 24,62, soit une baisse de 4 %, et pour la collection TREC9, la performance moyenne diminue de 19,86 à 19,32 (perte de 2,7 %).

	Précision moyenne					
stratégie fusion modèle	collection centralisée TREC9	«à chacun son tour»	degré de similarité	degré de similarité normalisé	CORI k=200,b=,75 defB=0,4	notre approche
Okapi, req:npn	19,86	12,38	18,10	12,72	18,47	19,32
doc:Lnu, req:ltc	16,81	9,81	16,11	9,36	15,37	15,03
doc:dnu, req:dtn	15,42	10,37	14,75	9,49	15,06	15,17
doc:atn, req:ntc	14,59	9,93	13,82	9,19	14,00	14,49
doc:ntc, req:ntc	8,00	5,37	8,17	5,19	8,05	7,90
doc:ltc, req:ltc	5,20	4,40	4,79	4,14	5,05	4,79
doc:bnn,req:bnn	5,13	3,91	5,04	4,39	4,71	4,84
doc:nnn,req:nnn	4,06	2,89	3,95	2,71	3,99	4,02
doc:lnc, req:ltc	3,79	3,21	3,48	2,88	3,64	3,54
perte moyenne en %		-28,91 %	-4,56 %	-31,44 %	-4,23 %	-4,22 %

Table 2b : Précision moyenne de différentes stratégies de fusion (TREC9)

4 Conclusion

Sur la base de nos expériences basées sur deux collections-tests, neuf stratégies de dépistage et des requêtes extrêmement courtes (mais similaires à celles rencontrées sur le Web), les conclusions suivantes peuvent être tirées :

1. le modèle probabiliste Okapi présente une précision moyenne très attractive. Il propose significativement la meilleure réponse dans le cadre de nos deux collections-tests œuvrant sur des volumes importants (2 GB et 10 GB);
2. le modèle vectoriel classique «doc: ntc, requête: ntc» ne propose pas une stratégie d'indexation et de dépistage efficace. Or cette approche (dénotée parfois «tf idf») est souvent recommandée;
3. la fusion selon le degré de similarité offre une alternative intéressante lorsque l'on doit fusionner plusieurs listes de résultats obtenues par le même processus d'indexation et le même moteur de recherche. Cette situation reflète bien le cas des bibliothèques numériques;
4. notre stratégie de fusion propose une alternative simple et efficace, meilleure que la fusion selon le degré de similarité et ayant une performance similaire au modèle CORI.

Cependant, le cadre de nos investigations correspondait soit à un moteur de recherche possédant plusieurs fichiers inversés, soit à une bibliothèque numérique œuvrant sur un ensemble disjoint de sources documentaires interrogées avec le même moteur de dépistage de l'information. Nos travaux en cours tentent de résoudre le problème de fusion en admettant que les différents

corpus sont gérés par des stratégies d'indexation et de recherche différentes, situation classique des métamoteurs de recherche (voir notre métamoteur, encore en phase expérimentale, à l'adresse www.unine.ch/info/News).

Remerciements

Cette recherche a été subventionnée en partie par le FNS avec le subside 21-58 813.99 (J. Savoy et Y. Rasolofo) et par la Région Rhône-Alpes (bourse Eurodoc de F. Abbaci).

Références

[BUCK 1995] Buckley C., Salton G., Allen J., Singhal A. : Automatic Query Expansion using SMART. Proceedings of TREC-3, 1995, pp. 69-80.

[BUCK 1996] Buckley C., Singhal A., Mitra M., Salton G. : New Retrieval Approaches using SMART. Proceedings of TREC-4, 1996, pp. 25-48.

[CALL 1995] Callan J. P., Lu Z., Croft W. B. : Searching Distributed Collections with Inference Networks. Proceedings of the ACM-SIGIR'95, 1995, pp. 21-28.

[DUMA 1994] Dumais S. T. : Latent Semantic Indexing (LSI) and TREC-2. Proceedings of TREC-2, 1994, pp. 105-115.

[GORD 1999] Gordon M., Pathak P. : Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines. Information Processing & Management, Vol. 35, 1999, pp. 141-180.

[GRAV 1997] Gravano L., Chang K., García-Molina H., Lagoze C., Paepcke A. : STARTS - Stanford Protocol Proposal for Internet Retrieval and Search. Computer Systems Laboratory, Stanford University, Stanford (CA). (available at http://www-db.stanford.edu/~gravano/starts_home.html).

[HAWK 1999] Hawking D., Thistlewaite P. : Methods for Information Server Selection. ACM Transactions on Information Systems, Vol. 17, 1999, pp. 40-76.

[KAHL 1993] Kahle B., Morris H., Goldman J., Erickson T., Curran J. : Interfaces for Distributed Systems of Information Servers. Journal of the American Society for Information Science, Vol. 44, 1993, pp. 453-485.

[KWOK 1995] Kwok K. L., Grunfeld L., Lewis D. D. : TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. Proceedings of TREC-3, 1995, pp. 247-255.

[LARK 2000] Larkey L. S., Connell M. E., Callan J. : Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. Proceedings of CIKM'2000, 2000, pp. 282-289.

[LAWR 1999] Lawrence S., Giles C. L. : Accessibility of Information on the Web. Nature, Vol. 400, 1999, pp. 107-110.

[ROBE 1995] Robertson S. E., Walker S., Hancock-Beaulieu M. M. : Large Test Collection Experiments on an Operational, Interactive System: Okapi at TREC. Information Processing & Management, Vol. 31, 1995, pp. 345-360.

[SALT 1989] Salton G. : Automatic Text Processing. Addison-Wesley, Reading (MA), 1989.

[SALT 1983] Salton G., McGill M. J. : Introduction to Modern Information Retrieval. McGraw-Hill, New York (NY), 1983.

[SCHW 1998] Schwartz C. : Web Search Engines. Journal of the American Society for Information Science, Vol. 49, 1998, pp. 973-982.

[SELB 1999] Selberg E.W : Towards Comprehensive Web Search. Ph.D. Thesis, University of Washington, 1999.

[SING 1999] Singhal A., Choi J., Hindle D., Lewis-D. D., Pereira F. : AT&T at TREC-8. Proceedings of TREC-8, 1999, pp. 239-251.

[VOOR 1995] Voorhees E. M., Gupta N. K., Johnson-Laird B. : Learning Collection Fusion Strategies. Proceedings of the ACM-SIGIR'95, 1995, pp. 172-179.

[XU 1998] Xu J., Callan J. P. : Effective Retrieval with Distributed Collections. Proceedings of the ACM-SIGIR'98, 1998, pp. 112-120.

Annexe 1. Exemples de requêtes

<num> Number: 428
<title> declining birth rates
<desc> Description: Do any countries other than the U.S. and China have a declining birth rate?
<narr> Narrative: To be relevant, a document will name a country other than the U.S. or China in which the birth rate fell from the rate of the previous year. The decline need not have occurred in more than the one preceding year.

<num> Number: 453
<title> hunger
<desc> Description: Find documents that discuss organizations/groups that are aiding in the eradication of the worldwidehunger problem.
<narr> Narrative: Relevant documents contain the name of any organization or group that is attempting to relieve the hunger problem in the world. Documents that address the problem only without providing names of organizations / groups that are working on hunger are irrelevant.

Annexe 2. Statistiques sur les deux collections-tests

Collection	Taille (en MB)	nb documents	nb de formes	nb requêtes pert.	nb requêtes rép.
FT	564	210 158	375 499	49	50
FR	395	55 630	196 220	18	50
FBIS	470	130 471	502 099	43	50
LA Times	475	131 896	337 492	45	50
TREC8	1 904	528 155	1 008 463	50	50

Table A.1 : Quelques statistiques sur les collections TREC8

Collection	Taille (en MB)	nb documents	nb de formes	nb requêtes pert.	nb requêtes rép.
TREC9.1	1 325	207 485	1 800 230	28	48
TREC9.2	1 474	207 429	2 274 318	44	48
TREC9.3	1 438	221 916	1 783 691	38	48
TREC9.4	1 316	202 049	1 684 452	21	48
TREC9.5	1 309	203 073	1 988 627	22	48
TREC9.6	1 311	215 451	2 136 650	24	48
TREC9.7	1 336	200 146	2 223 065	25	47
TREC9.8	1 524	234 547	2 134 040	43	48
TREC9	11 033	1 692 096		50	48

Table A.2 : Quelques statistiques sur les collections TREC9

Le tableau ci-dessus donne :

1. la taille de chaque collection;
2. le nombre de documents contenus dans chaque collection;
3. le nombre de formes distinctes dans chaque collection;
4. le nombre de requêtes pour lesquelles la collection possède au moins un document pertinent;
5. le nombre de requêtes pour lesquelles la collection retourne une réponse comportant au moins un document. Pour le corpus TREC9, les deux mêmes requêtes n'arrivent pas à extraire le moindre document des collections (requête 464 «nativityscenes» et 487 «angioplast7»). En effet, ces deux requêtes possèdent une faute d'orthographe.

Annexe 3. Formules d'indexation

Afin d'attribuer un poids w_{ij} reflétant l'importance de chaque terme d'indexation t_j, j = 1, 2, ..., t, dans un document D_i, nous pouvons recourir à l'une des formules décrites dans la table ci-dessous. Dans cette dernière, tf_{ij} indique la fréquence d'occurrence du terme t_j dans le document D_i (ou dans la requête), n représente le nombre de documents D_i dans la collection, df_j le nombre de documents dans lesquels le terme t_j apparaît (fréquence documentaire), et idf_j l'inverse de la fréquence documentaire ($idf_j = \ln[n / df_j]$). Les constantes ont été fixées aux valeurs suivantes : slope = 0,2, pivot = 150, b = 0,75, k = 2, k_1 = 1,2, advl = 900. De plus, la longueur du document D_i (ou le nombre de termes d'indexation associé à ce document) est notée par nt_i, la somme de valeurs tf_{ij} par l_i et $K = k \cdot [(1 - b) + b \cdot (l_i / advl)]$.

bnn	$w_{ij} = 1$	nnn	$w_{ij} = tf_{ij}$
dtn	$w_{ij} = [\ln(\ln(tf_{ij})+1)+1] \cdot idf_j$	atn	$w_{ij} = idf_j \cdot [0{,}5 + 0{,}5 \cdot tf_{ij} / \max tf_{i.}]$
Okapi	$w_{ij} = \dfrac{(k_1+1) \cdot tf_{ij}}{K + tf_{ij}}$	npn	$w_{ij} = tf_{ij} \cdot \ln\left[\dfrac{n - df_j}{df_j}\right]$
lnc	$w_{ij} = \dfrac{(\ln(tf_{ij})+1)}{\sqrt{\sum_{k=1}^{t}((\ln(tf_{ik})+1))^2}}$	ntc	$w_{ij} = \dfrac{tf_{ij} \cdot idf_j}{\sqrt{\sum_{k=1}^{t}(tf_{ik} \cdot idf_k)^2}}$
ltc	$w_{ij} = \dfrac{(\ln(tf_{ij})+1) idf_j}{\sqrt{\sum_{k=1}^{t}((\ln(tf_{ik})+1) \cdot idf_k)^2}}$		
dnu	$w_{ij} = \dfrac{\left(\dfrac{1+\ln(1+\ln(tf_{ij}))}{1+pivot}\right)}{(1-slope) \cdot pivot + slope \cdot nt_i}$		
Lnu	$w_{ij} = \dfrac{\left(\dfrac{1+\ln(tf_{ij})}{1+pivot}\right)}{(1-slope) \cdot pivot + slope \cdot nt_i}$		

Table A.3 : Formules de pondération lors de l'indexation

Approaches to Collection Selection and Results Merging for Distributed Information Retrieval

Yves Rasolofo
Institut interfacultaire d'informatique,
Université de Neuchâtel
Pierre à Mazel, 7
2000 Neuchâtel, Switzerland
Yves.Rasolofo@unine.ch

Faïza Abbaci
Ecole nationale supérieure des Mines
de Saint-Étienne
158 Cours Fauriel
42023 Saint-Étienne, France
Abbaci@emse.fr

Jacques Savoy
Institut interfacultaire d'informatique,
Université de Neuchâtel
Pierre à Mazel, 7
2000 Neuchâtel, Switzerland
Jacques.Savoy@unine.ch

ABSTRACT

We have investigated two major issues in Distributed Information Retrieval (DIR), namely: collection selection and search results merging. While most published works on these two issues are based on pre-stored metadata, the approaches described in this paper involve extracting the required information at the time the query is processed. In order to predict the relevance of collections to a given query, we analyse a limited number of full documents (e.g., the top five documents) retrieved from each collection and then consider term proximity within them. On the other hand, our merging technique is rather simple since input only requires document scores and lengths of results lists. Our experiments evaluate the retrieval effectiveness of these approaches and compare them with centralised indexing and various other DIR techniques (e.g., CORI [2][3][23]).

We conducted our experiments using two testbeds: one containing news articles extracted from four different sources (2 GB) and another containing 10 GB of Web pages. Our evaluations demonstrate that the retrieval effectiveness of our simple approaches is worth considering.

Keywords

Distributed information retrieval, collection selection, search results merging, evaluation.

1. INTRODUCTION

Conventional retrieval systems based on a single centralised index are subject to several limitations [11]. Among these is the very critical limitation due to the exponential growth in available information on the Internet. Even though search engines have currently increased their coverage of the content of the Web, They are still a long way from full coverage of the entire Web. Moreover, other limitations inherent to centralised approach might surface such as insufficient bandwidth server overloading and failures. Given these facts, it thus seems more appropriate to turn to the DIR approach for storage and search processing.

A simple DIR system is made up of collection servers and a broker. Typically, a user submits a request to the broker, which then forwards the query to a carefully selected subset of collection servers likely to contain relevant documents to the query (e.g., based on query terms, query language or a subset of servers pre-selected by the user). Each selected collection server processes the query and returns a ranked list to the broker. Finally, the broker merges the results lists received into a single list and forwards it to the user.

Some recent studies have reported that retrieval effectiveness in DIR systems may potentially be as effective as a single centralised IR system [17][24]. In our research, we have investigated how to select a subset of collection servers which are most likely to be relevant to a given query, and then how to merge the results lists in order to obtain improved retrieval effectiveness.

In this vein, the collection selection system we propose is different in two main aspects from various other selection methods. Firstly, our approach does not use any pre-stored metadata to predict the collection's relevance to the query. Secondly, it does not require collection ranking: each collection is selected independently from the others. On the other hand, our merging technique involves a rather simple recalculation of each document score.

We carried out our evaluations using a first testbed made up of news articles (about 2 GB from TREC8) and a second testbed containing pages extracted from the Web (more than 10 GB from TREC9).

In Sections 2 and 3, we will describe some well-known techniques for collection selection and results merging, and then describe our approaches in more details. Section 4 will discuss the details of evaluations we carried out on two testbeds (TREC8 and TREC9), and comparisons between performances of our strategies with those of other approaches. The final section will comment on experimental results and on our work in progress.

2. COLLECTION SELECTION

Collection selection refers to automatic or manual selection of a subset of collections (or servers) most likely to contain relevant documents for a given query. Obviously, ignoring this step and

sending the query to all known collections is one possible solution, but this method is proving to be very expensive in terms of resources, and it can increase user latency. Thus, the goal of collection selection is to reduce the number of collections searched as much as possible, without decreasing retrieval effectiveness [9][11].

2.1 Related Works

In order to select automatically a subset of servers, most collection selection techniques compute a score for each of the collections, based on their usefulness to the submitted query. The collections are then ranked according to these scores, thus the system could select either the N highest ranking collections or collections with scores exceeding a certain threshold. This type of selection requires the global collection of information in order to calculate the collection scores. Thus, the approaches differ in the nature of this information or in the manner in which it is acquired. Previous works consider collection descriptions [4] or collection statistics (frequency, co-occurrence, etc.) [10][16][25] in order to perform collection selection, but these techniques require collection cooperation.

Xu et al. [24] suggested that documents could be gathered according to their topics, and a language model associated with each topic. Callan et al. [2][3][23] have presented a Collection Retrieval Inference network (CORI), which considers each collection as a single gigantic document. Ranking the collections is similar to document ranking methods used in conventional information retrieval system. Several methods were developed by Zobel [25] to calculate collection scores. Moffat et al. [16] suggested to decompose each collection into blocks of documents, with the blocks being indexed by the broker. This index is then used to find those blocks having high-ranking scores with respect to the query, and the collections matching these blocks are then selected. The GlOSS system [10] ranks collections according to their goodness for the submitted query, and to do so it estimates the number of documents in each collection having similarities to the query greater than a predefined threshold. It then sums these similarities in order to obtain the collection score. At query time, Hawking et al. [11] proposed to broadcast a probe query (containing one to three terms) to all available collections, each of which responds with term information that is used to calculate collection score. Towell et al. [21] developed a learning based method for reducing search costs, whereby they determined the optimum number of documents to be retrieved from each collection rather than defining the number of collections to be searched. In their calculation of collection scores Craswell et al. [7] included the search engine's retrieval effectiveness for each collection.

We believe CORI to be a good representative of the above strategies, we will describe its selection procedure in more details and evaluate it in our experiments. This approach uses an inference network to rank collections. For the *i*th collection and for a given query Q, the collection score is computed as:

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

Where $s(t_j \mid C_i)$ indicates the contribution of the search term t_j in the score of collection C_i calculated as follows:

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

where:

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

m is the number of terms included in Q,

$|C|$ is the number of collections,

df_i is the number of documents in collection C_i containing the *j*th query term,

cf_j is the number of collections containing the query term t_j,

lc_i is the number of indexing terms in C_i,

avlc is the average number of indexing terms in each collection, and

defB, *b* and *k* are constants and, as suggested by Xu and Callan [23], were set at the following values: *defB* = 0.4, *k* = 200 and *b* = 0.75.

After ranking the collections according to their scores, one possibility is to select the N top ranked collections, where N is determined by the user. Another possibility is to use an algorithm to cluster the collection scores and the collections in the top clusters are then selected [2]. We have evaluated the latter case using a cluster difference threshold α=0.0002.

2.2 Our Selection Procedure

We have denoted our selection approach as TRD-CS, for 'using Top Ranked Documents for Collection Selection' and it differs from previous ones in that it does not assign scores to each collection. It bears a slight resemblance to the approach developed by Hawking et al. [11], both approaches assuming that no information is available *a priori* to perform the selection. The information needed is derived while processing the query.

In our approach, the broker broadcasts the query to all available collections ($|C|$ collections), with each collection returning *nb_doc* highly ranked documents to the broker. The broker then calculates the score for each document received (*nb_doc* * $|C|$), and sorts them according to their scores, with the collections matching the *n_first* documents being selected.

In order to compute document scores, we assume that the following are good relevance indicators: the number of query terms included in each document surrogate, the distance between query terms and their frequencies. From this perspective and inspired by [14], we calculate the document score as follows:

$$score(D,Q) = (c_1 \cdot nb_q) + (c_2 \cdot dis\_ind(D,Q)) + \frac{nb\_occ}{c_3}$$

where for each document D:

nb_q is the number of query terms present in D,

nb_occ is the total number of occurrences of query terms in D,

c_1, c_2, c_3 are constants and set to c_1=100, c_2 =1000, c_3=1000 in our experiments,

dis_ind is the indicator of distance between query terms in *D*. This function returns a real value greater or equal to zero.

According to the formula introduced by Clarke et al. [5] and assuming the two first query terms are the most important search keywords, we compute *dis_ind* for these two terms as follows.

$$dis\_ind(D,Q) = \sum_i dis(k,l)_i$$

where:

k and *l* are query term positions within the document *D* delimiting the *i*th block,

$dis(k,l)_i$ is the score for this block in the document *D*, which satisfies the query *Q* (i.e. the block contains the first two query terms in our case), and which does not include any other block satisfying the query (the block having the smallest size is selected). For example, we consider a given query consisting of two terms t_i and t_j. If t_i appears in the 5th and 25th positions and t_j in the 27th position, we may find the first block (*k*=5 and *l*=27) and the second block (*k*=25 and *l*=27). As the first block contains the second, this first block is ignored and *dist_ind* is therefore reduced to *dist_ind(D,Q)* = *dis(25, 27)=0.5*.

More formally:

$$dis(k,l)_i = \begin{cases} \frac{1}{|(k,l)_i|}, & if\ |(k,l)_i| > 1 \\ 1, & if\ |(k,l)_i| \le 1 \end{cases}$$

In the case of a mono-term query, and according to [14], *dis_ind* represents the inverse of the distance from the start of the document to the first occurrence of this search term. Finally, the document score is assigned a zero value if it does not contain any query terms.

3. RESULTS MERGING STRATEGIES

After defining the set of selected collections, results lists returned from collections must be combined into a final single ranked list. To resolve this problem, various merging strategies have been suggested.

3.1 Related Works

One of the best-known approaches assumes that each collection contains approximately the same number of relevant items and that they are equally distributed within results lists taken from the collections [22]. Under this assumption, we can set up a final result list in a round robin manner, a merging strategy used by earlier meta-search engines on the Web [19]. Our previous evaluation [20] demonstrated that this method did not perform well and thus is not included in this paper for further evaluation.

When collections are indexed by the same search model, we may assume that scores attributed to documents are comparable across collections [12]. The document scores are then used to merge the documents from collections into a single list. This strategy is called "Raw Score Merging" (RSM). However, Dumais [8] mentioned that various statistics may be collection dependant (e.g., the *idf* value used to weight documents and/or queries) and these values may vary widely across collections. Therefore, this phenomenon may invalidate the raw score merging hypothesis.

One variant of the RSM strategy is to assume that absolute document scores would not be compared. We could normalise the document scores based on the maximum document score for each collection.

The CORI [2][3][13][23] approach proposes a fourth merging strategy, and we will compare its performance with that of our merging approach. As shown previously, CORI computes a score for each collection (Section 2.1). Based on this collection score denoted as s_i, the weight w_i for *i*th collection is:

$$w_i = 1 + |C| \cdot [(s_i - \bar{s})/\bar{s}]$$

where:

s_i is the collection's score of the *i*th collection,

$\bar{s}$ is the mean of collection scores, and

$|C|$ is the number of collections as defined above.

The resulting weight w_i will be used to modify the score attached to each document. Instead of using document scores directly (as in RSM), each document score is multiplied by the value w_i of the corresponding collection and the broker merges the results lists according to these new scores.

3.2 Our Merging Strategy

Our merging strategy is denoted LMS for 'using result Length to calculate Merging Score', and it calculates a score for each collection. The underlying idea is to use weights in order to increase document scores from collections having scores greater than the average score, and to decrease those from any collections having scores less than the average score. Our approach has the advantage of being simple, since as input it only uses document scores and result lengths. Since collection statistics are not required, a broker using our approach does not need to store collection information. By contrast, when collections statistics are required within a dynamic environment such as the Web, they need to be updated frequently, and this is not possible without establishing some cooperation between the broker and collection servers. Thus, our approach is more practical.

Our merging strategy consists of calculating a collection score according to the proportion of documents retrieved (result length) by each collection. This score is based on our intuition that a collection would contain more relevant documents for a given query, if its collection server found more documents. The score for the *i*th collection is determined by:

$$s_i = \log\left(1 + \frac{l_i \cdot K}{\sum_{j=1}^{|C|} l_j}\right)$$

where:

K is a constant (set to 600 in our evaluations),

l_i is the number of documents retrieved by the *i*th collection, and

$|C|$ is the number of collections.

