Amélioration de la précision dans un système de question-réponse de domaine fermé

Hai Doan-Nguyen, Leila Kosseim

CLaC Laboratory, Department of Computer Science, Concordia University

Montréal, Québec, Canada, H3G-1M8
{haidoan, kosseim}@cs.concordia.ca

Abstract

This paper presents our experiments in constructing a question-answering system which aims at replying questions on services offered by a large company, here Bell Canada. We concentrate on improving the precision of the information retrieval module of the system. Our approach consists in finding a set of special terms which can effectively characterize the relevance of a retrieved candidate to its corresponding question. Combining this set with the information retrieval module has resulted in very good improvements.

Résumé

Ce papier présente nos expériences dans le développement d'un système de question-réponse qui vise à répondre des questions sur les services offerts par une grande compagnie, ici Bell Canada. Nous nous concentrons sur le problème d'amélioration de la précision du module de recherche d'information du système. Notre approche consiste à découvrir un ensemble de termes spéciaux qui peuvent efficacement caractériser la pertinence d'un candidat à la question correspondante. La combinaison de cet ensemble avec le