Our model uses a constant K in order to normalize the collection score as well as the natural logarithm, an order-preserving transformation used in similar contexts [15]. Based on this collection score, our merging algorithm calculates the collection weight denoted w_i for the ith collection as follows:

$$w_i = 1 + [(s_i - \bar{s})/\bar{s}]$$

where:

s_i is the ith collection score, and

$\bar{s}$ is the mean collection score.

As in the CORI approach, the final document score is the product of w_i and the document score as computed by the server for this document.

4. EXPERIMENTS

4.1 The Testbeds

Our first testbed consisted of documents written in English and used for the eighth TREC conference, called TREC8. It contains 528,155 documents (about 1,904 MB) extracted from four sources: *Financial Times* (FT, 210,158 documents), *Federal Register* (FR, 55,630 documents), *Foreign Broadcast Information Service* (FBIS, 130,471 documents) and *Los Angeles Times* (LA Times, 131,896 documents).

Collection	Size (MB)	# docs	# Topic-Rel.	# Topic-Ret.
FT	564	210,158	49	50
FR	395	55,630	18	50
FBIS	470	130,471	43	50
LA Times	475	131,896	45	50
TREC8	1904	528,155	50	50

Table 1. TREC8 statistics

An assessed set of 50 topics was provided, covering a rather broad range of subjects, including for example "Estonia, economy," "suicides," "airport security," "osteoporosis" and "cosmic events." We used only topic titles having two words on average (standard deviation: 0.96) in order to simulate typical queries sent by search engine users. We noted that these query words are ambiguous and occurres frequently within the documents.

The second testbed is the test collection used for the Web track of TREC9 conference, which differs from TREC8 in that it contains only Web pages from sites around the world. (1,692,096 Web pages with a total size of 11,033 MB). Thus, it is roughly six times greater than TREC8 and its contents are quite different, including a larger number of spelling mistakes. As with the first testbed, we used only topic titles based on real-life queries sent by users to the Excite search engine. They originated from various fields (e.g., "Parkinson's disease," "hunger," "Baltimore," "how e-mail benefits businesses" and "Mexican food culture"). Query terms are also ambiguous, and their average length is 2.4 words (standard deviation of 0.6).

We decided to split the TREC8 testbed into four collections according to their sources (FT, FR, FBIS, LA Times), with the number of documents and the collection size varying from one collection to another (see Table 1). This configuration more closely reflects a digital library environment, made up of several information sources (or servers) and using the same search model. We divided the TREC9 testbed into eight collections, each having roughly the same number of documents and the same size (see Table 2).

Collection	Size (MB)	# docs	# Topic-Rel.	# Topic-Ret.
TREC9.1	1,325	207,485	28	48
TREC9.2	1,474	207,429	44	48
TREC9.3	1,438	221,916	38	48
TREC9.4	1,316	202,049	21	48
TREC9.5	1,309	203,073	22	48
TREC9.6	1,311	215,451	24	48
TREC9.7	1,336	200,146	25	47
TREC9.8	1,524	234,547	43	48
TREC9	11,033	1,692,096	50	48

Table 2. TREC9 statistics

Tables 1 and 2 contain various statistics including for each collection the size, number of documents, number of topics having at least one relevant item, and number of topics having at least one document returned. For some topics there were no relevant documents returned by the collections. For example in TREC9, query topics #464 and #487 did not return any documents from any of the eight collections, due to spelling errors (topic #464: "nativityscenes" and topic #487: "angioplast7").

Merging	Single (baseline)	RSM		CORI		LMS	
Selection	Av. Prec.	Av. Prec.	Diff.	Av. Prec.	Diff	Av. Prec.	Diff
TREC8-NS	0.2566	0.2397	-6.59 %	0.2416	-5.85 %	0.2462	-4.05 %
TREC8-CORI	0.2566	0.2005	-21.86 %	0.2033	-20.77 %	0.1997	-22.17 %
TREC8-TRD-CS	0.2566	0.2440	-4.91 %	0.2453	-4.40 %	0.2462	-4.05 %
TREC8-OPT	0.2566	0.2543	-0.90 %	0.2533	-1.29 %	0.2480	-3.35 %
TREC9-NS	0.1986	0.1832	-7.75 %	0.1847	-7.00 %	0.1932	-2.72 %
TREC9-CORI	0.1986	0.1862	-6.24 %	0.1893	-4.68 %	0.1922	-3.22 %
TREC9-TRD-CS	0.1986	0.1828	-7.96 %	0.1867	-5.99 %	0.1944	-2.11 %
TREC9-OPT	0.1986	0.2097	5.59 %	0.2142	7.85 %	0.2144	7.96 %

Table 3. Average precision achieved by various selection and merging strategies

Merging / Selection	RSM vs. CORI	RSM vs. LMS	CORI vs. LMS
TREC8-NS	RSM < CORI	RSM < LMS	CORI < LMS
TREC8-CORI	RSM = CORI	RSM = LMS	CORI = LMS
TREC8-TRD-CS	RSM = CORI	RSM = LMS	CORI = LMS
TREC8-OPT	RSM = CORI	RSM = LMS	CORI = LMS
TREC9-NS	RSM < CORI	RSM < LMS	CORI < LMS
TREC9-CORI	RSM = CORI	RSM < LMS	CORI < LMS
TREC9-TRD-CS	RSM < CORI	RSM < LMS	CORI < LMS
TREC9-OPT	RSM = CORI	RSM = LMS	CORI = LMS

Table 4. Results of Sign tests for various selection and merging strategies

Selection/Merging / Collection	NS/RSM vs. CORI/CORI	NS/RSM vs. TRD-CS/LMS	CORI/CORI vs. TRD-CS/LMS
TREC8	NS/RSM = CORI/CORI	NS/RSM < TRD-CS/LMS	CORI/CORI < TRD-CS/LMS
TREC9	NS/RSM = CORI/CORI	NS/RSM < TRD-CS/LMS	CORI/CORI < TRD-CS/LMS

Table 5. Results of Sign tests for NS/RSM vs. CORI/CORI vs. TRD-CS/LMS

All collections were indexed by the SMART system [1], using the OKAPI [18] probabilistic search model (Appendix 1).

4.2 Our Baselines

In the evaluations below, we will refer to a number of baselines, defined as follows:

- the optimal collection selection: given knowledge about the complete set of relevant documents supplied with the TREC data, we select collections that have at least one relevant document (labelled 'TREC8-OPT' and 'TREC9-OPT' in our tables);
- the centralised approach: for each of the testbeds, all documents are located in a single database (labelled 'Single' in our tables);
- no selection (NS): all collections are searched (we do not apply any selection procedure in evaluations labelled 'TREC8-NS' and 'TREC9-NS').

4.3 Evaluation

Given the three collection selection approaches, namely TRD-CS, CORI using collection scores clustering, and optimal selection (OPT), and the three merging strategies LMS, CORI and RSM, our experiments combined each collection selection approach with each merging strategy. The results reported below were obtained by using the following parameter values:

Our selection approach (TRD-CS) used $nb\_doc = 5$ (the number of top documents returned by each collection to be inspected); $n\_first = 11$ for TREC8 testbeds and $n\_first = 22$ for TREC9 testbed.

We used two different methodologies to evaluate the various approaches. Firstly for a given testbed and a combination of a selection approach with a merging approach, we used the TREC_EVAL program to compute average precision and precisions achieved after retrieving 5, 10, 15, 20, 30, 100, 200, 500 and 1000 documents. Table 3 shows the average precision values achieved by various selection/merging combinations and compares them with those of the centralised approach (column labelled 'Single'). Appendix 2 (Figures 1 to 6) depicts differences between precisions achieved by different combinations, according to the number of retrieved documents by means of curves.

Secondly, in order to decide whether a retrieval strategy was statistically better than another was, we used the Sign test [6], with a significance level $\alpha = 0.05$. The decisions based on this test are reported in Tables 4 and 5.

4.4 Discussion

In Table 3, the baseline (labelled 'Single') represents the average precision achieved by the centralised approach. The optimal selection ('TREC8-OPT' and 'TREC9-OPT') can be viewed as an ideal selection[1]. This optimal selection procedure produced the best retrieval effectiveness, no matter which merging strategy was used. With the TREC9 corpus and for three merging strategies, this scheme even improved the retrieval effectiveness over the baseline. The second best selection procedure seems to be our approach ('TREC8-TRD-CS' and 'TREC9-TRD-CS'). Ignoring the selection procedure ('TREC8-NS' and 'TREC9-NS') decreases the retrieval effectiveness from 6% to 8% over the centralised approach for RSM and CORI merging strategies. However, this degradation is lower when using our merging strategy LMS (4% for TREC8 and 3% for TREC9). Finally when combining our selection procedure (TDR-CS) with our merging strategy (LMS), it is evident that the achieved retrieval effectiveness is very close to that of the centralised approach for TREC9 corpus (-2.11%) and very satisfactory for TREC8 corpus (-4.05%).

[1] We performed such an ideal selection by using relevance judgement information. For a given query, a collection is selected only if it contains at least one relevant document.

In order to analyse the relative significance of differences found in Table 3, we applied the Sign tests. The results shown in Table 4 indicate that raw score merging (RSM) attains a retrieval performance that could be considered either equal or less effective than either of the CORI and our suggested LMS merging strategies. Also worth noting is that the last column indicates that our merging strategy (LMS) produces better or at least equal retrieval effectiveness compared to that of CORI.

Finally, it seems more appropriate to directly compare various combinations of good selection and good merging strategies. In this vein, we also compared NS/RSM, CORI/CORI and TRD-CS/LMS. The results as reported in Table 5 indicate that our solution outperforms both that of the CORI/CORI and NS/RSM strategies.

5. CONCLUSION

This paper describes effective selection collection and results merging strategies useful in DIR. Our approaches do not require the creation of any pre-stored metadata, and as such, do not need any up-dates to reflect changes in collection content. Moreover, our experiments were conducted using very short queries, similar to those submitted to search engines and therefore viewed as "Web realistic." Our evaluations show that:

- a combination of our two strategies works better than other collection-selection/results merging combinations,
- our selection method works well even with RSM or CORI merging strategies, and our merging approach works also well when no selection is performed,
- however, our selection strategy requires more transfer traffic for the downloading of the first *nb_doc* (*nb_doc*=5 in our evaluations) documents per collection. Thus, response delay may increase slightly.

The investigation described in this paper used the same search engine on several collections, a context that corresponds to a digital library environment where all sources are managed by the same search engine. Our current work will also consider the use of several collections indexed and searched by different search engines, without requiring a learning phase as described in [15].

6. ACKNOWLEDGEMENTS

This material is based on work supported in part by SNSF (Swiss National Science Foundation, under grant #21-58 813.99, J. Savoy and Y. Rasolofo) and by Région Rhône-Alpes (Eurodoc grant from F. Abbaci).

7. REFERENCES

[1] Buckley C.: Implementation of SMART Information Retrieval System. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, May 1985.

[2] Callan J. P., Lu Z., Croft W. B.: Searching Distributed Collections with Inference Networks. Proceedings of the ACM-SIGIR'95, 1995, pp. 21-28.

[3] Callan J.: Distributed Information Retrieval. In W. B. Croft (Ed.), Advances in Information Retrieval. Kluwer Academic Publishers, 2000 pp. 127-150.

[4] Chakravarthy A. S., Haase K. B.: NetSerf: Using Semantic Knowledge to Find Internet Information Archives. Proceeding of the ACM-SIGIR'95, 1995, pp. 4-11.

[5] Clarke C. L. A., Cormack G. V., Burkowski F. J.: Shortest Substring Ranking (MultiText Experiments for TREC-4). Proceedings of TREC-4, 1995, pp. 295-304.

[6] Conover W.J.: Practical Nonparametric Statistics (2nd ed.). John Wiley & Sons, 1980, pp. 122-129.

[7] Craswell N., Bailey P., Hawking D.: Server Selection in the World Wide Web. Proceedings of The Fifth ACM Conference on Digital Libraries, 2000, pp. 37-46.

[8] Dumais S. T.: Latent Semantic Indexing (LSI) and TREC-2. Proceedings of TREC-2, 1994, pp. 105-115.

[9] French J. C., Powell A.L., Callan J., Viles C. L., Emmitt T., Prey K. J, Mou Y.: Comparing the Performance of Database Selection Algorithms. Proceedings of ACM-SIGIR'99, 1999, pp. 238-245.

[10] Gravano L., Garcia-Molina H., Tomasic A.: GlOSS: Text-Source Discovery Over the Internet. ACM Transactions on Database Systems, 24(2), 1999, pp. 229-264.

[11] Hawking D., Thistlewaite P.: Methods for Information Server Selection. ACM Transactions on Information Systems, 17(1), 1999, pp. 40-76.

[12] Kwok K. L., Grunfeld L., Lewis D. D.: TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. Proceedings of TREC-3, 1995, pp. 247-255.

[13] Larkey L. S., Connell M. E., Callan J.: Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. Proceedings of CIKM'2000, 2000, pp. 282-289.

[14] Lawrence S., Giles C. L.: Inquirus, the NECI Meta Search Engine. Proceedings of The Seventh International World Wide Web Conference, 1998, pp. 95-105

[15] Le Calvé A., Savoy J. : Database Merging Strategy Based on Logistic Regression. Information Processing & Management, 36(3), 2000, pp. 341-359.

[16] Moffat A. , Zobel J.: Information Retrieval Systems for Large Document Collections. Proceedings of TREC-3, 1995, pp. 85-94.

[17] Powel A. L., French J. C., Callan J., Connell M., Viles C. L.: The Impact of Database Selection on Distributed Searching. Proceedings of the ACM-SIGIR-2000, 2000, pp. 232-239.

[18] Robertson S. E., Walker, S., Beaulieu M.: Experimentation as a Way of Life: Okapi at TREC. Information Processing & Management, 36(1), 2000, pp. 95-108.

[19] Selberg E.W.: Towards Comprehensive Web Search. Ph.D. Thesis, University of Washington, 1999.

[20] Savoy J., Rasolofo Y.: Report on TREC-9 Experiment: Linked-based Retrieval and Distributed Collections. Proceedings of TREC9, 2000, to appear.

[21] Towell G., Voorhees E. M., Narendra K. G., Johnson-Laird B.: Learning Collection Fusion Strategies for Information

Retrieval. Proceedings of The Twelfth Annual Machine Learning Conference, 1995, pp. 540-548.

[22] Voorhees E. M., Gupta N. K., Johnson-Laird B.: Learning Collection Fusion Strategies. Proceedings of the ACM-SIGIR'95, 1995, pp. 172-179.

[23] Xu J., Callan J. P.: Effective Retrieval with Distributed Collections. Proceedings of the ACM-SIGIR'98, 1998, pp. 112-120.

[24] Xu J., Croft, W. B.: Cluster-based Language Models for Distributed Retrieval. Proceedings of ACM-SIGIR'99, 1999, pp. 254-261.

[25] Zobel J.: Collection Selection via Lexicon Inspection. Proceedings of The Second Australian Document Computing Symposium, 1997.

Appendix 1. Search Model Equation

The Okapi probabilistic model [18] calculates the weight of the term t within a document d as follows:

$$w_{td} = (k_1 + 1) \cdot \frac{tf_{td}}{K + tf_{td}}$$

where:

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

l_d is the document length,

$advl$ is the average of document length (set to 750),

b is a constant (set to 0.9),

k is a constant (set to 2),

k_1 is a constant (set to 1.2),

tf_{td} is the occurrence frequency of the term t in document d.

The following formula shows the weight given to the same term t within the query q:

$$w_{tq} = \frac{tf_{tq}}{k_3 + tf_{tq}} \cdot \log\left(\frac{n - df_t}{df_t} \right)$$

where:

tf_{tq} is the search term frequency,

df_t is the number of documents in the collection containing the term t,

n is the number of documents included to the collection.

k_3 is a constant (set to 1000).

Appendix 2. Precision of Various Selection and Merging Strategies Combinations

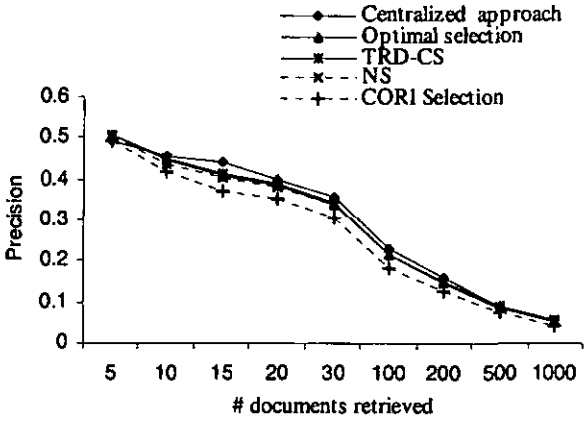

Figure 1. TREC8 testbed, LMS strategy.

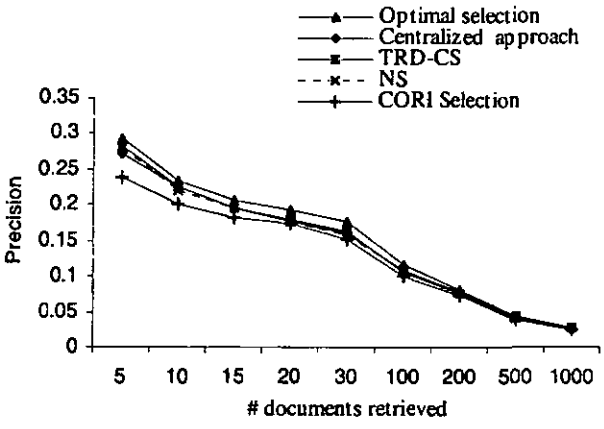

Figure 2. TREC9 testbed, LMS strategy.

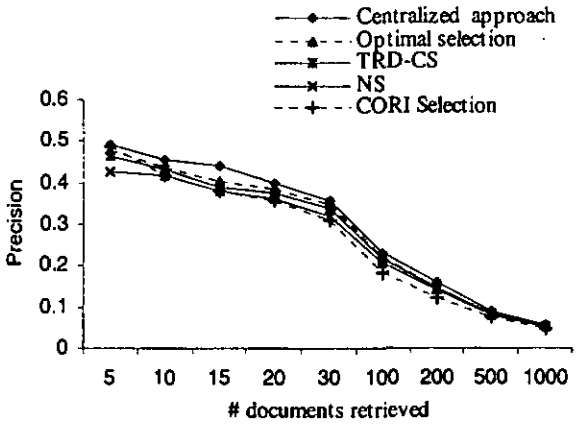

Figure 3. TREC8 testbed, Cori merging.

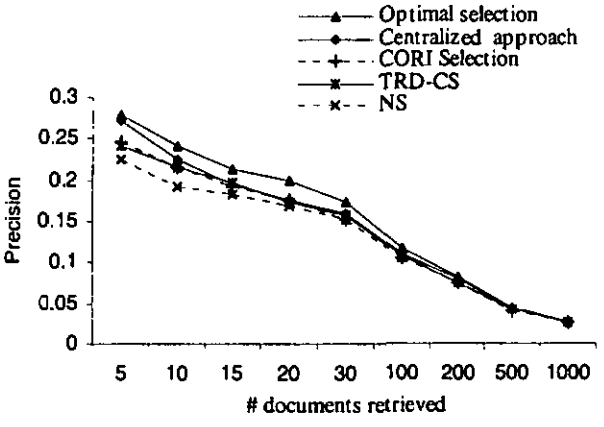

Figure 4. TREC9 testbed, Cori merging.

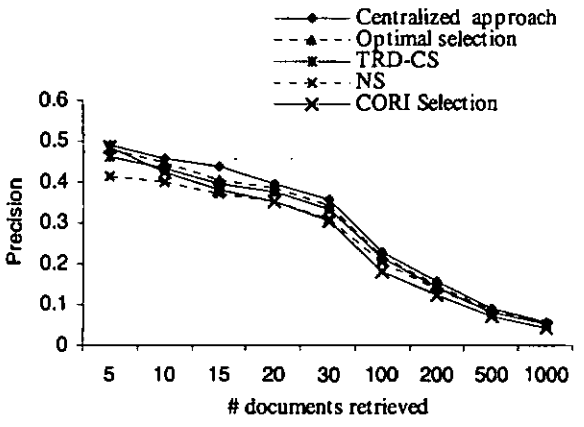

Figure 5. TREC8 testbed, RSM strategy.

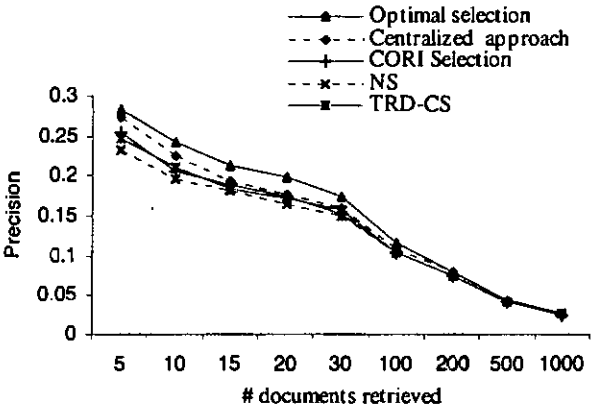

Figure 6. TREC9 testbed, RSM strategy.

Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique, Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch Web site: http://www.unine.ch/info/

Summary

For our participation in TREC-10, we will focus on the searching distributed collections and also on designing and implementing a new search strategy to find homepages. Presented in the first part of this paper is a new merging strategy based on retrieved list lengths, and in the second part a development of our approach to creating retrieval models able to combine both Web page and URL address information when searching online service locations.

Introduction

The Web of today represents a new paradigm, one that generates new challenges for the IR community. Included among these are: managing huge amounts of documents via distributed IR models, crawling through the Web in order to find appropriate Web sites to index, accessing documents written in various languages, measuring the quality or authority of available information, providing answers to very short user requests often expressed in ambiguous terms, satisfying a large range of search types (ad hoc, question-answering, location of online services, and interactive searches for specific document types or Web pages in order to satisfy a particular geographical or time constraint).

For our participation in TREC-10, we are focusing on two problems. One involves the presentation of a new merging strategy (collection fusion problem, Chapter 1) for the Web ad hoc track, and the other developing a search strategy intended to resolve homepage search problems (Chapter 2).

In order to evaluate our hypothesis when implementing the Okapi probabilistic model (Robertson *et al.*, 2000) we will use the SMART system as a test bed. This year our experiments are fully automated.

1. Distributed collections

In order to evaluate the retrieval effectiveness of various merging strategies, we formed four separate collections from the WT10g test collection (Savoy & Rasolofo, 2001). The same indexing scheme and retrieval procedure is used for each collection involved in this study. This type of distributed context more closely reflects digital libraries or search engines available on the Internet than do meta search engines, where different search engines may collaborate in response to a given user request (Selberg, 1999; Le Calvé & Savoy, 2000).

This chapter is organized as follows: Section 1.1 explains our indexing and search model. Section 1.2 describes related work on database merging strategies, while Section 1.3 presents our merging procedure. Finally, in Section 1.4 we evaluate our search model.

1.1. Indexing and retrieval scheme

From the original Web pages, we retained only the following logical sections: <TITLE>, <H1>, <CENTER>, <BIG>, with the most common tags <P> (together with </P>) being removed. Texts delimited by <DOCHDR>, </DOCHDR> tags were also removed. For longer requests, various insignificant keywords were removed (such as "Pertinent documents should include ..."). Moreover, search keywords appearing in topic title sections were assigned a term frequency of 3 (a feature that should have no impact on short requests).

For the ad hoc Web track, we conducted different experiments using the Okapi probabilistic model, in which the weight w_{ij} was assigned to a given term t_j in a document d_i and was computed according to the following formula:

$$w_{ij} = \frac{(k_1 + 1) \cdot tf_{ij}}{K + tf_{ij}} \quad (1)$$

$$\text{with } K = k_1 \cdot \left((1 - b) + b \cdot \frac{l_i}{avdl} \right) \quad (2)$$

where tf_{ij} indicates the within-document term frequency, and b, k_1 are parameters. K represents the ratio between the length of d_i measured by l_i (sum of tf_{ij}) and the document mean length is denoted by advl.

To index each search keyword t_j included in the request, the following formula was used:

$$w_{qj} = \frac{tf_{qj}}{k_3 + tf_{qj}} \cdot \ln\left(\frac{N - df_j}{df_j}\right) \quad (3)$$

where tf_{qj} indicates the search term frequency, df_j the collection-wide term frequency, N the number of documents in the collection, and k_3 is a parameter.

To adjust the underlying Okapi search model parameters, we used the values suggested by Walker *et al.* (1998): advl = 900, b = 0.75, k_1 = 1.2, and k_3 = 1000. We did however believe that it might be more effective to assign a lower value to the parameter b, and in order to verify this assumption. We also evaluated the Okapi model using b = 0.7 or b = 0.5, values, resulting in interesting retrieval performances for TREC-9 topics.

Finally, for the request q containing m search terms the retrieval status value (denoted RSV_i) of a Web page d_i was estimated as:

$$RSV(d_i, q) = RSV_i = \sum_{j=1}^{m} w_{ij} \cdot w_{qj} \quad (4)$$

In order to obtain a broader picture of our evaluations, we considered two different query formulations: (1) using only the Title section (T) or (2) all three logical sections (Title, Descriptive and Narrative, noted T-D-N). Finally, we should mention that these queries were "real topics" in the sense that they were taken from a MSNSearch log.

1.2. Previous work on merging strategies

Various solutions have been suggested for merging separate result lists obtained from distributed collections. As a first approach, and taking only the rank of the retrieved items into account, we might interleave results in a round-robin fashion. According to previous studies (Voorhees *et al.*, 1995; Callan *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001), such interleaving schemes have a retrieval effectiveness of around 20% to 40% below that achieved from single retrieval schemes, working with a single huge collection representing an entire set of documents.

In order to account for document scores computed for each retrieved item (or its retrieval status value), we might formulate the hypothesis that each collection is searched by the same or very similar search engines and that RSV values are therefore directly comparable (Voorhees *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001). Such a strategy, called raw-score merging, produces a final list sorted by the document score computed by each collection. However, as indicated by Dumais (1994), collection-dependent statistics contained in document or query weights may vary widely among collections, and therefore this phenomenon may invalidate the raw-score merging hypothesis.

To deal with this fact, we could normalize document scores within each collection through dividing them by the maximum score (i.e., the document score of the retrieved record found in the first position).

Callan *et al.* (1995) and Xu & Callan (1998) suggested a merging strategy called CORI, one that incorporates scores achieved by both collection and document. The collection score corresponds to the probability that the related collection would respond appropriately to the current request. In this scheme, each collection is viewed as a huge document and we might therefore use an IR scheme to rank the various collections according to the submitted request, since IR systems rank these documents according to their retrieval status values. In a second step, we simply multiply the document scores by the corresponding collection scores and then sort the result lists according to this value.

1.3. *Our merging strategy*

Our new merging strategy, denoted LMS for "using result Length to calculate Merging Score", and as does the CORI model, begins by estimating a score for each collection. The underlying idea is to use these weights to increase document scores from those collections having scores greater than the average score, and to decrease those for any collections having scores less than the average score. Our approach has the advantage of being simple, since it only uses document scores and result lengths as input. Also, since collection statistics are not required, systems using our approach do not need to store collection information. By contrast, when collections statistics are required within a dynamic environment such as the Web, they need to be updated frequently, and this is not possible without establishing some sort of cooperation between the main system and collection servers. Thus, our approach is more practical.

Our merging strategy consists of calculating a collection score according to the proportion of documents retrieved (result length) by each collection. This score is based on our intuition that a collection would contain more relevant documents for a given query if its collection server were to find more documents. The score for the *k*th collection is determined by:

$$s_k = \ln\left(1 + l_k \cdot K \Big/ \sum_{j=1}^{n} l_j\right)$$

where

- K is a constant (set to 600 in our evaluations),
- l_k is the number of documents retrieved by the *k*th collection, and
- |C| is the number of collections.

Our model uses a constant K in order to normalize the collection score as well as the natural logarithm, an order-preserving transformation used in similar contexts (Le Calvé & Savoy, 2000). Based on this collection score, our merging algorithm calculates the collection weight denoted w_k for the *k*th collection as follows:

$$w_k = 1 + [(s_k - s_m)/s_m]$$

where

- s_k is the *k*th collection score, and
- s_m is the mean collection score.

As in the CORI approach, the final document score is the product of w_k and the document score RSV_i computed by the server for this document. The value of this product is used as the key to sort the retrieved items in the final single result list.

1.4. Evaluation

To evaluate our propositions, we first used the TREC-9 topics (50 queries, 2,617 relevant documents) taken from the WT10g test collection. Average precision comparisons (computed by the TREC-EVAL system based on 1,000 retrieved items) achieved by the raw-score merging approach are depicted in the second column of Table 1a. On the other hand, average precision decreases with the use of round-robin merging strategy (third column of Table 1a). As one can see, considering result lengths in the merging process may marginally improve average precision over this baseline (last column of Table 1a).

After baviog considered different values for the parameter b, a smaller value (e.g., b = 0.5) seems to improve average precision (from 20.04 to 20.64, meaning an enhancement of +3% using raw-score merging or +2.6% when using our result length merging scheme (20.24 vs. 20.76)).

From studying the retrieval performance using TREC-10 topics (Table 1b), our previous findings were confirmed: the round-robin strategy results in lower average precision. From an analysis of parameter b, one can see that when setting b = 0.5, there is an improvement of +3.4% (from 16.59 to 17.16 in average precision). This fact is not however confirmed by longer requests, where the best value for the parameter b seems to be 0.7.

The data in Table 1b also indicates that by taking more search terms into account we can increase retrieval effectiveness substantially (around +30%). Finally, Table 3 provides a detailed analysis of our official runs applied to the Web ad hoc track, and Table 1b lists their retrieval performance in bold characters.

In order to analyze our merging strategy, we listed the number and the percentage of relevant items provided by each collection in Table 4. As one can see,

	Average precision (% change)		
Query (Title only) Model / merging strategy	TREC-9 50 queries Raw-score	TREC-9 50 queries Round-robin	TREC-9 50 queries Result lengths
Okapi (b=0.75)	20.04	17.66 (-11.9%)	20.24 (+0.9%)
Okapi (b=0.7)	20.34	17.96 (-11.7%)	20.60 (+1.3%)
Okapi (b=0.5)	20.64	17.66 (-14.4%)	20.76 (+0.6%)

Table 1a. Average precision of various retrieval schemes based on TREC-9 topics

	Average precision (% change)				
Query Model / merging	TREC-10 Title only Raw-score	TREC-10 Title only Round-robin	TREC-10 Title only Result lengths	TREC-10 Title-Desc-Narr Raw-score	TREC-10 Title-Desc-Narr Round-robin
Okapi (b=0.75)	**16.59**	16.43 (-1.0%)	**17.15** (+3.4%)	22.12 (+33.3%)	20.39 (+22.9%)
Okapi (b=0.7)	16.73	16.24 (-2.9%)	**16.99** (+1.6%)	22.42 (+34.0%)	20.76 (+24.1%)
Okapi (b=0.5)	17.16	16.03 (-6.6%)	17.50 (+2.0%)	21.68 (+26.3%)	19.87 (+15.8%)

Table 1b. Average precision of various retrieval schemes based on TREC-10 topics

Run name	Aver. pr.	Query	Parameter	Merging
UniNEtd	16.59	T	b = 0.75	raw-score merging
UniNEtdL	17.15	T	b = 0.75	result-length mergiog
UniNEt7dL	16.99	T	b = 0.7	result-length merging
UniNEn7d	22.42	T-D-N	b = 0.7	raw-score merging

Table 3. Description of official Web ad hoc run

the first collection (WT10g.1) contains a larger number of pertinent Web pages and the third collection (WT10g.3) a smaller number. From looking at the top 10, the top 100 and the first 1,000 retrieved pages, we can see how the percentage of pages extracted from each collection varies. More precisely, these numbers increase for the first and fourth collection and decrease for the other two.

Number of queries	50
Number of relevant doc.	3,363
Mean rel. doc. / request	67.26
Standard error	11.81
Median	39
Maximum	372 (q#: 541)
Minimum	2 (q#: 506, 538, 548)

Table 2. Relevance judgment statistics (TREC-10)

	Percentage of retrieved items			
	WT10g.1	WT10g.2	WT10g.3	WT10g.4
# rel. items	1007	800	640	916
% of rel.	29.94%	23.79%	19.03%	27.24%
Round-robin	25%	25%	25%	25%
Top 10	13.2%	28.8%	39.6%	18.4%
Top 100	19.54%	27.02%	30.62%	22.82%
Top 100	23.53%	25.16%	27.50%	23.82%

Table 4. Distribution of retrieved items (UniNEtd, raw-score merging, 50 topics)

2. Homepage searching

In the previous chapter, users sending a request to our search engine would obtain a ranked list of Web pages containing pertinent information about their information need. In this chapter, our objective is to design and implement a search strategy that would retrieve, at the limit, only one pertinent Web page that corresponds to the entrypage or to the online service location being sought by the user. For example, when users submit a request for "Quantas", they will retrieve the Quantas Airlines homepage, not several Web pages about this airline company.

To achieve this objective, we will first search URL addresses (Section 2.1). As a second search strategy, we will implement a combined retrieval model (Section 2.2) that searches the Web pages (Section 2.3) and then reranks the retrieved list by URL address length (Section 2.4). We will then examine any similarity between the query and the corresponding URL addresses (Section 2.5), and finally combine these three approaches (Section 2.6). An evaluation of our official runs is given in Section 2.7.

2.1. Searching the URL address only

For each of the 1,692,096 Web pages included in the WT10g test collection, we know the corresponding URL address. Thus as a first approach, we will build a text collection from these URLs and then obtain a mean number of distinct indexing terms (5.58 per URL address, max = 28, min = 1). From the available requests, we might then search this text database using the various IR models described using SMART notations (Savoy & Picard, 2001).

In this first attempt, we considered using a classical retrieval scheme to find the correct URL address (e.g., "www.cdsnet.net:80/vidiot/") when responding to the query "Vidiot". From examining usability studies, it was recommended that URL addresses contain information about the owner's name (usually the company name) and/or about content that might help users find their way around the Web or within the site (Nielsen, 2000, p. 246). If this principle is applied, our approach may work well.

	Simple queries		Extended queries	
IR model	MRR	# top 10	MRR	# top 10
Okapi	0.161	29	0.141	27
Lnu-ltc	0.077	15	0.091	17
atn-ntc	0.013	3	0.016	6
dtu-dtn	0.108	20	0.108	21
ltn-ntc	0.010	2	0.014	5
ntc-ntc	0.215	37	0.187	41
ltc-ltc	0.217	40	0.192	44
lnc-ltc	0.203	37	0.197	47
bnn-bnn	0.009	1	0.009	1
nnn-ntn	0.009	1	0.012	3
nnn-nnn	0.004	1	0.005	1
Mean	0.093	16.91	0.088	19.36

Table 5. Evaluation of URL searches (TREC-10, 145 topics)

In this first experiment, we evaluated eleven IR models, and as a retrieval performance measure, we calculated the mean reciprocal rank (MRR) over 145 topics (see Table 5, Column 2). As a second measure, we noted the number of topics for which a correct entrypage was found in the top 10 (see Table 5, Column 3). Overall, this search strategy does not work well for the problem of finding homepages. Moreover, although the Okapi probabilistic model provides the best search engine performance (Savoy & Picard, 2001), it is not best in terms of retrieval performance. For this particular retrieval task it seems that the vector-space model "doc=ltc, query=ltc" represents the best IR scheme, being able to retrieve 40 correct entrypages in the top 10 (over a total of 145, or 27.6% of the cases).

This rather limited performance thus reflects the fact that words found in an URL address are not necessarily those one would place in a query. For example, URLs often contain abbreviations (e.g., "www.iti.gov.sg" is the URL for "Information Technology Institute"). Moreover, if a given query contains very frequently occurring words (e.g., "of" "at", "in"), we add a second form of acronym that ignores these terms (e.g., from the request "Point of View Cafe", we form a first acronym as "povc" and a second as "pvc").

Concatenation represents another form of URL construction, with two (or more) words being joined (e.g., "www.dogzone.com" and "Dog Zone's dog clubs") or only the first (of the first two) letter(s) of a word are concatenated with the second word (e.g., "Digital Realms" expressed as "www.drealms.co.uk"). In order to deal with these various word formations, we designed our system such that it considers various URL construction possibilities. For example, for a simple request such as "Worldnet Africa", our system would construct the following expanded query: "Worldnet Africa wa worldnetafrica worldneta aworldnet woafrica worldnetaf".

Evaluating these extended request forms does not however result in appreciable performance enhancements, as can be seen in the last two columns in Table 5. Although the number of correct entrypages found in the top 10 seems to increase, the MRR measure indicates degradation. Thus, using only URL texts does not seem to be an adequate strategy, since establishing a link between query words and a URL addresses is a difficult task (e.g., based on the query "PiperINFO" which finds the URL "www.hamline.edu/" or "DaMOO" that finds the URL "lrc.csun.edu").

2.2. Guidelines for our combined search model

In order to develop a better search strategy, we decided to construct a two-stage retrieval strategy. In the first stage we used the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model used in the Web ad hoc track (see Section 1.1). Rather, we adapted the search model described in Section 2.3, and from the list of retrieved Web pages we were able to generate a corresponding list of URL addresses.

In the second stage of our retrieval strategy we inspected the corresponding URL addresses in order to verify whether or not they could be considered as appropriate URL candidates. To do so, we considered URL length, attributing more importance to short addresses (Section 2.4). Thus, in order to appear within the first positions in our final result list, a Web page would contain the search's keywords within its first 50 terms and its URL address would have to be short. As an alternative, we believe that URL address content should bear some similarity to the submitted request (Section 2.5), meaning we would again rank our retrieved list according to this similarity.

So far we have considered three types of retrieval expertise. The first retrieves and ranks Web sites according to page content, the second reranks these results according to URL address length and the last reranks the results according to URL address and submitted request similarity. In order to account for the results of these three approaches, we suggested reranking the retrieved items according to these three expert opinions (Section 2.6). Thus, with this search strategy, an entrypage will be found within the first positions if its Web page shares common words with the request, if its URL address is short and if this address contains some of the search keywords (or an abbreviation, a concatenation of two search keywords, or some URL construction as shown in Section 2.1).

2.3. Okapi model adaptation

In the first and most important stages of our combined retrieval strategy, we employed the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model as used in the Web ad hoc track (see Section 1.1). In fact, we added a precision device that considered only the first 50 words of each item retrieved and ranked only those documents having at least one query term within the first 50 words. This precision device was based on our intuition that document titles (or document headings) should provide an adequate indication as to whether or not corresponding Web pages are relevant to a given homepage search.

This feature works as follows. First, we form the set of all search keywords pairs. For example, from the

query $q = (t_i, t_j, t_k)$, we may deduce the following six terms pairs (t_i, t_j), (t_j, t_i), (t_i, t_k), (t_k, t_i), (t_j, t_k) and (t_k, t_j). We then inspect the document to see if the corresponding couple of search terms appears within a maximum distance of 5 (or with a maximum of four terms between the pair of search words). For example, supposing we are looking for the pair of search terms (t_j, t_k), then if we find a word sequence $(t_j, t_a, t_b, t_c, t_d, t_k)$, we search it for an occurrence of our pair of search terms within a maximum distance of 5. The weight assigned to the occurrence of this search keyword pair (t_j, t_k) in the document d_i is denoted as δw_{jk} and computed as follows:

$$\delta w_{jk} = 0.5 + 5 \Big/ \sqrt{\text{position}(t_k)}$$

where position(t_k) indicates the position (word number) of the term t_k from the beginning of the Web page. Thus, if the term t_k appears in the second position (in this case, the first position occupied by t_j), the function position(t_k) returns 1 and δw_{jk} is 5.5. On the other hand, if the distance is greater than 5, we ignore this occurrence.

It is possible however that a pair of search keywords (t_j, t_k) would appear more than once (within a maximum distance of 5) in the document d_i. To account for all occurrences of the search term pairs (t_j, t_k), we compute the following expression:

$$w_{i(j,k)} = \left(\frac{(k_1 + 1) \cdot \sum_{occ(j,k)} \delta w_{jk}}{K + \sum_{occ(j,k)} \delta w_{jk}} \right) \cdot \min(w_{qj}; w_{qk})$$

where $w_{i(j,k)}$ represents the weight attached to the search term phrase (t_j, t_k) in the document d_i, the parameter k_1 and K are evaluated as described in Equation 2 and w_{qj} and w_{qk} represent the weights of the search keyword t_j and t_k in the request (as shown in Formula 3).

In order to consider all occurrences of all search keywords pairs, we compute an additional weight $ph_{(d_i,q)}$ attached to the presence of these multiple search keywords pair occurrences in document d_i as follows:

$$ph_{(d_i,q)} = \sum_{all(j,k)} w_{i(j,k)}$$

The presence of search keyword pairs within the beginning of a given Web page d_i would change the value of its RSV_i, and this would be done simply by adding the phrase weight $ph_{(d_i,q)}$ to the previously computed RSV_i (see Equation 4), as follows:

$$RSV_i' = RSV_i + ph_{(d_i,q)} \qquad (5)$$

However, we suggest a variation that assigns a lower phrase weight $ph_{(d_i,q)}$ if the document d_i does not appear in the top ranked retrieved documents. Thus an alternative retrieval status value is assigned using the following formula:

$$RSV_i' = RSV_i + (1 - \alpha) \cdot ph_{(d_i,q)} \qquad (6)$$

$$\text{with } \alpha = \frac{RSV_{max} - RSV_i}{RSV_{max} - RSV_{min}}$$

where RSV_{max} and RSV_{min} are the RSV values assigned by the first and the last retrieved items respectively (computed according to Equation 4).

Run name	MRR	# in top 10	# not found
Okapi stem	0.261	65 (44.8%)	31 (21.4%)
Okapi nostem	0.274	72 (49.7%)	29 (20.0%)
Eq.5, stem	0.348	85 (58.6%)	26 (17.9%)
Eq.5, nostem	0.354	84 (57.9%)	26 (17.9%)
Eq.6, stem	0.343	83 (57.2%)	**24** (16.6%)
Eq.6, nostem	**0.367**	**86** (59.3%)	**24** (16.6%)

Table 6. Evaluation of results using various Okapi probabilistic models (TREC-10)

Table 6 lists the various results of our search models, in which we reported the mean reciprocal rank (MRR), the number of queries for which the correct homepage was found within the first ten retrieved items, and the number of queries for which the relevant entrypage could not be found in our result list. Row two of this table contains an evaluation of the classical Okapi probabilistic model, as described in Section 1.1. As a variation in this particular search problem, we decided to analyze the impact of the stemming procedure, and as seen in row three, we were able improve retrieval effectiveness compared to the classical Okapi model, when the stemming procedure was discarded.

In the fourth and fifth rows are listed our adapted Okapi model's retrieval performance, based on the retrieval status values computed according to Equation 5. Finally, the last two rows show the performance achieved by using Equation 6 with our adaptation of the Okapi model.

From analyzing this data we concluded that the stemming procedure was not really appropriate for this type of search. Moreover, our modified Okapi model provides better results than does the classical Okapi model, and computing retrieval status value using Equation 6 exhibits the best retrieval performance.

2.4. Reranking based on URL length

Upon examining the result lists obtaioed using our Okapi homepage search model, we can find corresponding URL addresses usiog a look-up procedure and pass this list to one of our reranking schemes. In this second step, we first consider the URL address length, its length being defined by the number of "/"s contained within it. If however a URL address ends with a "/", then this final slash is ignored. Also, for any URL addresses ending with "index.html" or "index.htm", these terms are removed before we compute the length. Thus, the URL "www.ibm.com" has the same length as "www.ibm.com/index.html" or "www.ibm.com/".

URL length	Number of observations		
	Number	Percentage	Cumul.
= 1	11,622	0.69%	11,622
= 2	253,250	14.97%	264,872
= 3	458,356	27.09%	723,228
= 4	451,255	26.67%	1,174,483
= 5	297,339	17.57%	1,471,822
= 6	119,035	7.03%	1,590,857
= 7	69,091	4.08%	1,659,948
= 8	19,612	1.16%	1,679,560
= 9	6,399	0.38%	1,685,959
= 10	1,235	0.07%	1,687,194
≥ 11	4,902	0.29%	1,692,096

Table 7a. URL length distribution

Table 7a shows the URL length distribution across our test collection, while Table 7b depicts the same distribution based on our two relevance sets (entrypage 2000 and entrypage 2001). From the training data (denoted "2000"), we fouod that correct answers usually correspond to short URL addresses, those with lengths of 1 (77 cases in Table 7b) or 2 (15 observations). Data from the Entrypage 2001 test collection provided by relevance assessments displays a similar pattern. Thus it seems reasonable to assign more importance to short URL addresses than to longer ones.

Based on these findings, we reranked the retrieved URL addresses according to the inverse of their length, with ties being broken by using retrieval status values (RSV_i) computed according to our Okapi model (Section 2.3).

Table 8 shows the first five URLs retrieved after the first query, according to our adaptation of the Okapi model (top part) or according to our reranking scheme based on URL length (second part). As one can see, the Okapi model retrieved various Web pages from the same Web site ("africa.cis.co.za"). The retrieval status value computed according to this search model, as depicted in Column 4, does not vary greatly. When we reranked this result list according to the inverse of URL length, the relevant item ("africa.cis.co.za:81") appears in the first positioo. However, the first five URLs have the same leogth (1 in this case), and the second key used is always the retrieval status value computed by the Okapi system.

	All relevant items		One rel. item / query	
URL length	2000	2001	2000	2001
= 1	79	138	77	93
= 2	19	56	15	32
= 3	8	33	7	9
= 4	2	16	1	6
= 5	0	7	0	4
= 6	0	0	0	0
= 7	0	2	0	1
Total	108	252	100	145

Table 7b. URL length distribution for a set of relevant items

2.5. Reranking based on URL similarity

URL address length does not however accouot for similarity between request and URL terms. Based oo the data depicted in Table 8, for the response to "Worldnet Africa" we can see that the URL address "www.kvvp.com" response is listed in secood place. Thus, it seems a good idea to rerank the result list provided by our Okapi model, based on a similarity between the URL address and the request.

This type of scheme is advantageous because we can account for any similarity between requests and Web pages, as well as requests and URL addresses. To compute this similarity between requests and URL addresses, we must however take various phenomena into consideration, including acronyms and concatenations of two search keywords found in URL addresses, etc. (see Section 2.1).

The basic principals underlying our similarity measure are described in Table 9, where we distinguish mainly between three cases. First, when a request is one word only, our similarity function determines whether this word appears in the URL head (defined as the server name) or in the URL's last component (defined as the tail of the URL). If it does and the corresponding URL is short (with a length of 1 or 2), we return the maximum value of 1.0. One the other hand, if this search term appears within the URL, the similarity is defined as the ioverse of the URL's length. Finally, we determine whether there might be a fuzzy match between the search keyword and the URL. In this "fuzzySimilarity()" function, we counted the maximum length of the ordered sequence of letters between the search word and each term appearing in

the URL. For example, for the search word "market" and the word "markCie", the maximum ordered sequence is "mark" which has a length of 4. This length is then divided by the maximum length of the two corresponding terms (7 in our case, giving a final fuzzy similarity of 4/7).

As a second case, we computed the similarity between the request and a URL with a length of one. Here we tried to establish a similarity between the request and some variations of this short URL (its acronym, the concatenation of adjacent word pairs, the concatenation of a word with the two letters of the next term).

			Similarity measurement		
Query	URL address	Rank	RSV_i, Okapi	URL request	URL length
	based on our adaptation of the Okapi model				
1	africa.cis.co.za:81/nibs/postnet/ad.html	1	5242.61	0.25	0.25
1	africa.cis.co.za:81/about/newprice.html	2	5140.34	0.333333	0.333333
1	africa.cis.co.za:81/buy/ad/kyalami/kyalami.html	3	5116.08	0.2	0.2
1	africa.cis.co.za:81/buy/ad/fasa/fbhs2.html	4	5110.59	0.2	0.2
1	africa.cis.co.za:81/cape/ctcc/business/taxation.html	5	5109.63	0.2	0.2
	rerank based on URL length				
1	**africa.cis.co.za:81/**	1	4967.93	0.9999	1.0
1	www.kvvp.com:80/	2	2672.09	0.12	1.0
1	www.krok.com:80/	3	2636.3	0.16	1.0
1	www.starhustler.com:80/	4	2560.11	0.18	1.0
1	www.lcrtelecom.com:80/	5	1987.4	0.2	1.0
	rerank based on the similarity URL - request				
1	**africa.cis.co.za:81/**	1	4967.93	0.9999	1.0
1	www.att.com:80/worldnet/	2	2607.29	0.997	0.5
1	www.legnetwork.com:80/worldnet.htm	3	2408.45	0.997	0.5
1	interknowledge.com:80/south-africa/index.html	4	2275.12	0.997	0.5
1	www.biodiv.org:80/africa.htm	5	2143.38	0.997	0.5
	Merged results from our three experts				
1	**africa.cis.co.za:81/**	1	9935.86	1.9998	2.0
1	africa.cis.co.za:81/facility.html	2	10086.1	0.778	1.0
1	africa.cis.co.za:81/fin&tegn/	3	9887.3	0.778	1.0
1	africa.cis.co.za:81/cpyright.html	4	9850.68	0.778	1.0
1	africa.cis.co.za:81/comp.html	5	9809.52	0.778	1.0

Table 8. Example of four ranking approaches for the request "Worldnet Africa" (relevant item depicted in bold)

For example as depicted in Table 9, the request "Worldnet Africa" and the URL "africa.cis.co.za:81" represents a similarity evaluated as 0.9999, and if no match was found, we applied our fuzzy match function.

In the latter case, we computed the similarity between each search keyword and a given URL (function inFuzzy()). To define the similarity measure, we took the number of matches, the length of the URL, the value of the match between the URL head and the URL tail into account, as shown in the last lines of Table 9.

In order to evaluate this reranking scheme, we ranked the URL address result list according to request their similarity. An example of the results of this reranking is shown in Table 8 (third part). For this query one can see the relevant item "africa.cis.co.za:81/" appears in the first position. The following four URL addresses have the same similarity value (depicted in fifth column of Table 8) and are ranked according to their retrieval status values, computed from our adaptation of the Okapi model (shown in the fourth column).

2.6. Evaluation and data mergers

So far, we have described two reranking schemes that might hopefully improve the ranking obtained from our adaptation of the Okapi model (for which the corresponding evaluation is shown in Table 6). Table 10a lists an evaluation of these two reranking schemes under the label "URL length" and "URL simil." The results depicted in this table indicate that we can enhance the mean reciprocal rank (MRR) and the number of queries for which the correct homepage can be found within the first ten retrieved items. Moreover, we may also decrease the number of queries for which the relevant entrypage cannot be found in our result list (having a size of 100). Based on these results, the best scheme seems to be that of reranking, based on URL length.

As shown in Table 8 however, each of our three search systems seems to retrieve different URL addresses. Thus, we have decided to combine these three expert techniques. To do so, we selected the first fifteen retrieved items from each of three result lists and

separately added the corresponding similarities achieved by our three experts. This additive process was chosen because in other data merging contexts, it also provided the best results (Savoy *et al.*, 1997). The result lists are then sorted by URL length, with ties being broken by using the sum of retrieval status values computed by our Okapi model (Section 2.3). For example, the last part of Table 8 shows the first five retrieved items, listed according to this fusion scheme. For the first item, $RSV'_i = 9935.86$ because it was found by both the URL length expert ($RSV'_i = 4967.93$) and the URL similarity expert ($RSV'_i = 4967.93$). For the same reason, the new URL length is set to two and the new URL-request similarity is fixed at 1.9998 (= 2 * 0.9999).

```
similarity (aQuery, anURL) : Real;
    queryLength := queryLength (aQuery);                  // john smith canada -> 3
    urlLength := urlLength (anURL);                       // www.ibm.com/uk/products/ -> 3
    urlHead := urlHead (anURL);                           // www.ibm.com/uk/products/ -> www.ibm.com
    urlTail := urlTail (anURL);                           // www.ibm.com/uk/products/ -> products

    if (queryLength = 1) {
       if ( in(aQuery, urlHead) & (urlLength = 1))  return(1);          // market & www.market.com/
       if ( in(aQuery, urlTail) & (urlLength <= 2))  return(1);     // market & www.iti.com/market/
       if  (in(aQuery, anURL))  return(1/urlLength);                // market & www.iti.com/data/market/
       return (fuzzySimilarity (aQuery, anURL));                    // market & www.marketCie.ch/prod/data/
       }
    if (urlLength = 1) {
       expandQuery := acronym (aQuery);                   // chicago science center + csc
       if (in (expandQuery, anURL) >= 2)  return(1);      // chicago science center csc & www.csc.science.com/
       if (in (expandQuery, anURL))  return(0.9999);      // chicago science center csc & www.csc.com/
       expandQuery := concat (aQuery);                    // john smith + johnsmith
       if (in (expandQuery, anURL))  return(1);           // john smith johnsmith & www.johnsmith.com/
       expandQuery := concat2 (aQuery);                   // advice corp + adviceco
       if (in (expandQuery, anURL))  return(1);           // advice corp adviceco & www.adviceco.com/
       return (fuzzySimilarity (aQuery, anURL));          // advice corp & www.advicorp.com/
       }
    simHead := fuzzySimilarity (aQuery, urlHead);
    simTail := fuzzySimilarity (aQuery, urlTail);
    if (urlLength = 2) {
       if (simTail >= 0.8)  return (simTail);             // sirius bar & www.store.com/sirius/
       if (simHead >= 0.8)  return (0.456)                // iris corp & www.iris.com/ca/
          else return (0);                                // iris corp & www.irt.com/canada/
       }
    aSetMatchFuzzy := inFuzzy (aQuery, anURL);            // desk publ & www.publ.com/uk/desk/ -> (1, 0, 1)
    nbMatch := sizeSet (aSetMatchFuzzy);                  // (1, 0, 1) -> 2
    if (nbMatch = 0)  return (0);                         // when no fuzzy match can be found
    if (urlLength - nbMatch <= 1) {                       // numerous matches
       if ((aSimHead >= 0.8) & (aSimTail >= 0.8))         // if good match with the head and the tail
          return (aSimTail - 0.1);                        // e.g., desk publ & www.publ.com/uk/desk -> 0.9
       if (aSimHead >= 0.8)                               // if good match with only the head
          return (aSimTail - 0.15);
          else return (0.234);                            // if numerous match inside the url
       }
    return (max (0.2, nbMatch / urlLength));              // if some matches
```

Table 9. Outline of algorithm used to determine similarity between query and URL address

However, from considering only the top 15 items for each of our three search models, a maximum of 45 retrieved items per query could be obtained. In order to increase these results to 100 (and to help generate relevance assessments), we expanded this list by adding URL addresses found by our Okapi model.

Run name	MRR	# in top 10	# not found
Okapi only	0.367	86 (59.3%)	24 (16.6%)
URL length	0.653	112 (77.2%)	21 (14.5%)
URL simil.	0.470	95 (65.5%)	18 (12.4%)
Fusion	**0.693**	**115** (79.3%)	**10** (6.9%)

Table 10a. Evaluation of our three search models and their combinations (corrected results)

The evaluation of our combined search model is depicted in the last row of Table 10a. The retrieval performance seems to have increased when considering MRR values or the number of queries for which the relevant item appeared in the top ten records. Moreover, the combination of our experts seems to have a clear impact on the number of queries for which the retrieval system cannot find corresponding relevant items (last column of Table 10a). If our combioed retrieval model seems to perform well, it also has a drawback in that it retrieves various items corresponding to the same Web site, as shown in the last part of Table 8. Thus incorporating a pruning process in our fusion scheme may hopefully enhance retrieval performance.

When we created our official results for the homepage search problem, we selected the wrong Okapi results list before considering our two reranking schemes and our combioed approach. The evaluations based on this incorrect result list are shown in Table 10b, and they reveal the same conclusions as do our unofficial but corrected search schemes (Table 10a).

Run name	MRR	# in top 10	# not found
Okapi only	0.295	64 (44.1%)	38 (26.2%)
URL length	0.598	96 (66.2%)	21 (14.5%)
URL simil.	0.431	81 (55.9%)	31 (21.4%)
Fusion	**0.637**	**100** (69.0%)	12 (8.3%)

Table 10b. Evaluation of our three search models and their combinations (official results)

2.7. Description of our official runs

Table 11 shows our official homepage search runs. The "UniNEep1" run correspoods to the search model merges described in Section 2.6. To produce the "UniNEep2" run in positions 45 to 50 we added the top five URL addresses fouod by our simple search model, as described in Section 2.1 (doc=nnn, query=ntn), if these URLs were not found previously. As depicted in Table 11, this feature does not have any significant impact on retrieval performance.

Run name	MRR	# in top 10	# not found
UniNEep1	0.637	100 (69.0%)	12 (8.3%)
UniNEep2	0.637	100 (69.0%)	11 (7.6%)
UniNEep3	0.529	99 (68.3%)	10 (6.9%)
UniNEep4	0.477	99 (68.3%)	16 (11.0%)

Table 11. Official run evaluation

The "UniNEep4" run represents a variation of our search model, based on the normalized merging of the URL address searches (more precisely the "doc=nnn, query=ntn" model using our both our extended queries (see Table 5)) and our adaptation of the Okapi model (search Web page content, Section 2.3). The last run labeled "UniNEep3" represents the combined search model based on the "UniNEep4" run (with reranking based on URL length and URL similarity).

Conclusion

The various experiments carried out within the Web track demonstrate that:

- our new merging strategy based on results list length may improve average precisioo slightly;
- using a lower value for the parameter b when dealing with short requests may improve retrieval performance;
- our adaptation of the Okapi model for the homepage search problem performs relatively well;
- reranking the URL addresses based on a combination of URL length and URL similarity with the request improves retrieval performance for our Okapi model.

Acknowledgments

The authors would like to thank C. Buckley from SablR for allowing us the opportunity to use the SMART system. This research was supported by the SNSF (Swiss National Science Foundation) under grant 21-58'813.99.

References

Callan, J.P., Lu, Z. & Croft, W.B. (1995). Searching distributed collections with inference networks. In *Proceedings of the ACM-SIGIR'95*, 21-28.

Dumais, S.T. (1994). Latent semantic indexing (LSI) and TREC-2. In Proceedings of TREC'2, 105-115.

Le Calvé, A., & Savoy, J. (2000). Database merging strategy based on logistic regression. *Information Processing & Management*, 36(3), 341-359.

Nielsen, J. (2000). *Designing Web usability: The practice of simplicity*. Indianapolis: New Riders.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001). Approaches to collection selection and results merging for distributed information retrieval. In Proceedings ACM-CIKM'2001, 191-198.

Robertson, S.E., Walker, S. & Beaulieu, M. (2000). Experimentation as a way of life: Okapi at TREC. *Information Processing & Management*, 36(1), 95-108.

Savoy, J. & Picard, J. (2001). Retrieval effectiveness on the Web. *Information Processing & Management*, 37(4), 543-569.

Savoy, J. & Rasolofo, Y. (2001). Report on the TREC-9 experiment: Link-based retrieval and distributed collections. In Proceedings TREC'9, (to appear).

Savoy, J., Le Calvé, A. & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In Proceedings TREC'5, 489-502.

Selberg, E.W. (1999). Towards comprehensive web search. Ph.D. Thesis, University of Washington (WA).

Voorhees, E. M., Gupta, N. K. & Johnson-Laird, B. (1995). Learning collection fusion strategies. In Proceedings of the ACM-SIGIR'95, 172-179.

Walker, S., Robertson, S.E., Boughamen, M., Jones, G.J.F. & Sparck Jones, K. (1998). Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. In Proceedings of TREC'6, 125-136.

Xu, J. & Callan, J. P. (1998). Effective retrieval with distributed collections. In Proceedings of the ACM-SIGIR'98, 112-120.

Fusion de collections dans les métamoteurs

Jacques Savoy[1], Yves Rasolofo[1], Faïza Abbaci[2]

[1]Institut interfacultaire d'informatique, Pierre-à-Mazel 7, 2000 Neuchâtel (Suisse)

[2]Ecole nationale supérieure des Mines de Saint-Etienne
158 cours Fauriel, 42023 St-Etienne cedex 2 (France)

Abstract

We investigate the problem of combining ranked lists of documents provided by multiple search engines. Such a problem must be solved by meta-search engines. In this paper, we suggest a new merging strategy using only the rank of the retrieved items. Moreover, we evaluate various merging approaches based on both a corpus of 2 GB containing news, and a second test-collection of 10 GB of Web pages. Based on our evaluations, our merging approach presents interesting performance and it is well adapted for meta-search engines.

Résumé

Les métamoteurs disponibles sur le Web offrent la possibilité d'interroger de nombreux serveurs d'information soulevant le problème de la fusion des résultats provenant des différents moteurs interrogés. Dans cet article, nous proposons une nouvelle stratégie de fusion n'utilisant que le rang des documents dépistés par les divers moteurs de recherche consultés. De plus, nous évaluons plusieurs approches en utilisant un corpus de 2 GB comprenant des articles de quotidiens et une seconde collection de pages Web d'environ 10 GB. Basée sur nos expériences, notre stratégie, simple et efficace pour la fusion de collections, présente une performance intéressante et se révèle bien adaptée aux métamoteurs de recherche.

Mots-clés : recherche d'informations distribuée, fusion de collection, sélection de collection, Web.

1. Introduction

Pour dépister de l'information dans un environnement distribué comme le Web, la navigation a elle seule ne peut plus être considérée comme une stratégie adéquate et efficace, même avec l'introduction de différentes listes thématiques (par exemple, Yahoo!). Un mécanisme basé sur des requêtes, écrites en langue naturelle, doit être fourni aux internautes afin de rechercher efficacement les pages Web souhaitées. Dans ce but, les moteurs de recherche ont été créés et ils 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 (Schwartz, 1998) comme premier outil de recherche d'informations.

Mais ces outils de dépistage de l'information connaissent plusieurs lacunes. Ainsi, même des systèmes disposant d'une capacité de disque très importante n'indexent qu'une fraction de toute l'information disponible et la couverture de ces moteurs n'augmente pas aussi rapidement que la taille du Web. Par exemple, Northern Light, moteur disposant selon les dernières estimations de la plus grande couverture du Web, n'indexe que 16 % des pages tandis que Lycos couvre environ 2,5 % (Lawrence et Lee Giles, 1999). Afin de proposer une meilleure couverture des sources documentaires du Web, les métamoteurs comme, par exemple, www.MetaCrawler.com ou www.all4one.com interrogent différents moteurs de recherche. Ayant obtenu une liste ordonnée de la part de chaque moteur consulté, ces outils doivent alors fusionner ces réponses afin de ne présenter qu'une seule liste à l'internaute. Selon une étude

récente (Lawrence et Lee Giles, 1999), une couverture presque totale peut être obtenue en soumettant la requête à six moteurs ou plus. A côté de ces métamoteurs généraux, nous voyons apparaître de tels outils pour le Web francophone (par exemple www.ariane6.com ou www.kartoo.com) et d'autres se spécialisent pour apporter une réponse dans un domaine particulier comme, par exemple, les dépêches d'agences et les quotidiens (notre métamoteur disponible à l'adresse www.unine.ch/info/news).

Ce problème de fusion de collections se rencontre également dans les bibliothèques numériques. Dans ce cas, chaque fond documentaire peut être interrogé individuellement mais une réponse adéquate nécessite généralement la fusion des résultats provenant de plusieurs voire de toutes les sources. Pour être plus précis, notre démarche s'inscrit plus particulièrement dans le contexte des métamoteurs ne disposant que de listes ordonnées de résultats. En effet, tous les moteurs de recherche n'indiquent pas un score ou un degré de pertinence calculé pour chaque site retourné.

Cet article est organisé de la manière suivante. Dans le deuxième chapitre, nous décrirons les deux collections de documents utilisées et nous présenterons une première évaluation en optant pour une approche centralisée dans laquelle tous les documents disponibles sont regroupés pour former une seule banque de documents. Ensuite, nous distribuerons nos sources selon différents critères et nous analyserons la performance obtenue en interrogeant les différentes collections ainsi créées. Dans cette troisième partie, nous présenterons également une nouvelle stratégie pour résoudre efficacement et simplement le problème de fusion de collections.

2. Approche centralisée

La recherche d'informations s'appuie sur une longue tradition empirique encourageant les chercheurs à évaluer leur système de dépistage sur la base de collections-tests. Ces corpus disposent d'un nombre important de documents et un ensemble de requêtes. De plus, pour chaque requête, nous disposons de la liste des documents jugés pertinents, jugement posé par un être humain. Dans cet article, nous évaluerons différents modèles de recherche à l'aide de deux collections afin d'obtenir une plus grande validité pour nos résultats. En effet, les caractéristiques, souvent inconnues, d'un corpus peuvent favoriser une approche au détriment d'une autre. De plus, nos investigations ne seront pas limitées à un seul modèle de dépistage de l'information, mais nous évaluerons plusieurs stratégies de recherche permettant ainsi d'obtenir une meilleure compréhension de l'efficacité de divers modèles de dépistage de l'information.

2.1. Les collections-tests

Afin d'évaluer expérimentalement nos propositions, nous avons choisi un corpus de documents rédigés en langue anglaise correspondant 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*, des documents du *Federal Register*, des dépêches du *Foreign Broadcast Information Service* et des articles du journal *Los Angeles Times*. Avec ce corpus, nous disposons de 50 requêtes couvrant non pas un domaine restreint mais présentant un éventail assez large de thèmes (par exemple «Estonia, economy», «suicides», «airport security», «osteoporosis», «cosmic events»). Comme l'ordinateur ne possède que le titre des besoins d'information, le texte disponible se révèle très bref, comportant en moyenne, deux mots (écart type de 0,97). La table 1 présente quelques statistiques associées à ce corpus ainsi que le nombre de formes dis-

tinctes par collection, le nombre de requêtes pour lesquelles la collection possède au moins un document pertinent et le nombre de requêtes pour lesquelles le corpus retourne une réponse comportant au moins un document.

Afin de compléter cette première évaluation, nous avons également repris la collection de pages Web utilisée lors de la neuvième conférence TREC, corpus nommé TREC9. Cet ensemble comprend 1 692 096 pages Web écrites en anglais pour un volume de 11 033 MB. Cette seconde collection possède donc un volume approximativement six fois supérieur et présente des documents de nature très varié. Ainsi, il n'existe pas de vrai contrôle éditorial sur le contenu et le nombre de fautes d'orthographe se révèle plus important. Dans le cadre de ce corpus, une division logique selon les sources n'avait pas beaucoup de sens et nous avons donc généré huit collections possédant un nombre approximativement égal de documents. Selon les statistiques disponibles dans les tables 1 et 2, nous remarquerons que, pour nos deux collections-tests, chacune des collections ainsi générée ne possède pas des réponses pertinentes pour toutes les requêtes. Ainsi, si le corpus TREC9.2 possède au moins un article pertinent pour 44 des 50 requêtes, le corpus TREC9.4 ne dispose de réponses adéquates que pour 21 requêtes. La situation se présente de manière assez similaire pour le corpus TREC8.

Collection	Taille (en MB)	nb documents	nb de formes	nb req. pert.	nb req. réponse
FT	564	210 158	375 499	49	50
FR	395	55 630	196 220	18	50
FBIS	470	130 471	502 099	43	50
LA Times	475	131 896	337 492	45	50
TREC8	1 904	528 155	1 008 463	50	50

Table 1. Quelques statistiques sur les collections TREC8

Collection	Taille (en MB)	nb documents	nb de formes	nb req. pert.	nb req. réponse
TREC9.1	1 325	207 485	1 800 230	28	48
TREC9.2	1 474	207 429	2 274 318	44	48
TREC9.3	1 438	221 916	1 783 691	38	48
TREC9.4	1 316	202 049	1 684 452	21	48
TREC9.5	1 309	203 073	1 988 627	22	48
TREC9.6	1 311	215 451	2 136 650	24	48
TREC9.7	1 336	200 146	2 223 065	25	47
TREC9.8	1 524	234 547	2 134 040	43	48
TREC9	11 033	1 692 096		50	48

Table 2. Quelques statistiques sur les collections TREC9

Avec le corpus TREC9 correspondant plus à notre centre d'intérêt, nous disposons aussi de 50 requêtes différentes, couvrant plusieurs domaines et correspondant à des exemples soumis au moteur de recherche *Excite*. Les thèmes abordés sont très variables et le texte soumis se révèle très bref comportant en moyenne 2,4 mots (écart type de 0,6). De plus, les termes employés sont très souvent ambigus et très fréquents (par exemple, «deer», «baltimore»). L'orthographe n'est pas toujours exacte («tartin» pour «tartan», «angioplast7» ou «nativityscenes»), l'emploi des majuscules n'est pas systématique («CHEVROLET TRUCKS», «Toronto FIlm Awards» ou «steinbach nutcracker») et parfois la ponctuation est absente («what is the compostion of zirconium»). Ces exemples démontrent les difficultés du traitement de la langue naturelle dans notre contexte.

2.2. Performance de l'approche centralisée

Afin de décrire un modèle de dépistage de l'information, nous devons répondre à trois questions. D'abord savoir comment l'ordinateur représente les documents, ensuite comment les requêtes seront traitées et représentées. Dans les différentes stratégies étudiées dans cet article, cette représentation se limite à un ensemble de termes simples pondérés. Dans tous les cas, un pré-traitement est prévu afin d'éliminer les formes très fréquentes et peu ou non porteuses de sens (comme les articles «the», «of», les prépositions «and», «or» ou les pronoms «I», «we») d'une part, et d'autre part, à éliminer la marque du pluriel («beavers» deviendra «beaver») voire d'autres dérivations suffixales (par exemple, «discussing» redonnera «discus») (Savoy, 1997).

Finalement, un modèle spécifie la manière dont l'appariement entre la requête et les documents se réalise. Pour ce dernier aspect, et dans toutes nos expériences, le degré de similarité de chaque document avec la requête (ou son degré de pertinence jugé par la machine) est obtenu par le calcul du produit interne (Salton, 1989, p. 318).

Pour indexer un document ou une requête, différentes stratégies de pondération existent. La manière la plus simple est d'effectuer une indexation binaire (approche notée «bnn»). Le système d'indexation retiendra les formes les plus représentatives du contenu sémantique du document ou de la requête. Si nous recourons à cette approche tant pour les documents et que pour les requêtes (approche notée «doc : bnn, requête : bnn»), le degré de similarité de chaque document indiquera le nombre de termes communs entre celui-ci et la requête.

Evidemment, d'autres stratégies d'indexation ont été proposées et nous pouvons tenir compte de la fréquence d'occurrence des termes d'indexation dans le document ou la requête (approche «nnn»). Ainsi, si un mot se répète dans une page, son importance sera accrue lors de l'indexation. Dans le modèle «doc : nnn, requête : nnn», le calcul du degré de similarité 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 du terme dans le document et de l'inverse de sa fréquence documentaire (ou nombre de documents dans lesquels le terme apparaît). Ainsi, si un mot apparaît très souvent dans des documents, son poids diminuera lors de l'indexation. En effet, un tel terme ne permettra pas à l'ordinateur de distinguer les documents pertinents des autres. Finalement, dans cette stratégie «doc : ntc, requête : 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).

Ces dernières années, et en particulier sous l'impulsion des conférences TREC, d'autres modèles de dépistage de l'information ont été proposés, en particulier diverses variantes du modèle vectoriel comme les stratégies «doc : Lnu, requête : ltc» (Buckley et al., 1996), «doc : dnu, requête : dtc» (Singhal et al., 1999) ou l'approche «doc : atn, requête : ntc». Depuis quelques années, l'approche probabiliste Okapi (Robertson et al., 1995) rencontre également un grand succès grâce à la bonne précision moyenne de cette stratégie. Ce modèle sera donc aussi repris dans nos évaluations.

Comme méthodologie d'évaluation, nous avons retenu la précision moyenne (Salton, 1983, Section 5.2) mesurant la qualité de la réponse fournie par l'ordinateur, mesure utilisée par la conférence TREC. L'évaluation des différentes stratégies de dépistage retenues au regard de nos deux collections-tests est indiquée dans la table 3. Finalement, pour décider si un système de dépistage est meilleur qu'un autre, nous admettons comme règle d'usage qu'une différence de 5 % dans la précision moyenne peut être considérée comme significative.

	Précision moyenne (% changement)	
collection	TREC8	TREC9
nombre doc. pertinents	4 728	2 617
modèle	50 requêtes	50 requêtes
doc: Okapi, requête: npn	25,66	19,86
doc : dnu, requête : dtn	24,03 (-6,4 %)	15,42 (-22,4 %)
doc : atn, requête : ntc	22,75 (-19,1 %)	14,59 (-26,5 %)
doc : Lnu, requête : ltc	21,63 (-15,7 %)	16,81 (-15,4 %)
doc : lnc, requête : ltc	14,44 (-43,7 %)	3,79 (-80,9 %)
doc : ltc, requête : ltc	13,60 (-47,0 %)	5,20 (-73,8 %)
doc : ntc, requête : ntc	12,21 (-52,4 %)	8,00 (-59,7 %)
doc : bnn, requête : bnn	11,51 (-50,1 %)	5,13 (-74,2 %)
doc : nnn, requête : nnn	4,47 (-82,6 %)	4,06 (-79,6 %)

Table 3. Précision moyenne des moteurs de recherche sur les deux corpus (approche centralisée)

Sur la base des évaluations indiquées dans la table 3, nous noterons que le modèle probabiliste Okapi présente la meilleure précision moyenne et que cette dernière est significativement supérieure aux autres approches (différence de précision moyenne supérieure à 5 %). En deuxième position, nous trouvons l'approche «doc : dnu, requête : dtn» pour le corpus TREC8 ou «doc : Lnu, requête : ltc» pour la collection TREC9 extraite du Web. Ces résultats indiquent également que le modèle vectoriel classique «doc : ntc, requête : ntc» occupe seulement le septième rang pour le corpus TREC8 et le cinquième dans le cadre de la collection TREC9.

Les requêtes étant différentes d'un corpus à l'autre, une comparaison directe entre les performances sur la collection TREC8 et TREC9 n'est donc pas possible. Nous remarquons que les modèles vectoriels «doc : lnc, requête : ltc» ou «doc : ltc, requête : ltc» ont plutôt été conçus pour des collections de documents possédant peu de fautes d'orthographe. Pour ces approches, la précision moyenne sur des pages Web se dégradent fortement. De plus, l'approche basée sur une indexation binaire («doc : bnn, requête : bnn») semble aussi connaître une baisse d'efficacité lorsque cette stratégie est utilisée sur le Web.

3. Fusion de collections

Dans un premier temps, l'ensemble des documents étaient réunis pour former une seule banque de documents. Cette approche constituait un axiome dans la recherche d'informations classique (Salton, 1989). Or, le corpus TREC8 peut se diviser logiquement en différentes sources correspondant aux différents quotidiens. Cette situation reflète bien le cas d'une bibliothèque numérique possédant plusieurs sources d'information pouvant être interrogées par le même moteur. Ainsi, en réponse à une requête d'un usager, le système d'interrogation envoie cette demande à l'ensemble des moteurs de recherche. Ces derniers calculent leur réponse et la retournent au système d'interrogation. Ce dernier peut soit présenter la liste obtenue par chaque source séparément, soit fusionner ces listes pour n'en visualiser qu'une seule. La manière de générer cette liste unique en ordonnant les résultats en fonction de leur pertinence par rapport à la requête soumise correspond au problème de fusion des collections. Ce phénomène se rencontre également dans les métamoteurs de recherche dont la qualité de réponse dépend fortement de la stratégie de fusion de collections utilisée.

3.1. Stratégies de fusions

Afin de résoudre ce problème de fusion, différentes propositions ont été avancées. Comme première approche, nous pouvons admettre que chaque collection 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, nommée «à chacun son tour», se rencontre souvent dans certains métamoteurs (Selberg, 1999) et, en particulier, cette approche a été préconisée par les premiers outils de ce type disponibles sur Internet.

Cependant, en plus du rang de chaque article dépisté, les moteurs de recherche fournissent parfois un score (degré de similarité) entre la requête et le document retourné. Or cette information n'est pas fournie de façon systématique par tous les moteurs de recherche. Dans le cas des métamoteurs, nous ne pouvons donc pas baser une stratégie de fusion sur ce renseignement, invalidant ainsi plusieurs stratégies de fusion proposées (Callan et al., 1995).

Dans un article récent, Yager et Rybalov (1998) suggèrent une généralisation de la fusion «à chacun son tour» à l'aide d'un paramètre noté α. Pour comprendre les grandes lignes de cette stratégie de fusion, un exemple nous aidera. Imaginons que nous ayons interrogé quatre collections dont la première retourne les neuf documents ordonnés suivants (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9), la deuxième cinq documents (b_1, b_2, b_3, b_4, b_5), la troisième trois documents (c_1, c_2, c_3) et la quatrième un seul document (d_1). La stratégie «à chacun son tour» effectuera la fusion en retournant la liste résultante suivante (a_1, b_1, c_1, d_1, a_2, b_2, c_2, a_3, b_3, c_3, a_4, b_4, a_5, b_5, a_6, a_7, a_8, a_9). Le résultat sera identique si l'on recourt à l'algorithme de Yager et Rybalov avec $\alpha = 0$. Dans ce cas, nous privilégions les documents retournés dans les premiers rangs par les divers moteurs. En revanche, si nous fixons le paramètre $\alpha = 1$, nous obtiendrons (a_1, a_2, a_3, a_4, a_5, b_1, a_6, b_2, a_7, b_3, c_1, a_8, b_4, c_2, a_9, b_5, c_3, d_1). Comme les listes résultantes possèdent des longueurs différentes, nous favorisons les plus longues listes au détriment des listes plus brèves. Entre ces deux possibilités extrêmes, nous pouvons faire varier le paramètre α pour donner au rang une plus grande importance (α faible) ou plutôt pour favoriser les longues listes (α proche de 1). En fixant le paramètre $\alpha = 0{,}5$, nous obtiendrons (a_1, a_2, a_3, b_1, a_4, b_2, c_1, a_5, b_3, c_2, d_1, a_6, b_4, c_3, a_7, b_5, a_8, a_9) ce qui correspond à accorder autant d'importance à la longueur des listes qu'au rang des documents dépistés.

Pour être complet, signalons les travaux de (Le Calvé et Savoy, 2000) et de (Voorhees et al., 1995) qui proposent de recourir à un algorithme d'apprentissage (régression logistique respectivement classification) afin d'améliorer la fusion des listes de résultats. Or, dans la pratique, nous ne disposons pas souvent de requêtes passées avec leurs jugements de pertinence afin d'ajuster au mieux les paramètres sous-jacents à un modèle.

3.2. Stratégies de sélections

Afin d'améliorer la fusion des résultats provenant de différentes sources, nous avons imaginé une stratégie de sélection visant à exclure les moteurs dont la réponse n'apportait aucun document pertinent. En effet, en réponse à une requête très brève, les moteurs de recherche interrogés retournent très souvent une réponse dans laquelle les articles dépistés contiennent bien le (ou les) terme(s) de la requête mais utilisé(s) dans un contexte différent. Ainsi, face à la requête «hair transplant», un moteur dépistera des sites Web parlant de transplantation cardiaque ou de coiffure mais pas du sujet précis intéressant l'internaute.

Afin de mettre en œuvre cette stratégie de sélection, notre système d'interrogation inspecte le contenu des cinq premiers documents retournés à la recherche de l'occurrence de deux (ou plusieurs) termes de la requête dans un contexte restreint, comme le titre, une phrase ou un paragraphe. Or les expériences menées nous ont conduit dans une impasse. En effet, les pages Web ne disposent pas souvent d'un titre et souvent les deux termes de la requête n'apparaissent pas ensemble dans la partie logique encadrée par la balise <H1>. De plus, il était rare de voir conjointement plusieurs termes de la requête dans la même page. Enfin, l'évaluation de ces différentes approches n'a pas permis d'améliorer la précision moyenne ; l'effet de ces diverses stratégies de sélection était, le plus souvent, une baisse significative de la performance moyenne.

3.3. Notre stratégie de fusion

Ayant renoncé à opérer une sélection, nous avons conçu une stratégie de fusion fonctionnant en deux étapes. En premier lieu, nous évaluons l'importance de chaque collection consultée. Cette pondération est proportionnelle au nombre de documents extraits (ou longueur de la liste L_i retournée) par ce corpus par rapport au maximum des différentes listes retournées par tous les moteurs (maximum noté $\max(L_j)$). Cette pondération, attribuée à la i^e collection et représentée par la valeur α_i, s'évalue selon la formule suivante :

$$\alpha_i = (1 - k) + k \cdot [\log(1 + L_i) / \log(1 + \max(L_j))]$$

dans laquelle k est une constante (fixée à 0,4 dans nos évaluations). Dans notre modèle, le recours au logarithme se justifie par notre volonté d'attribuer une pondération de moins en moins forte par rapport au nombre croissant de documents retournés. Ainsi, si ce nombre passe de 10 à 11, nous considérons que cet accroissement unitaire doit être valorisé de manière plus importante que le passage de 210 à 211.

Sur la base de cette pondération, notre algorithme de fusion calcule une probabilité de pertinence, notée Prob [D | rg], dépendant du rang (noté rg) du document D extrait par la i^e collection selon l'équation suivante :

$$\text{Prob}[D \mid rg] = \exp(\alpha_i + \beta \cdot \log(rg)) / [1 + \exp(\alpha_i + \beta \cdot \log(rg))]$$

dans laquelle rg indique le rang obtenu par le document D dans la i^e collection considérée, β une constante (fixée à 0,05 dans nos évaluations). Cette dernière équation dérive directement de la méthode de la régression logistique (Bookstein et al., 1992). Durant la phase de fusion des listes, cette probabilité calculée pour chaque document dépisté est utilisée comme critère de tri, débutant par la probabilité la plus élevée vers la valeur la plus faible.

3.4. Evaluation

A priori, il se révèle difficile de sélectionner la meilleure stratégie de fusion de collections. Afin d'obtenir une meilleure vue d'ensemble de ces différentes approches, nous les avons évaluées à l'aide de nos deux collections-tests. Dans les tables 4 et 5, en plus de la performance obtenue par un moteur gérant de manière centralisée l'ensemble des documents disponibles, il nous paraît intéressant d'évaluer la précision moyenne de l'approche «à chacun son tour» optimale. Dans ce cas, le système de fusion connaît le rang de tous les documents pertinents dépistés. Dans une situation aussi favorable, il peut ignorer certaines collections et extraire autant d'éléments d'une liste que nécessaire. Selon nos expériences, la fusion optimale permet d'accroître la précision moyenne d'environ 11,5 % sur la collection TREC8 et 49,1 % sur le corpus TREC9.

stratégie de fusion / modèle	Précision moyenne: collection centralisée TREC8	«à chacun son tour» optimale	«à chacun son tour»	Yager *et al.* α = 0,5	notre approche
doc:Okapi, req:npn	25,66	27,63	18,84	19,30	22,14
doc : dnu, req : dtn	24,03	25,60	17,40	18,12	20,77
doc : atn, req : ntc	22,75	24,17	16,18	16,58	18,93
doc : Lnu, req : ltc	21,63	22,06	15,31	17,07	18,57
doc : lnc, req : ltc	14,44	15,37	10,23	11,43	12,63
doc : ltc, req : ltc	13,60	15,42	10,24	10,44	12,05
doc : ntc, req : ntc	12,21	12,24	8,17	9,99	9,97
doc : bnn, req : bnn	11,51	12,45	7,58	8,28	9,39
doc : nnn, req : nnn	4,47	6,82	4,37	5,36	5,20
gain / perte moyen en %		+11,47 %	-26,18 %	-18,67 %	-11,40 %

Table 4. Précision moyenne de différentes stratégies de fusion (TREC8)

La stratégie «à chacun son tour» permet d'obtenir une précision moyenne d'environ 26,2 % à 28,9 % inférieure à l'interrogation d'une collection unique regroupant tous les documents. Nous pouvons donc affirmer que chaque collection contient un nombre variable de documents pertinents et que ceux-ci ne se retrouvent pas distribués de manière identique dans les réponses obtenues des serveurs. Pour mesurer l'efficacité de l'approche proposée par (Yager et Rybalov, 1998), nous avons fixé le paramètre α à 0,5, valeur attribuant autant de poids au rang qu'à la longueur des listes retournées. Cette dernière présente une meilleure efficacité pour le corpus TREC8 (table 4) que la fusion «à chacun son tour». Cette constatation ne se révèle pas confirmée par le corpus extrait du Web (table 5).

Notre modèle de fusion propose la meilleure stratégie pour les deux corpus utilisés. Tenir compte de l'efficacité des différents moteurs de recherche en prenant en compte le nombre d'articles retournés par chacun d'eux s'avère une stratégie intéressante. Cependant, l'accroissement n'est pas aussi net et significatif pour la collection TREC9. En se limitant au meilleur moteur de dépistage de l'information, c'est-à-dire le modèle Okapi, notre stratégie de fusion propose, pour le corpus TREC8, une précision moyenne de 22,14 contre 18,84 pour l'approche «à chacun son tour» soit une augmentation de 17,5 %. Pour la collection TREC9, la performance obtenue s'élève à 12,6, soit 1,8 % de mieux que la fusion «à chacun son tour».

stratégie de fusion / modèle	Précision moyenne: collection centralisée TREC9	«à chacun son tour» optimale	«à chacun son tour»	Yager *et al.* α = 0,5	notre approche
doc:Okapi, req:npn	19,86	22,58	12,38	11,76	12,60
doc : Lnu,req : ltc	16,81	19,29	9,81	9,05	9,88
doc : dnu, req : dtn	15,42	18,82	10,37	9,53	10,63
doc : atn, req : ntc	14,59	19,01	9,93	10,63	10,18
doc : ntc, req : ntc	8,00	11,99	5,37	5,14	5,88
doc : ltc, req : ltc	5,20	10,01	4,40	4,46	4,43
doc : bnn, req : bnn	5,13	8,27	3,91	3,47	3,96
doc : nnn, req : nnn	4,06	6,22	2,89	3,16	3,26
doc : lnc, req : ltc	3,79	7,74	3,21	3,20	3,15
gain / perte moyen en %		+49,09 %	-28,91 %	-30,26 %	-26,64 %

Table 5. Précision moyenne de différentes stratégies de fusion (TREC9)

La performance calculée dans les tables 4 et 5 s'est effectuée en utilisant le même moteur pour interroger les quatre respectivement huit collections des corpus TREC8 et TREC9. Or, les métamoteurs interrogent des moteurs œuvrant avec des stratégies de dépistage différentes pour chaque collection. Afin d'analyser cette situation, nous allons interroger les deux corpus en recourant à différents moteurs de recherche. Dans notre cas, nous avons retenu le modèle probabiliste Okapi et les trois approches vectorielles dont la distribution pour la collection TREC8 est la suivante FT interrogée par «doc : Okapi, requête : npn», FR par «doc: dnu, requête : dtn», FBIS via «doc : Lnu, requête : ltc» et LA Times par «doc : atn, requête : ntc». Pour le corpus TREC9, la distribution des stratégies de recherche est la suivante : TREC9.1 et TREC9.5 par «doc : Okapi, requête : npn», TREC9.2 et TREC9.6 via «doc : dnu, requête : dtn», TREC9.3 et TREC9.7 par «doc : Lnu, requête : ltc» et TREC9.4 et TREC9.8 par «doc : atn, requête : ntc». Dans cette situation, nous ne pouvons donc plus donner la performance obtenue par une approche centralisée pour fin de comparaison.

	Précision moyenne (% changement)			
stratégie de fusion modèle	«à chacun son tour»	«à chacun son tour» optimale	Yager *et al.* $\alpha = 0,5$	notre approche
TREC8	16,85	24,73	17,64	19,95
TREC8 + permutation	16,83	24,02	18,02	19,92
TREC8 + permutation	17,12	24,47	17,97	20,27
TREC8 + permutation	17,16	24,99	17,33	20,06
moyenne TREC8	16,99	24,55 (+44,5%)	17,74 (+4,4%)	20,05 (+18%)
TREC9	10,19	19,83	10,42	10,35
TREC9 + permutation	10,88	19,19	10,86	10,96
TREC9 + permutation	11,42	20,87	10,49	11,37
TREC9 + permutation	9,86	19,34	9,37	10,15
moyenne TREC9	10,59	19,81 (+19,8%)	10,29 (-2,9%)	10,71 (+1,1%)

Table 6. Précision moyenne de différentes stratégies de fusion

La table 6 indique la précision moyenne de quatre stratégies de fusion. Selon ces dernières expériences, nous constatons que notre stratégie de fusion présente encore la meilleure performance. Afin de renouveler notre expérience, nous avons permuté trois fois la distribution des quatre stratégies de recherche par rapport aux différentes collections. En observant l'efficacité dans chaque cas ou en calculant la moyenne, notre approche propose une meilleure efficacité qui se révèle même significative dans le cas de la collection TREC8 et presque significative dans le cadre du corpus TREC9 (10,71 contre 10,29, augmentation de 4,1 % par rapport à l'approche de Yager et Rybalov (1998)).

4. Conclusion

Sur la base de nos expériences basées sur deux collections-tests, neuf stratégies de dépistage et des requêtes extrêmement courtes (mais similaires à celles que nous rencontrons sur le Web), les conclusions suivantes peuvent être tirées. D'abord, le modèle probabiliste Okapi présente une performance très attractive. Il propose significativement la meilleure réponse dans le cadre de nos deux collections-tests œuvrant sur des volumes importants (2 GB et 10 GB). Ensuite, le modèle vectoriel classique «doc : ntc, requête : ntc» ne propose pas une stratégie d'indexation et de dépistage efficace. Or cette approche est souvent recommandée. Finalement, notre stratégie de fusion propose une alternative simple et efficace dont la précision moyenne se révèle meilleure que celle de la fusion «à chacun son tour» ou que

l'approche proposée par (Yager et Rybalov, 1998). N'utilisant que le rang et la longueur des listes retournées, notre stratégie de fusion est donc très adaptée aux métamoteurs ou aux bibliothèques numériques œuvrant sur un ensemble de sources documentaires interrogées avec le même moteur de dépistage de l'information. Nos travaux en cours tentent à trouver de nouvelles stratégies afin d'améliorer encore la performance des métamoteurs de recherche (voir notre métamoteur, encore en phase expérimentale, à l'adresse www.unine.ch/info/news). Enfin, le problème de fusion de collections se rencontre également dans la recherche multilingue (Peters, 2001) dans laquelle la requête soumise dans une langue donnée doit dépister des documents écrits dans différentes langues. Dans de tels systèmes, la requête soumise est souvent traduite automatiquement et est soumise à diverses collections, chacune renfermant des documents rédigés dans une langue donnée. La fusion doit donc s'opérer dans un contexte quelque peu différent ; chaque collection étant interrogée par une requête différente car écrite avec des mots appartenant à diverses langues naturelles (Savoy, 2002).

Remerciements

Cette recherche a été subventionnée en partie par le FNS avec le subside 21-58 813.99 (J. Savoy et Y. Rasolofo) et par la Région Rhône-Alpes (bourse Eurodoc de F. Abbaci).

Références

Bookstein A., O'Neil E., Dillon M., Stephen D. (1992). Applications of Loglinear Models for Informetric Phenomena. *Information Processing & Management*, vol.(28):75-88.

Buckley C., Singhal A., Mitra M., Salton G. (1996). New Retrieval Approaches using SMART. Proceedings of *TREC'4*, pages 25-48.

Callan J. P., Lu Z., Croft W. B. (1995). Searching Distributed Collections with Inference Networks. Proceedings of the *ACM-SIGIR'95*, pages 21-28.

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

Lawrence S., Lee Giles C. (1999). Accessibility of Information on the Web. *Nature*, vol.(400):107-110.

Le Calvé A., Savoy J. (2000). Database Merging Strategy Based on Logistic Regression. *Information Processing & Management*, vol.(36):341-359.

Peters, C. (Ed.) (2001). *Cross-Language Information Retrieval and Evaluation, Workshop of the Cross-Language Evaluation Forum, CLEF-2000.* Lecture Notes in Computer Science, vol.(2069), Berlin : Springer-Verlag.

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):345-360.

Salton G. (1989). *Automatic Text Processing : The Transformation, Analysis, and Retrieval of Information by Computer.* Reading (MA) : Addison-Wesley.

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

Savoy J., Rasolofo Y. (2000). Recherche d'informations dans un environnement distribué. Actes de *TALN 2000*, pp. 317-326.

Savoy J. (2002). Report on CLEF-2001 Experiments: Effective Combined Query-Translation Approach. In Peters, C., editor, *Cross-Language Information Retrieval and Evaluation, Workshop*

of the Cross-Language Evaluation Forum, CLEF-2001. Lecture Notes in Computer Science. Berlin : Springer-Verlag.

Savoy J. (1997). Statistical Inference in Retrieval Effectiveness Evaluation. *Information Processing & Management*, vol.(33):495-512.

Schwartz C. (1998). Web Search Engines, *Journal of the American Society for Information Science*, vol. (49):973-982.

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

Singhal A., Choi J., Hindle D., Lewis D. D., Pereira F. (1999). AT&T at TREC-8. Proceedings of *TREC-8*, pp. 239-251.

Voorhees E. M., Gupta N. K., Johnson-Laird B. (1995). Learning Collection Fusion Strategies. Proceedings of the *ACM-SIGIR'95*, pp. 172-179.

Yager R. R., Rybalov A. (1998). On the Fusion of Documents from Multiples Collection Information Retrieval Systems. *Journal of the American Society for Information Science*, vol.(49):1177-1184.

Annexe 1. Formules d'indexation

Afin d'attribuer un poids w_{ij} reflétant l'importance de chaque terme d'indexation T_j, $j = 1, 2, ..., t$, dans un document D_i, nous pouvons recourir à l'une des formules décrites dans la table ci-dessous. Dans cette dernière, tf_{ij} indique la fréquence d'occurrence du terme T_j dans le document D_i (ou dans la requête), n représente le nombre de documents D_i dans la collection, df_j le nombre de documents dans lesquels le terme T_j apparaît (fréquence documentaire), et idf_j l'inverse de la fréquence documentaire ($idf_j = \log[n/df_j]$). Les constantes ont été fixées aux valeurs suivantes : slope=0,2, pivot=150, b=0,75, k=2, k_1=1,2, advl=900. De plus, la longueur du document D_i (ou le nombre de termes d'indexation associé à ce document) est notée par nt_i, la somme de valeurs tf_{ij} par l_i et $K = k \cdot [(1 - b) + b \cdot (l_i/advl)]$.

bnn	$w_{ij} = 1$	nnn	$w_{ij} = tf_{ij}$
dtn	$w_{ij} = [\log(\log(tf_{ij}) + 1) + 1] \cdot idf_j$	atn	$w_{ij} = idf_j \cdot [0{,}5 + 0{,}5 \cdot tf_{ij} / \max tf_{i.}]$
Okapi	$w_{ij} = \left((k_1 + 1) \cdot tf_{ij}\right) \Big/ \left(K + tf_{ij}\right)$	npn	$w_{ij} = tf_{ij} \cdot \log\left[\left(n - df_j\right) \Big/ df_j\right]$
lnc	$w_{ij} = \dfrac{\log(tf_{ij}) + 1}{\sqrt{\sum_{k=1}^{t} \left(\log(tf_{ik}) + 1\right)^2}}$	ntc	$w_{ij} = \dfrac{tf_{ij} \cdot idf_j}{\sqrt{\sum_{k=1}^{t} \left(tf_{ik} \cdot idf_k\right)^2}}$
ltc	$w_{ij} = \left(\log(tf_{ij}) + 1\right) \cdot idf_j \Big/ \sqrt{\sum_{k=1}^{t} \left(\left(\log(tf_{ik}) + 1\right) idf_k\right)^2}$		
dnu	$w_{ij} = \dfrac{\left(1 + \log\left(1 + \log(tf_{ij})\right)\right) \Big/ \left(1 + pivot\right)}{(1 - slope) \cdot pivot + slope \cdot nt_i}$		
Lnu	$w_{ij} = \dfrac{\left(1 + \log(tf_{ij})\right) \Big/ \left(1 + pivot\right)}{(1 - slope) \cdot pivot + slope \cdot nt_i}$		

Table 7. Formules de pondération lors de l'indexation

Result Merging Strategies for a Current News MetaSearcher

Yves Rasolofo[†], David Hawking[‡], Jacques Savoy[†]

[†] Institut interfacultaire d'informatique
Université de Neuchâtel, Switzerland
Yves.Rasolofo@unine.ch, Jacques.Savoy@unine.ch

[‡] CSIRO Mathematical and Information Sciences
Canberra, Australia
David.Hawking@cmis.csiro.au

Abstract

Metasearching of online current news services is a potentially useful Web application of distributed information retrieval techniques. We constructed a realistic current news test collection using the results obtained from 15 current news websites (including ABC News, BBC and AllAfrica) in response to 107 topical queries. Results were judged for relevance by independent assessors. Online news services varied considerably both in the usefulness of the results sets they returned and also in the amount of information they provided which could be exploited by a metasearcher. Using the current news test collection we compared a range of different merging methods. We found that a low-cost merging scheme based on a combination of available evidence (title, summary, rank and server usefulness) worked almost as well as merging based on downloading and re-scoring the actual news articles.

Keywords: Metasearch, distributed information retrieval, evaluation, Web.

1. Introduction

A metasearcher is a broker which forwards search queries to a set of primary search engines (each assumed to provide incomplete coverage[1] of the set of documents to be searched)

[1] Coverage is the proportion of available documents which are actually indexed.

and presents to the searcher a single merged list of results. This situation is depicted in Figure 1. In general, a metasearcher may incorporate solutions to the problems of:

1. identification and characterization of primary search services whose results are to be merged;
2. selection, for reasons of efficiency or effectiveness, of a subset of available search services (so called server or collection selection problem);
3. translation of the searcher's requests into the relevant query language of each primary search service and getting/parsing results;
4. merging of results from the primary search services into a single high-quality list (collection fusion or results merging problem).

A number of Web metasearchers have been available to the public for some time. MetaCrawler (Selberg and Etzioni, 1997) is perhaps the best known example. Like others, this search engine merges the results of a set of ten or a dozen general-purpose primary search engines, each of which attempts to index "the whole of the Web". MetaCrawler increases effective coverage because of the relatively poor overlap between the sets of documents indexed by the primary engines (Lawrence and Lee Giles, 1999). It also attempts to improve result quality by boosting the rank of documents ranked highly by several of the engines (voting). As another example, we may cite www.all4one.com which sends the user's request to only four search engines, (Alta Vista, Yahoo!, HotBot and Excite) and presents the retrieved items in four different windows, one for each search engine. When using a non-merging metasearcher like this, the user must inspect different windows, a situation that is subject of various criticisms from a usability point of view (Nielsen, 2000).

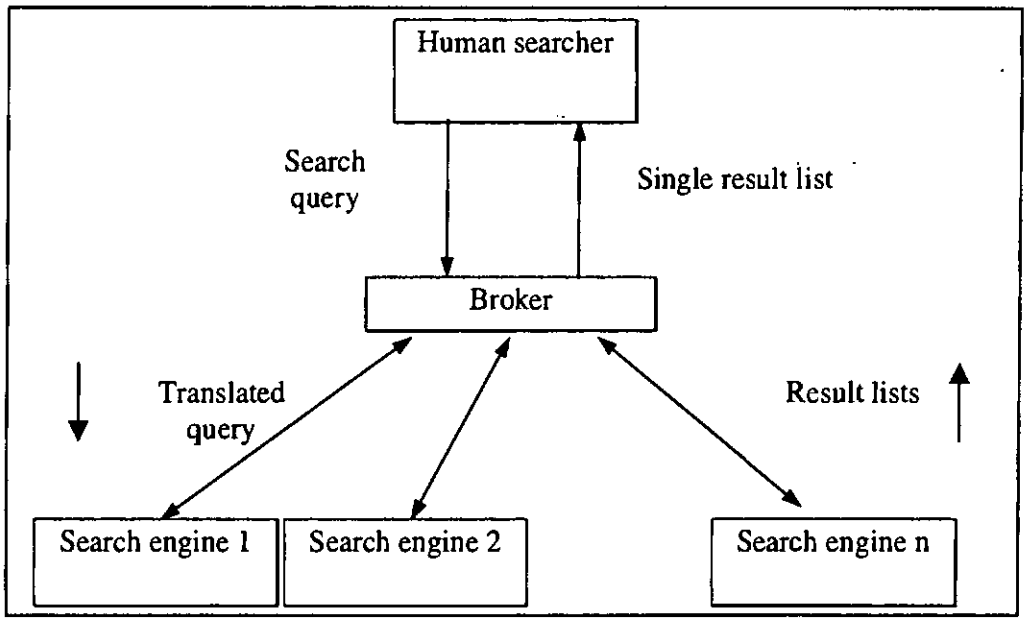

Figure 1: Human searcher, broker and primary search services

Metasearching, a search approach based on the paradigm of distributed information retrieval, have been promoted as a solution to some of the difficulties experienced by centralized search engines such as Alta Vista and Google, due to the huge size and rapid growth of the World Wide Web. Despite some doubts, the proportional coverage of the largest search engines is believed to have increased despite several years of sustained Web growth. More importantly, several recent studies of search engine effectiveness (Craswell *et al.*, 2001), (Hawking *et al.*, 2001a), (Hawking *et al.*, 2001b) have failed to show an effectiveness benefit on precision-oriented tasks from this type of metasearching.

It is possible that the Web will grow to be so large that metasearching becomes essential. It is equally possible that other more clearly beneficial applications for metasearching techniques already exist.

This paper is organized as follows: The rest of this section reviews past work and describes the underlying characteristics of current news servers. Section 2 describes our experimental framework and the methodology used in this paper. Section 3 evaluates individually the various online news sites used in our study. In Section 4, we present and evaluate various merging

strategies used to present to the user a single list of the retrieved items. Finally, we present some conclusions that can be drawn from our study.

1.1. Previous work in selection and results merging strategies

A large number of published studies have addressed the server selection and results merging problems in the context of test collections such as TREC ad hoc, TREC VLC and WT2g, as for example (Callan *et al.*, 1995), (Callan, 2000), (French *et al.*, 1998), (Gravano *et al.*, 1994), (Hawking and Thistlewaite, 1999), (Le Calvé and Savoy, 2000), (Rasolofo *et al.*, 2001). In these experiments, the test collection is divided into a set of disjoint partitions, each representing the documents indexed by a search service.

When analyzing selection procedures, suggested methods for identifying the partitions containing the greatest number of relevant or "good" documents are compared, and the results of retrieval over selected subsets of partitions are compared with retrieval results for the full collection. The methods proposed by Gravano *et al.* (1997), Hawking and Thistlewaite (1999) for server selection and by Kirsch (1997) are unrealistic in the sense that they rely on the widespread adoption of protocols by which search engines can communicate statistics or metadata about their holdings. At the time of writing, this seems very unlikely.

On the other hand, when studying merging experiments, the operation of a search service is typically simulated over each test collection partition and the quality of merged results lists corresponding to various merging strategies are compared. Merging of partitioned collections represents real-world search in the sense that several public Web search engines (such as Inktomi, Google and Fast) in fact operate by dividing the collection (the set of pages discovered and retrieved for indexing by a spider[2]) and indexing each partition separately. However, the suggested merging schemes are simple in this particular case because each partition is managed by the same search algorithm and because communication of global statistics is straightforward. When dealing with metasearching, we are faced with different retrieval engines for which the indexing and retrieval strategies are usually unknown. This fact invalidates most of the previous

[2] A spider (also known as a crawler or robot) discovers pages for indexing by recursively following hyperlinks from a seedlist of Web pages.

studies in results merging approach, with the exception of the work of Le Calvé & Savoy (2000) which however requires a learning phase and Craswell *et al.*'s work (1999) which used mixed engines.

1.2. The current study

As mentioned, it is probable that real-world search applications exist in which metasearching offers significant benefits over centralized indexes. We consider that search of current news services such as CNN, the BBC and ABCNEWS might constitute such an example, because:

- there are many online news services which provide a search interface to their own recent news stories;
- some news services may be only available to subscribers. However, article titles and summaries are often available for searching and they are enough for metasearching. It is up to the user to decide whether he or she will subscribe;
- high quality current news search (as opposed to search of news archives) requires that searchers be able to retrieve news stories the instant they are placed on the site. This is clearly not achievable by a centralized index which crawls news sites periodically. Of course, we recognize that there are ways in which centralized search engine companies can provide rapid indexing of news articles. For example, business relationships may be formed with news organizations in which search engine companies are notified of each new news story as it is posted, or in which subscriber-only data may be indexed for a fee. Alternatively, current news services may be identified and "spidered" much more frequently than other Web sites by centralized search engines.

Current news metasearch represents a more realistic and a more difficult application area in comparing metasearching methods than does disjoint partitioning of an existing test collection. The partitioning and distribution of a collection across multiple servers has no obvious application on the Web. By contrast, current news metasearch is realistic for reasons listed above. Characteristics of current news metasearch are: The search algorithms used by the various news sites are unknown and heterogeneous. These search servers do not cooperate with each other or with a broker, and the information available to guide selection and merging is variable in

type and is generally small in quantity. Finally, response from current news servers is often slow and the issue of timeouts must be addressed.

In order to address these questions, we have created a current news metasearch test collection by broadcasting more than one hundred topical news queries to fifteen current news Web sites, and judging responses for relevance. In this paper, we will document this new test collection and report its use in comparing various results merging strategies based on information we were able to extract from news articles and servers.

1.3. Issues specific to current news servers

This study focused upon the evaluation of news metasearcher addressing fifteen servers. It is important to note that a metasearcher for current news servers differs in different aspects from those manipulating conventional search engines. These differences are the following:

- in general, the titles of documents returned by current news services are more accurate and reliable than those typically available with other Web documents. Thus, titles may be a beneficial source of evidence in ranking articles;
- the documents are particularly volatile and become out of date very quickly. The article date is therefore critical for some topics. Moreover, it is important to choose appropriate topics for the evaluation;
- the retrieval effectiveness and efficiency of the news search engines vary widely from one server to another. It would be beneficial to take into account these differences when merging results lists.

Moreover when setting up our news metasearcher, we faced the following additional problems:

- although a server may not return any document for a given query at a given time, it does not always mean that it does not contain any documents related to the query. The reason could be transient server or network overload. This problem is not serious for our evaluation because our main goal is to find effective ways to merge the results lists from servers;

- it is important to write wrappers[3] that are able to collect all required information which might be of value. We concentrated our efforts on extracting URL, title, summary,. document date when such information was available;
- each search engine included in news servers has its own query language. For our evaluation, we used the "AND" Boolean operator or equivalent;
- some engines return front pages containing only headlines. As far as possible, we tried not to include them in the final results list of the corresponding server.

2. Experimental framework

In order to conduct metasearching experiments, we have written a metasearch engine which is the fruit of an international collaboration between Switzerland and Australia. The code of this metasearch engine was written in Perl and represents around 5,000 lines of code. While the user interface is available both in Switzerland (http://www.unine.ch/info/news/) and Canberra (http://peace.anu.edu.au/Yves/MetaSearch/), the core engine is running in Canberra. This section introduces the basic ideas underlying our metasearcher. Section 2.1 describes the fifteen selected online news servers together with examples of queries generated by users. Section 2.2 presents the problem of unavailable documents or broken links included in the servers answers. Section 2.3 explains how we have established the relevance assessments of our selected queries. Finally, the last section describes the evaluation methodology used in this paper.

2.1. Queries and current news sites

People from different backgrounds (Australian, African and Swiss) and professions (computer scientists, psychologists, research assistants in sociology, students in philosophy and art-history) were asked to generate short "bag of words" queries that they might write to a search interface of a current news service on the Web. Queries covered world news, science & technology, business, entertainment and sport. A total of 118 queries were collected with a query

[3] A wrapper provides a common programming interface to a search service. It translates a searcher's request into the query language of a primary search service and extracts information from returned results.

length varying from one to eight words (average length = 3.21) corresponding roughly to the mean length of search engine queries. From these requests, 114 search queries have at least one document returned by a server and a total of 107 queries were evaluated, corresponding to queries having at least one relevant document returned. Some of the queries are:

- Lockerbie trial
- dotcom stocks
- ELF corruption case
- MIR space station
- Madagascar eclipse 2001
- Tom Cruise Nicole Kidman separation
- George W. Bush Administration
- Arnaud Clement

Queries were submitted to news sites on February 9th and 26th, 2001. We used a script to remove stop words from queries and submit the remaining words to news servers using the "AND" Boolean operator or equivalent. The news sites included in our evaluation were selected among sites from around the world (east and west Europe, USA, Africa, Australia and Asia) and are:

- ABCNEWS: http://abcnews.go.com
- BBC Online: http://www.bbc.co.uk
- CNET: http://news.cnet.com
- CNN Financial Network: http://cnnfn.cnn.com
- FINANTIAL TIMES: http://www.ft.com
- MSNBC: http://www.msnbc.com
- NEWS.COM:AU: http://www.news.com.au
- THESTAR.COM: http://www.thestar.ca
- THE TIMES: http://www.thetimes.co.uk
- THE TOKYO WEEKLY: http://www.tokyo-weekly.ne.jp
- ALLAFRICA.COM: http://allafrica.com
- THE ST. PETERSBURG TIMES: http://www.sptimes.ru
- USATODAY: http://www.usatoday.com

- DISPATCH Online: http://www.dispatch.co.za
- WORLDNEWS.COM: http://www.worldnews.com

When building our test collection, we considered only the top ten documents returned by each news server, excluding front pages (going past ten if some pages are front pages). Limiting the number of documents per server and request to ten documents, we may potentially find a total of $114 \cdot 10 \cdot 15 = 17{,}100$ documents. However, for many queries (see Table 1), some servers do not return any documents, and others return less than ten articles. Therefore, a total of only 5,544 documents were returned, including 236 broken links. This corresponds to a mean of approximately 46 live documents per query.

As shown in Table 1, from a total of 114 submitted requests, the news servers return an answer for 50.5% of these queries. With a standard deviation of 30.07, we can clearly see that there is a great variation across news servers answers. Moreover, one can see that few queries are answered by TOKYO-WEEKLY because its database is only updated weekly and it is devoted mainly to local news. As another example, THESTAR for its part tended to respond slowly and was timed out many times.

Table 2 depicts the number of queries over a total of 114 having at least one relevant item for each of the selected servers. For all news sites, there is a clear decrease (relative to Table 1) in the number of queries, showing that servers tend to send an answer to a user's request even if its database does not contain any relevant information. In other words, the underlying search engine does not know when it knows nothing on a given subject.

Table 1: Number of queries having at least one document returned

Server	# queries	%
WORLDNEWS	111	97.4%
THETIMES	95	83.3%
ALLAFRICA	90	78.9%
DISPATCH	90	78.9%
SPTIMES	71	62.3%
MSNBC	63	55.3%
FT	56	49.1%
BBC	56	49.1%
CNNFN	54	47.4%
USATODAY	50	43.9%
ABCNEWS	41	36.0%
NEWS.COM.AU	36	31.6%
CNET	32	28.1%
TOKYO-WEEKLY	10	8.8%
THESTAR	9	7.9%
Total	114	100%
Average	57.60	50.53%
Standard deviation	30.07	

Table 2: Number of queries having at least one relevant doc returned (among 114 queries)

Server	# queries	%
WORLDNEWS	91	79.8%
DISPATCH	71	62.3%
THETIMES	66	57.9%
BBC	53	46.5%
FT	47	41.2%
ALLAFRICA	45	39.5%
SPTIMES	43	37.7%
ABCNEWS	40	35.1%
MSNBC	36	31.6%
USATODAY	31	27.2%
CNNFN	24	21.1%
CNET	24	21.1%
NEWS.COM.AU	23	20.2%
TOKYO-WEEKLY	4	3.5%
THESTAR	1	0.9%
Total	107	93.9%
Average	39.93	35.03%
Standard deviation	24.21	

2.2. Broken links

When documents were not available due to broken links, they were removed from the test collection because assessors did not have enough information to judge them. As shown in Table 3, most of the selected news sites keep the number of broken links as low as possible (ten sites out of fifteen have less than 4% of broken links). For two cases however, MSNBC (17.35% of the retrieved articles were not available) and THESTAR (53.57% of broken links), we can see that these online services seem not to update their indexes sufficiently often.

Table 3: Broken links across the news servers

Server	Total links	Broken links	%
NEWS.COM.AU	115	0	0.00%
TOKYO-WEEKLY	28	0	0.00%
BBC	443	1	0.23%
FT	303	1	0.33%
ALLAFRICA	618	3	0.49%
USATODAY	353	2	0.57%
ABCNEWS	161	1	0.62%
DISPATCH	425	4	0.94%
CNNFN	289	4	1.38%
CNET	253	9	3.56%
THETIMES	728	32	4.40%
WORLDNEWS	1,108	66	5.96%
SPTIMES	277	26	9.39%
MSNBC	415	72	17.35%
THESTAR	28	15	53.57%
Sum	5,544	236	4.26%

2.3. Relevance assessments

The relevance assessments were carried out in Canberra, Australia. Five research assistants were recruited for this purpose - four Australians and one U.S. citizen. Each had completed or very nearly completed a university degree. All were familiar with searching and browsing via the Web but none were information technology professionals. None of them were connected with the companies operating the current news sites being evaluated.

Judging was carried out using a Web browser. We implemented a Perl CGI script which presented a merged list of all the results pages for a particular query, interleaved by source. Unfortunately, judging could not be blind as results pages were usually heavily branded with the identity of the source. Clicking on a link opened a new window displaying the target page from the news site. Relevant / Irrelevant (binary) judgments were recorded using radio buttons.

Judges were asked to imagine that the queries were theirs and to evaluate on that basis. All the documents retrieved for a query were judged by the same person, ensuring consistency of the results. The assessment was performed from February 14th, 2001 to March 13th, 2001.

2.4. Evaluation measures

In order to obtain an overall measure of performance, we used TREC average precision, a relatively stable single-number measure which combines elements of both precision and recall. This measure is used for results merging comparison (Voorhees and Harman, 2000).

A decision rule is required to determine whether or not a given search strategy is better than another. We considered statistical inference methods such as Wilcoxon's signed rank test or the Sign test (Salton and McGill, 1983), (Hull, 1993) applied to average precision. In these cases, the null hypothesis H_0 states that both retrieval schemes produce similar retrieval performance. Such a null hypothesis plays the role of a devil's advocate, and this assumption will be accepted if two retrieval schemes return statistically similar average performance on a query-by-query basis, and rejected if not. Thus, in the tables found in this paper we will use the Sign test, based on the average precision with a significance level fixed at 5%. However, a decision to accept H_0 is not equivalent to the opinion that the null hypothesis H_0 is true, but instead represents the fact that "H_0 has not been shown to be false" resulting in insufficient evidence against H_0. When reading the Sign test results, the decision ">" or "<" means that the corresponding hypothesis "<=" or ">=" is rejected.

As our experimental news metasearcher displays only one page of merged results, we were interested to know the improvement that could be obtained on the average precision after ten and after twenty retrieved documents. These values were also included in our evaluations.

3. Individual news server evaluation

Servers vary considerably in the number of queries for which they return any answer. As shown in column 2 of table 4, out of a total of 118 queries, the selected servers provide answers for an average of 57.6 queries. There is a large difference between the best server, WORLDNEWS which provided 111 answers, and THESTAR with only 9 answers.

In order to obtain a first overview of the retrieval effectiveness of our fifteen news servers, we computed the average precision on an individual basis for each of the selected news services. This mean precision is calculated by averaging the precision achieved by each query returning at least one document (see Table 4).

When analyzing the mean performance provided by these servers (third column of Table 4), the average retrieval performance is 40.67 (median: 39.33) and the standard deviation around this mean is relatively high (30.07). On the other hand, the standard deviation associated with each server's performance shown in the last column indicates that we may encounter a relatively large variation across queries, varying from 25 (THESTAR) to 43.35 (NEWS.COM.AU).

Table 4: Average precision over queries for which the server returned at least one document

Server	# queries	Mean precision (%)	Standard deviation
ABCNEWS	41	81.43	27.68
BBC	56	68.07	31.63
FT	56	63.92	37.47
NEWS.COM.AU	36	50.57	43.35
DISPATCH	90	44.97	36.24
WORLDNEWS	111	43.57	31.18
THETIMES	95	40.78	36.52
CNET	32	39.33	33.10
MSNBC	63	37.01	38.67
SPTIMES	71	35.71	37.90
TOKYO-WEEKLY	10	27.08	41.07
CNNFN	54	25.30	34.66
USATODAY	50	22.01	25.39
ALLAFRICA	90	21.94	29.56
THESTAR	9	8.33	25.00
Average	57.60	40.67	33.96
Standard deviation	19.42	30.07	

In Table 5, we do the same computation except that we exclude requests having a zero precision, resulting in performance increases. From this table, one can see that the average precision for a news server is 62.32%. The performance difference between the best server (ABCNEWS, 83.46) and the worst (USATODAY, 35.51) reflects clearly that the retrieval performance varies to a greater extent in this kind of online service. Individually, the variation in mean performance is shown in the last column of Table 5. In this case, the difference between the largest variation (TOKYO-WEEKLY, 37.33) and the smallest (USATODAY, 23.63) is smaller than in Table 4.

Table 5: Average precision over queries for which the server returned at least one relevant document

Server	# queries	Mean precision (%)	Standard deviation
ABCNEWS	40	83.46	24.73
NEWS.COM.AU	23	79.15	24.97
FT	47	76.17	26.95
THESTAR	1	75.00	N/A
BBC	53	71.92	27.85
TOKYO-WEEKLY	4	67.71	37.33
MSNBC	36	64.77	28.29
SPTIMES	43	58.96	31.46
THETIMES	66	58.70	29.34
DISPATCH	71	57.00	31.21
CNNFN	24	56.92	29.84
WORLDNEWS	91	53.15	25.95
CNET	24	52.44	27.56
ALLAFRICA	45	43.88	27.99
USATODAY	31	35.51	23.63
Average	39.93	62.32	28.36
Standard deviation	24.21	13.45	

When studying the differences between Table 4 and 5, one can see that some servers may contain relevant articles for a subset of the selected topics, and the average precision is high only for those queries. On the other hand, for other requests, the retrieval performance tend to be 0. As an example, Table 5 shows that NEWS.COM.AU is better than BBC and FT news servers when computing only queries that return at least one relevant document. However, the ranking is reversed when looking at Table 4.

4. Merging strategies

In the previous section, we have evaluated individually each news server. However, our purpose is to evaluate metasearching and to achieve this purpose, we have to send the user's request to our fifteen selected servers and to merge the results lists of retrieved documents into a single list to be presented to the user. To achieve this, our broker may adopt various merging strategies that will be analyzed in this section.

As a first approach, we may assume that each news server contains approximately the same number of relevant documents and that they are equally distributed across results lists from servers (Voorhees *et al.*, 1995). Under this assumption, we can set up the final results list in a round-robin manner. Such a round-robin merging strategy (RR) is already used by earlier metasearch engines on the Web. This merging scheme will serve as a baseline for further comparisons in the following sections.

Moreover, we might also consider merging the results lists by sorting retrieved articles according to the score computed by the server they are originated from. Such a merging approach will be called "raw-score merging". However, it was not practical because the document scores were seldom reported. In any case scores would not have been comparable due to differences in indexing and retrieval strategies used by the servers.

In this section, we will thus consider various merging schemes that may produce comparable scores based on the information available from the servers, such as document title. To this purpose, Section 4.1 presents a general scoring function employed by our broker to compute a score to each retrieved article. Section 4.2 explains how we may exploit the document title in defining a document score. In Section 4.3, we explore the possibility to take account of document summaries in our merging scheme. In Section 4.4, we develop a merging approach that utilizes the title and the summary of the retrieved item in order to improve our merging strategy. As another merging approach, Section 4.5 suggests using the document date while Section 4.6 presents how we may obtain various estimated collection statistics in order to propose another merging model. Section 4.7 describes how we may take into account the server usefulness. Finally, the last section proposes to inspect the article contents in order to define a more effective merging strategy.

4.1. Generic document scoring function

Since document scores were seldom reported by the news servers and are not comparable, we had to define a general scoring function that returns comparable scores based on various document fields (e.g., document title, article summary or date) in order to define an effective merging strategy. Thus, for each document i belonging to collection j for the query Q, we will compute a weight, denoted w_{ij} as follows:

$$w_{ij} = \frac{NQW_i}{\sqrt{L_q^2 + LF_i^2}}$$

within which NQW_i is the number of query words appearing in the processed field of the document i, L_q is the length (number of words) of the query, and LF_i is the length of the processed field of the document i.

This function returns a value of 0 when the intersection between the request and the selected document field is empty. On the other hand, when all search terms and only those appear in the processed field, our function returns the maximum value of $1/\sqrt{2}$ (or 0.7071).

This suggested formula is based on intuition that the more the search keywords appear in the processed document field(s) the greater the probability that the corresponding article is relevant. However, a weighting expression should not be a simple proportion of the number of query words appearing in the document field(s), it is also important to consider at the same time the length of both the request and the document field(s) as suggested in the previous formula.

Some of the algorithms described below may use the rank score, denoted RS_{ij}, and defined as $RS_{ij} = 1000 - SR_{ij}$ where SR_{ij} is the rank given by the server j to the document i. Since we consider a maximum of 10 items per server, this rank score function returns values between 999 for the best ranked document to 990 for the last retrieved article.

When we are able to provide a comparable score for each retrieved document, our broker must merge the various results lists to form a unique list of retrieved articles. To achieve this, we may adopt the RR merging scheme corresponding to the round-robin merging strategy based only on the ranks defined by servers. As an alternative, we may compute for each document a score using the XX field with our generic document score function. Based on these scores, we may adopt the raw-score merging approach that will be denoted SM-XX. However, instead of directly using the document score, we may also consider using only the rank obtained after applying our generic document scoring function. In this case, we will denote such a merging model as RR-XX.

The difference between SM-XX and RR-XX merging approaches is the following. In both cases, we compute a new document score based on the defined scoring function. When using the SM-XX approach, we merge all the results lists together and we sort it according to the document

score. Ties, if any, are broken according to document rank and in favor of the article appearing in the lower rank. In the RR-XX scheme, we independently sort the results list for each server. After these sorts, we apply the round-robin strategy on the new results lists. In this case, we expect to have roughly the same number of documents extracted from each server (the number of extracted items from each server will be the same if all results lists have the same length). On the other hand, using the SM-XX merging strategy, a server providing articles with higher document score will furnish more items in the final merged list.

4.2. Document title scoring (TS)

As a first merging strategy, we may consider using the document title of the retrieved records. Scoring the article using its title is done by assigning to the document the weight $w_{ij} \cdot 100{,}000$ in which w_{ij} is computed according to our generic function described in the previous section and using the title as processed field. However, the value w_{ij} can be zero for some documents (e.g., article having no title or when no search keyword appears in the title). In this case, instead of assigning a value of 0 to such document, we attach to it its rank score RS_{ij}. After all, such a document cannot be viewed as irrelevant simply because its title does not share any common word with the request. Knowing that the rank score value varies from 990 to 999 on the one hand, and, on the other, that the document title score varies from 0 to 70,710, we are sure when multiplying w_{ij} by 10,000 that the retrieved documents having a document score greater than 0 will be ranked before articles having no title or having a title without any word in common with the request.

In order to compare this merging strategy with the round-robin approach, we have reported in Tables 6 and in Figure 2 the retrieval effectiveness achieved using RR-TS (round-robin merging scheme based on the rank obtained by the document title scoring), SM-TS (raw-score merging approach based on document title scoring) and the baseline solution RR (round-robin merging strategy based on the original ranked lists given by the news servers).

Table 6a: Document title scoring

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
RR (baseline)	48.97		42.52		40.42	
RR-TS	55.00	12.31%	49.16	15.62%	44.95	11.21%
SM-TS	63.76	30.20%	61.03	43.53%	50.56	25.09%

Table 6b: Sign test results

RR < RR-TS
RR < SM-TS
RR-TS < SM-TS

From data depicted in Tables 6, one can see that both RR-TS and SM-TS merging strategies increase the retrieval effectiveness over the round-robin approach. As shown in Table 6b, this improvement is statistically significant. When comparing the relative merit of our two suggested merging schemes, we may see that the improvement is greater for the SM-TS compared to the RR-TS scheme and this variation is also statistically significant (last row of Table 6b). In fact, the round robin hypothesis is invalidated by the fact that servers hold different numbers of relevant documents (see Section 2.1)

To have an overview of the retrieval performance of these three merging approaches, Figure 2 depicts the precision computed after retrieving different numbers of document. From this figure, it is clear that the SM-TS approach shows better performance at low cut-off values and seems to perform well when considering the first five or the first ten retrieved items.

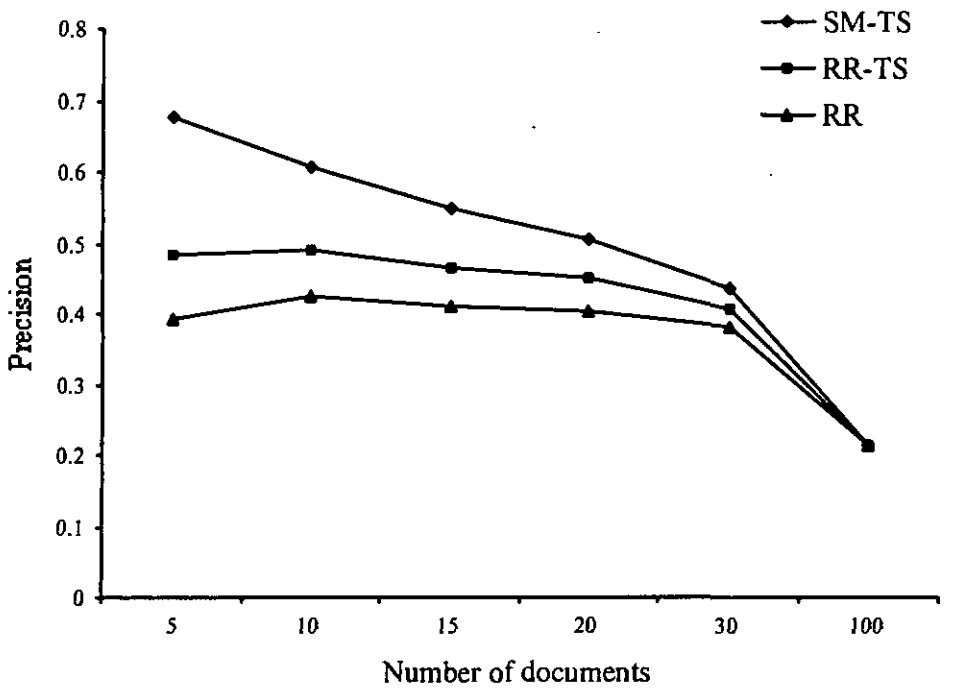

Figure 2: Precision of top results of SM-TS, RR-TS and RR

4.3. Document summary scoring (SS)

When retrieving a document from a current news server, we usually find a document title and an article summary. Such a summary consists of an abstract or sometimes corresponds to the head of the article (e.g., the first 200 words of the news). However, we must indicate that the title is more frequently associated with the document than a summary field.

In defining a merging scheme for our news services, we may exploit our generic document scoring function with this summary field. The retrieval performances achieved by the RR-SS round-robin merging strategy and the SM-SS raw-score merging approach based on the summary field are reported in Table 7a. As shown in Table 7b, these two schemes improve significantly the retrieval effectiveness over the simple round-robin model (RR). As before, the SM-SS raw-score approach presents a better performance that is also statistically significantly better than both round-robin models. Comparing the data of Table 7a with those of Table 6a, we can see that document title scoring seems to produce better retrieval performance than document summary scoring.

Table 7a: Document summary scoring

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
RR (baseline)	48.97		42.52		40.42	
RR-SS	53.45	9.15%	47.57	11.88%	43.50	7.62%
SM-SS	62.21	27.04%	60.47	42.22%	50.00	23.70%

Table 7b: Sign test results

RR < RR-SS
RR < SM-SS
RR-SS < SM-SS

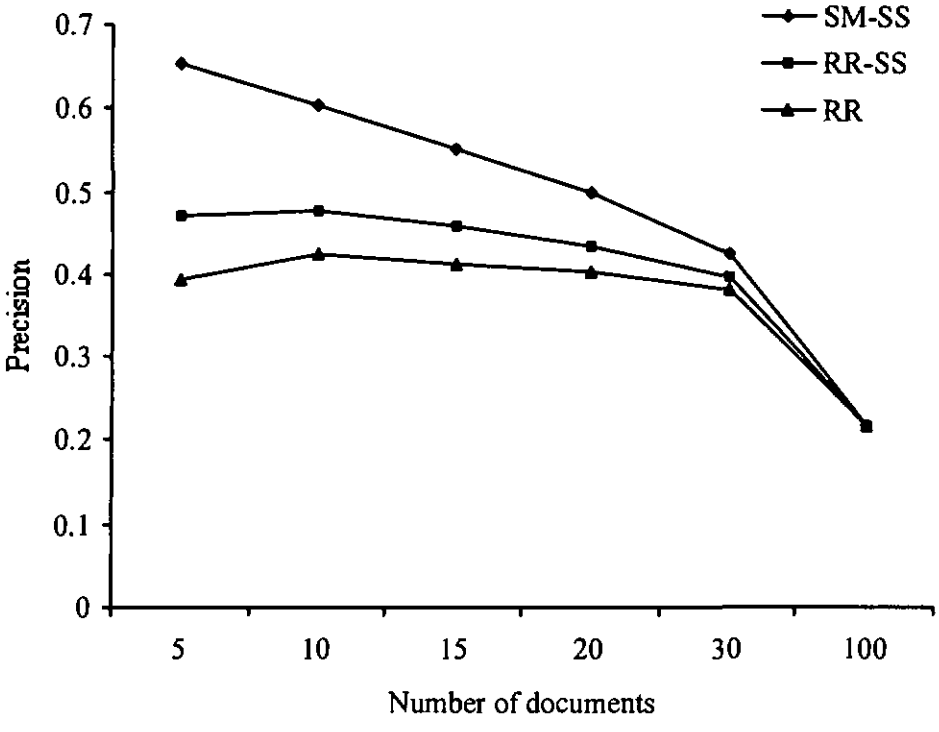

Figure 3: Precision of top results of SM-SS, RR-SS and RR

4.4. Combining title score and summary score (TSS)

In order to improve our merging process, we may take account for both the document title and the article summary in our generic document scoring. To achieve this, a first combined scoring approach for document i belonging to collection j is used as follows:

```
W_ij = TS_ij  /* first use the title field */
if W_ij == 0
     W_ij = SS_ij  /* if none, use the summary field */
endif
if W_ij == 0
     W_ij = RS_ij /* if none, use the rank */
endif
```

where RS_{ij} is the rank score described in Section 4.1. Recall that this latter value is always lesser than both the TS_{ij} or SS_{ij} score value as long as they are greater than zero. This combined scoring approach is grounded on our observation that the document title seems to reflect more accurately the article content than does the associated document summary. Therefore, we only considered the summary score when the title score was null.

The evaluation of this combined merging approach is depicted in Table 8a under the label "RR-TSS1" when we adopt a round-robin merging procedure or "SM-TSS1" when we chose the raw-score merging strategy. The average precision or the precision achieved after retrieving ten or twenty documents represents the highest retrieval performance seen so far. As shown in Table 8b, these two merging schemes are statistically better than the simple round-robin merging strategy (RR). Moreover, the Sign test indicates that the SM-TSS1 approach significantly improves the retrieval effectiveness over both the document title scoring (SM-TS) and the document summary scoring approach (SM-SS).

Table 8a: Title and summary combined scoring

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
RR (baseline)	48.97		42.52		40.42	
RR-TSS1	56.94	16.28%	52.24	22.86%	46.59	15.26%
SM-TSS1	67.14	37.10%	63.74	49.91%	53.88	33.30%

Table 8b: Sign test results

RR < RR-TSS1
RR < SM-TSS1
RR-TSS1 < SM-TSS1
SM-TS < SM-TSS1
SM-SS < SM-TSS1

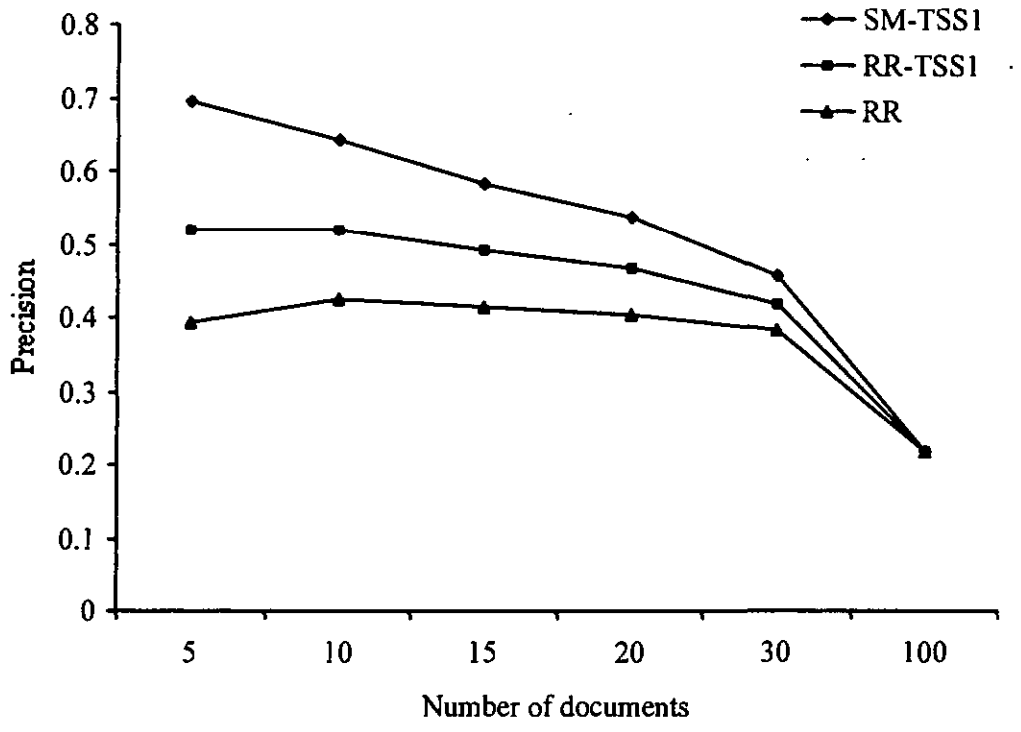

Figure 4: Precision of top results of SM-TSS1, RR-TSS1 and RR

As a second combined approach, we compute a new document score as a linear combination of the document title and article summary scoring as follows:

$$w_{ij} = k \cdot TS_{ij} + (1-k) \cdot SS_{ij}$$

where k is a constant between 0 and 1. We arbitrarily chose k = 0.9 for which we obtained the best retrieval performance. Table 9 depicts the retrieval effectiveness of this second combined merging strategy. As one can see, the resulting performances are close to those achieved by our first merging approach (see Table 8a). When using the Sign test (results not shown in this paper), we obtain the same conclusions than those presented in Table 8b. However, the drawback of this second merging procedure is the need to set the underlying constant k.

Table 9: Linear combination of the title and summary scoring function

Merging strategies	Average Precision	% change	Precision @10	% change	Precision @20	% change
RR (baseline)	48.97		42.52		40.42	
RR-TSS2	56.59	15.56	51.96	22.20	46.50	15.04
SM-TSS2	67.06	36.94	63.08	48.35	53.79	33.08

4.5. Document date

In the previous raw-score merging schemes, ties were broken according to document rank and in favor of the article appearing in the lower rank. Such ties appear when two articles have very similar titles (or summaries) or both titles have the same length and share the same number of keywords with the request. Instead of breaking ties according to the document rank, we suggest breaking ties according to document date and in favor of the article owning a more recent date. Such an approach is already used in Boolean search engines and this scheme seems to be even more appropriate when dealing with current news services. Formally, the date score is calculated as follows:

$$SS = 1000 - (\text{TodayDate} - \text{DocumentDate})$$

where TodayDate was set to the assessment date. In case of a date difference greater than 1,000, SS is set to 0. This breaking scheme can be incorporated in the various raw-score merging strategies described previously, namely the document title (SM-TS-D), the document summary (SM-SS-D) or the combined merging approach (SM-TSS1-D).

Table 10a: Including document date for breaking ties

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
SM-TS	63.76		61.03		50.56	
SM-TS-D	65.21	2.35%	61.31	0.46%	50.93	0.83%
SM-SS	62.21		60.47		50.00	
SM-SS-D	62.94	1.88%	60.84	0.61%	49.63	-0.74%
SM-TSS1	67.14		64.11		53.83	
SM-TSS1-D	68.01	1.22%	64.11	0.73%	53.41	-0.61%

Table 10b: Sign test results

SM-TS < SM-TS-D
SM-SS < SM-SS-D
SM-TSS1 < SM-TSS1-D

Table 10a depicts the retrieval effectiveness of this strategy compared to the scheme based on the document rank. All the results based on the document date seem to be slightly better than those achieved by the scoring function using the document rank computed by the news servers. When using the Sign test as shown in Table 10b, the document date brought some improvement. When studying our second combined approach, we obtained the same result as with our first combined scheme.

4.6. Estimated collection statistics (ECS)

Well-known and effective merging techniques like CORI (Callan *et al.*, 2000) are based on the knowledge of various collection statistics like document frequency. However, in our context, it is impossible to obtain cooperation among news servers in order to obtain the required statistics. Moreover, we cannot download all documents containing a given search term from all servers to get the needed information. As an approximation to obtain an estimate of the required statistics, we use the title and summary fields provided within the list of the top 10 retrieved documents in order to calculate document frequency. Such information will be used to define a collection score.

With this objective in mind, we adopt the Okapi probabilistic model (Robertson *et al.*, 2000) which will be applied not to rank retrieved documents but to define a collection score for each selected servers. Thus, for each incoming query, our broker will consider each search keyword. For each such term and each news server, we compute a weight denoted wt_{ij} for the term t of the collection j as follows:

$$wt_{ij} = \frac{(k_1+1)\cdot df_{tj}}{K + df_{tj}} \cdot \log\left(\frac{C}{cf_t}\right) \quad \text{with } K = k_1 \cdot \left((1-b)+b\cdot {}^{l_j}\!/_{avl}\right)$$

where

- df_{tj} indicates the number of top documents of collection j containing the term t within its title and summary fields;

- cf_t denotes the number of collection returning at least one document containing the term t in its title and summary fields;
- C indicates the number of collections;
- l_j denotes the number of documents returned by the collection j for the current query;
- avl means the average number of documents returned by the collections for the current request;
- k_1 and b are constants and are set respectively to 1.5 and 0.50.

To define a collection score denoted p_j for the jth server, we simply sum over all weights wt_{ij}. However, what is really important is not the absolute value of the collection score p_j but the relative deviation from the mean collection score denoted $\bar{p}$ (computed as the arithmetic mean over all p_j values). Finally, we may exploit this collection score in a new document score. Thus for each document i belonging to the collection j, this new document score value (denoted nw_{ij}) is computed as follows:

$$nw_{ij} = w_{ij} \cdot \left(1 + 0.4 \cdot \frac{(p_j - \bar{p})}{\bar{p}} \right)$$

where w_{ij} indicates the document score computed according to one of our previous document scoring function like document title scoring (SM-TS, see Section 4.2), document summary scoring (SM-SS, see Section 4.3) or one of our combined document scoring approaches (SM-TSS1 or SM-TSS2, see Section 4.4).

In Table 11a, we evaluate four different merging approaches based on the estimated collections statistics, namely SM-ECS-TS (based on the document title), SM-ECS-SS (using the article summary), and our two combined merging models SM-ECS-TSS1 and SM-ECS-TSS2. The retrieval performance depicted in Table 11a shows that we obtain positive improvement for all measures, and the results of the Sign tests exhibited in Table 11b reveal that these differences are always significant.

Table 11a: Performances using estimated collection statistics

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
SM-TS	63.76		61.03		50.56	
SM-ECS-TS	67.15	5.32%	63.55	4.13%	52.66	4.15%
SM-SS	62.21		60.47		50.00	
SM-ECS-SS	64.25	3.28%	62.52	3.39%	50.51	1.02%
SM-TSS1	67.14		64.11		53.83	
SM-ECS-TSS1	68.52	2.06%	65.42	2.04%	54.25	0.78%
SM-TSS2	67.08		63.46		53.74	
SM-ECS-TSS2	68.24	1.73%	64.67	1.91%	53.93	0.35%

Table 11b: Sign test results

SM-TS < SM-ECS-TS
SM-SS < SM-ECS-SS
SM-TSS1 < SM-ECS-TSS1
SM-TSS2 < SM-ECS-TSS2

4.7. Server usefulness

In order to improve the merging process, we may take account of the servers retrieval effectiveness or, more precisely, of their retrieval performance compared to the server mean retrieval performance. Our underlying hypothesis is that we may extract more documents from servers presenting a precision better than the mean precision. We thus take account not only of the presence of pertinent items in the corresponding news collection but of the server's capability to extract relevant items and to present them among the top retrieved documents. When having such a usefulness measure for all servers, we can define a new score for document i belonging to collection j (denoted nw_{ij}) as follows:

$$nw_{ij} = w_{ij} \cdot \left[1 + (0.8 \cdot (u_j - \bar{u})/\bar{u})\right]$$

where u_j indicates the usefulness of server j, $\bar{u}$ corresponds to the mean server usefulness, and w_{ij} the score of document i of collection j. This suggested weighting expression takes account of the document score (w_{ij}) on the one hand, and on the other, of the server capability, compared to the servers mean precision, to extract and present pertinent documents in the top of the retrieved items. Finally, the weighting constants were chosen by tuning experiments.

As a first approach to measure server usefulness, we may consider the server precision as described in Table 5. To define the document score w_{ij}, we may use the document title (SM-Uprec-TS), the document summary (SM-Uprec-SS), and our combined merging model (SM-Uprec-TSS1). The retrieval performances depicted in Table 12a show that we obtain positive improvement (around +4.5%) for all measures when comparing the raw-score merging based on various document logical sections (SM-TS, SM-SS or SM-TSS1) with their corresponding counterpart using the server usefulness (SM-Uprec-TS, SM-Uprec-SS or SM-Uprec-TSS1).

Table 12a: Performances using server usefulness

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
SM-TS	63.76		61.03		50.56	
SM-Uprec-TS	67.28	5.52%	63.74	4.44%	54.25	7.30%
SM-Uprec.ntc-TS	65.00	1.94%	61.78	1.23%	50.70	0.28%
SM-Uprec.oka-TS	65.39	2.56%	61.68	1.07%	51.07	1.01%
SM-SS	62.21		60.47		50.00	
SM-Uprec-SS	65.28	4.93%	62.99	4.17%	51.78	3.56%
SM-Uprec.ntc-SS	63.37	1.86%	61.12	1.07%	49.25	-1.50%
SM-Uprec.oka-SS	63.50	2.07%	61.03	0.93%	49.58	-0.84%
SM-TSS1	67.14		64.11		53.83	
SM-Uprec-TSS1	69.83	4.01%	65.51	2.18%	54.95	2.08%
SM-Uprec.ntc-TSS1	68.59	2.16%	65.89	2.78%	53.04	-1.47%
SM-Uprec.oka-TSS1	68.66	2.26%	65.98	2.92%	53.08	-1.39%

In order to apply this approach, we must have a set of queries with their relevance assessments to define the average precision of each server. However such information is not simple to obtain and requires the definition of the relevance information for all retrieved items. Thus, when not all relevance judgment is available, we may estimate the server usefulness according to Craswell's *et al.* (2000) study. In this case, we will inspect only the first twenty retrieved items to define the relevance assessments. To achieve this, we have downloaded the documents corresponding to the first twenty retrieved items and indexed whole articles using the tf-idf (cosine normalization, denoted CBS-ntc.ntc) on the one hand, and on the other, the Okapi probabilistic model (denoted CBS-oka.bnn, see Section 4.8). Based on this limited relevance information, we may estimate for each server its average precision. To achieve this objective, we consider the top twenty documents of each results list and the usefulness of server j having two

pertinent documents among the top twenty will be fixed to 2 / 20 = 0.1. In our evaluations shown in Table 12a, we used the classical tf-idf approach (denoted SM-Uprec.ntc) and the Okapi probabilistic model (SM-Uprec.oka).

Table 12b: Sign test results

SM-TS < SM-Uprec-TS
SM-TS < SM-Uprec.ntc-TS
SM-TS < SM-Uprec.oka-TS
SM-SS < SM-Uprec-SS
SM-SS < SM-Uprec.ntc-SS
SM-SS = SM-Uprec.oka-SS
SM-TSS1 < SM-Uprec-TSS1
SM-TSS1 < SM-Uprec.ntc-TSS1
SM-TSS1 < SM-Uprec.oka-TSS1

In this table, one can see that even with limited relevance information, the retrieval models based on the server usefulness result in better retrieval performance than their corresponding counterpart ignoring server usefulness (e.g., SM-Uprec.ntc-TS vs. SM-TS, SM-Uprec.oka-SS vs. SM-SS or SM-Uprec.ntc-TSS1 vs. SM-TSS1). As described in Table 12b, these difference are always significant with the exception of SM-SS (0.6221) vs. SM-Uprec.oka-SS (0.6350).

Of course, as indicated in Table 12a, having all relevance information to measure the server usefulness presents a better solution (SM-Uprec) than our estimation based on the first twenty retrieved documents (SM-Uprec.ntc or SM-Uprec.oka).

4.8. Content based scoring (CBS)

Instead of using only one logical part of a document to build a document surrogate, we may download the whole article and index it using the SMART system. In this case, we build a single index for each query using all retrieved items and avoid the merging process. To achieve this, we may choose various vector-processing models or a probabilistic approach.

As a first approach, we adopted a binary indexing scheme within which each document or request is represented by a set of keywords without any weight. To measure the similarity between documents and requests, we count the number of common terms, computed according to the inner product (retrieval model denoted CBS-bnn.bnn in which the three code letters (e.g.,

bnn) indicates the weighting scheme applied to documents and the last three letters the model uses for indexing requests; see Appendix 1 for details).

Binary logical restrictions are often too limiting for document and query indexing. It is not always clear whether or not a document should be indexed by any given term, meaning a simple "yes" nor "no" is insufficient. In order to create something in between, the use of term weighting allows for better term distinction and increases indexing flexibility. As noted previously, the similarity between a document and the request is based on the number of terms they have in common, weighted by the component tf (retrieval model notation: CBS-nnn.nnn).

In a third IR model (Salton, 1989), those terms that do occur very frequently in the collection are not believed to be too helpful in discriminating between relevant and non-relevant items. Thus we might count their frequency in the collection, or more precisely the inverse document frequency (denoted by idf), resulting in a larger weight for sparse words and a smaller weight for more frequent ones. In this case, higher weights are given to terms appearing more often in a document (tf component) and rarely in other articles (idf component). As such, each term does not have an equivalent discrimination power, and a match on a less widely used keyword must therefore be treated as being more valuable than a match on a more common word. Moreover, using a cosine normalization (retrieval model notation: CBS-ntc.ntc), may prove beneficial and each indexing weight may vary within the range of 0 to 1.

Other variants may also be created, especially when considering that the occurrence of a given term in a document is a rare event. Thus, it may be a good practice to give more importance to the first occurrence of this word as compared to any successive, repeating occurrences. Therefore, the tf component may be computed as the log(tf) + 1.0 (retrieval model notation: CBS-ltc.ltc) or as $0.5 + 0.5 \cdot [\text{tf} / \max \text{tf in a document}]$. In this latter case, the normalization procedure is obtained by dividing tf by the maximum tf value for any term in the document (retrieval model denoted "atn"). Different weighting formulae may of course be used for documents and requests, leading to other different weighting combinations (e.g., CBS-atn.ntc or CBS-lnc.ltc).

Finally we should consider that a term's presence in a shorter article provides stronger evidence than it does in a longer document. To account for this, we integrate document length

within the weighting formula, leading to a more complex IR model; for example, the IR model denoted by CBS-Lnu.ltc (Buckley *et al.*, 1996), CBS-dnu.dtn (Singhal *et al.*, 1999) or the Okapi probabilistic search model (Robertson *et al.*, 2000) denoted CBS-oka.bnn. In these schemes a match on a small document will be treated as more valuable than a match on a longer document. The question that then arises is: How will these retrieval models behave when used with our test collection?

Table 13a: Content based scoring

Merging strategies	Average precision	% change	Precision @10	% change	Precision @20	% change
SM-Uprec-TSS1	69.83		65.51		54.95	
CBS-ntc.ntc	73.76	5.63%	70.47	7.57%	57.80	5.19%
CBS-dnu.dtn	73.15	4.75%	69.72	6.43%	57.90	5.37%
CBS-oka.bnn	73.11	4.70%	69.91	6.72%	57.99	5.53%
CBS-Lnu.ltc	72.57	3.92%	69.53	6.14%	57.80	5.19%
CBS-ltc.ltc	72.36	3.62%	69.81	6.56%	58.55	6.55%
CBS-lnc.ltc	72.15	3.32%	68.79	5.01%	57.48	4.60%
CBS-nnn.nnn	68.92	-1.30%	66.64	1.72%	55.61	1.20%
CBS-atn.ntc	63.81	-8.62%	61.78	-5.69%	53.04	-3.48%
CBS-bnn.bnn	52.51	-24.80%	46.45	-29.09%	42.99	-21.77%

Table 13b: Sign test results

SM-Uprec-TSS1 < CBS-ntc.ntc
SM-Uprec-TSS1 < CBS-dnu.dtn
SM-Uprec-TSS1 < CBS-oka.bnn
SM-Uprec-TSS1 < CBS-Lnu.ltc
SM-Uprec-TSS1 < CBS-ltc.ltc
SM-Uprec-TSS1 < CBS-lnc.ltc
SM-Uprec-TSS1 = CBS-nnn.nnn
SM-Uprec-TSS1 > CBS-atn.ntc
SM-Uprec-TSS1 > CBS-bnn.bnn

The retrieval performances shown in Table 13a indicate that when used with an appropriate weighting models, the effectiveness of this approach having a low efficiency (due to downloading and indexing) is better than the effectiveness of the best merging approach presented so far (SM-Uprec-TSS1) which is based only on available data (title, summary, rank

and server usefulness). However, all "CBS-" approaches used only one inverted file and thus we do not need to merge various result lists into a single ranked list of retrieved items.

5. Conclusion

From our study, the following observations can be note:

- In current online news services, the mean percentage of unavailable documents (broken links) is around 4% (see Table 3);
- The average precision of various current online news services varies considerably (mean value of 40.67) and the associated standard deviation is also large (33.96) indicating that there is a large variation among requests (see Table 4 and 5);
- The information available from the servers which could be used in merging / selection is:
  - document rank;
  - document score (sometimes available but these values not comparable across servers);
  - usefulness of the server (calculated according to the retrieval effectiveness of the server);
  - document title (often available but not always present);
  - document summary (rarely available);
  - document date (almost always available);
  - document text (but it requires time to download).
- The raw-score merging strategy presents a better retrieval performance than the round-robin merging approach based on our generic document scoring function when using the document title (+30%, Table 6a), the document summary (+27%, Table 7a) or combining the title and summary of the document (+37%, Table 8a);
- Using the document date when breaking ties (and in favor of more recent articles) presents a small (+2%) but significant improvement in average precision (Tables 10);
- Estimating document frequency based on title and summary fields of the top retrieved items to define a collection weight seem to be enough to obtain better results (+3% improvement on average over the raw-score merging approach, see Table 11a);

- Taking account of the server's usefulness in defining the document score may improve the retrieval effectiveness (around +3%) compared to ranking schemes ignoring this feature (see Table 12);
- Downloading the documents and indexing them with the classical tf-idf vector-space model gives the highest quality results (see Tables 13). However, the main drawback of this strategy is the underlying cost of downloading and indexing the required documents.

Our research shows that in the current news context, it is possible to use low cost merging strategies. As a first attempt, we may consider the round-robin approach which achieves an average precision of 48.97% (Table 6a). Using the title and the summary of the retrieved items, we may apply the raw-score merging procedure which achieves an average precision of 67.14% (Table 10a) representing an improvement of +37%. This approach can still be improved when considering the server usefulness (average precision of 69.83, +4%, see Table 12a) or when downloading all retrieved documents and using the classical tf-idf (cosine normalization) which presents an average precision of 73.76% (see Table 13a). This last approach requires however a larger response time due to the downloading, indexing and retrieving of the documents.

Acknowledgments

This research was supported, in part, by the SNSF (Swiss National Science Foundation) under grant 21-58 813.99 (Y. Rasolofo & J. Savoy).

References

Buckley, C., Singhal, A., Mitra, M. & Salton, G. (1996). New retrieval approaches using SMART. In Proceedings of TREC'4, (pp. 25-48). Gaithersburg: NIST Publication #500-236.

Callan, J. P. (2000). Distributed information retrieval. In W. B. Croft (Ed.), *Advances in information retrieval*, (pp. 127-150). New York: Kluwer Academic Publishers.

Callan, J. P., Lu, Z. & Croft, W. B. (1995). Searching distributed collections with inference networks. In Proceeding of the ACM-SIGIR'95, (pp. 21-28). New York: The ACM Press.

Callan, J. P., Connell, M. & A. Du, A. (1999). Automatic discovery of language models for text databases. In Proceedings of ACM-SIGMOD'99, (pp. 479-490). New York: The ACM Press.

Craswell, N., Hawking, D. & Griffiths, K. (2001). Which search engine is best at finding airline site home pages? Technical report, CSIRO Mathematical and Information Sciences, http://www.ted.cmis.csiro.au/~nickc/pubs/airlines.pdf.

Craswell, N., Bailey, P. & Hawking, D. (2000). Server selection on the World Wide Web. In Proceedings of the ACM-DL'2000, (pp. 37-46). New York: The ACM Press.

Craswell, N., Hawking, D. & Thistlewaite, P. (1999). Merging results from isolated search engines. In Proceedings of the 10th Australian Database Conference, (pp. 189-200). New York: Springer-Verlag.

French, J. C., Powell, A. L., Viles, C. L., Emmitt, T. & Prey, K. J. (1998). Evaluating database selection techniques: A testbed and experiment. In Proceedings of ACM-SIGIR'98, (pp. 121-129). New York: The ACM Press.

Gravano, L., Garcia-Molina, H. & Tomasic, A. (1994). The effectiveness of GlOSS for the text-database discovery problem. In Proceedings of the ACM-SIGMOD 94, (pp. 126-137). New York: The ACM Press.

Gravano, L., Chang, K., Garcia-Molina, H., Lagoze, C. & Paepcke, A. (1997). STARTS - Stanford Protocol Proposal for Internet Retrieval and Search, http://www-db.stanford.edu/~gravano/starts.html.

Hawking, D. & Thistlewaite, P. (1999). Methods for information server selection. *ACM Transactions on Information Systems*, 17(1), 40-76.

Hawking, D., Craswell, N., Bailey, P. & Griffiths, K. (2001a). Measuring search engine quality. *Information Retrieval*, 4(1), 33-59.

Hawking, D., Craswell, N. & Griffiths, K. (2001b). Which search engine is best at finding online services? In Poster Proceedings of the Tenth International World Wide Web Conference, http://www.ted.cmis.csiro.au/~dave/www10poster.pdf.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. In Proceedings of the ACM-SIGIR'93, (pp. 329-338). New York: The ACM Press.

Kirsch S. T. (1997). Distributed search patent. U.S. Patent 5,659,732. http://software.infoseek.com/patents/dist_search/patents.htm.

Lawrence, S. & Lee Giles, C. (1999). Accessibility of information on the Web. *Nature*, 400, 107-109.

Le Calvé, A. & Savoy, J. (2000). Database merging strategy based on logistic regression. *Information Processing & Management*, 36(3), 341-359.

Nielsen, J. (2000). *Designing Web usability: The practice of simplicity*. Indianapolis: New Riders.

Rasolofo Y., Abbaci F., & Savoy, J. (2001). Distributed information retrieval: Approaches to collection selecting and results merging. In Proceedings ACM-CIKM'2001, (pp. 191-198). New York: The ACM Press.

Robertson, S. E., Walker, S. & Beaulieu, M. (2000). Experimentation as a way of life: Okapi at TREC. *Information Processing & Management*, 36(1), 95-108.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Reading: Addison-Wesley.

Salton, G. & McGill, M. J. (1983). *Introduction to modern information retrieval*. New-York: McGraw-Hill.

Savoy, J. (1997). Statistical inference in retrieval effectiveness evaluation. *Information Processing & Management*, 33(4), 495-512.

Savoy, J. & Picard, J. (2001). Retrieval effectiveness on the Web. *Information Processing & Management*, 37(4), 543-569.

Schwartz, M., & Task Force on Bias-Free Language (1995). *Guidelines for bias-free writing*. Bloomington: Indiana University Press.

Selberg, E. & Etzioni, O. (1995). Multi-service search and comparison using the Meta-Crawler. In Proceedings of the Fourth International World Wide Web Conference, http://www.w3.org/Conferences/WWW4/Papers/169.

Selberg, E. & Etzioni, O. (1997). The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert: Intelligent Systems and their Applications*, 12(1), 11-14.

Singhal, A., Choi, J., Hindle, D., Lewis, D. D. & Pereira, F. (1999). AT&T at TREC-7. In Proceedings TREC-7, (pp. 239-251). Gaithersburg, MD: NIST Publication #500-242.

Voorhees, E. M., Gupta, N. K. & Johnson-Laird, B. (1995). Learning collection fusion strategies. In Proceedings of the ACM-SIGIR'95, (pp. 172-179). New York: The ACM Press.

Voorhees, E. M. & Harman, D. (2000). Overview of the sixth text retrieval conference (TREC-6). *Information Processing & Management*, 36(1), 3-35.

Appendix 1. Weighting schemes

To assign an indexing weight w_{ij} that reflects the importance of each single-term j in a document i, we may take three different factors into account. They are represented by the following three code letters respectively:

- within-document term frequency, denoted by tf_{ij} (first letter);
- collection-wide term frequency, denoted by df_j (second letter);
- normalization scheme (third letter).

bnn	$w_{ij} = 1$	nnn	$w_{ij} = tf_{ij}$
ltn	$w_{ij} = (\log(tf_{ij}) + 1) \cdot idf_j$	atn	$w_{ij} = idf_j \cdot [0.5 + 0.5 \cdot tf_{ij} / \max tf_i.]$
oka	$w_{ij} = \left((k_1 + 1) \cdot tf_{ij}\right) / \left(K + tf_{ij}\right)$	dtn	$w_{ij} = \left(\log\left(\log(tf_{ij}) + 1\right) + 1\right) idf_j$
lnc	$w_{ij} = \dfrac{\log(tf_{ij}) + 1}{\sqrt{\sum_{k=1}^{t} \left(\left(\log(tf_{ik}) + 1\right)\right)^2}}$	ntc	$w_{ij} = \dfrac{tf_{ij} \cdot idf_j}{\sqrt{\sum_{k=1}^{t} (tf_{ik} \cdot idf_k)^2}}$
ltc	$w_{ij} = \left(\log(tf_{ij}) + 1\right) idf_j \Big/ \sqrt{\sum_{k=t}^{t} \left(\left(\log(tf_{ik}) + 1\right) \cdot idf_k\right)^2}$		
dtu	$w_{ij} = \dfrac{\left(\left(1 + \log\left(1 + \log(tf_{ij})\right)\right) idf_j / (1 + pivot)\right)}{(1 - slope) \cdot pivot + slope \cdot nt_i}$		
Lnu	$w_{ij} = \dfrac{\left(1 + \ln(tf_{ij}) / (1 + pivot)\right)}{(1 - slope) \cdot pivot + slope \cdot nt_i}$		

Table A.1: Weighting schemes

In Table A.1, document length (the number of indexing terms) of document i is denoted by nt_i, the constant advl is set at 900, the constant b at 0.75, the constant k_1 at 2, the constant pivot at 125 and the constant slope at 0.1. For the Okapi weighting scheme, K represents the ratio between the length of document i measured by l_i (sum of tf_{ij}) and the collection mean noted by advl.

ERRATUM

. Dans l'article "Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections":

- veuillez lire à la quatrième ligne de l'entête « Washington » à la place de « Washigton ».

. Dans l'article "Approaches to Collection Selection and Results Merging for Distributed Information Retrieval":

- veullez lire sur la référence [17] « Powell » à la place de « Powel ».

. Dans l'article "Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching":

- veullez lire à la section 2, première paragraphe « Qantas » à la place de « Quantas »

. Dans l'article "Fusion de collections dans les métamoteurs":

- veullez lire à la dernière ligne de la troisième colonnes de la table 6 « 87.06 %» à la place de « 19.8% ».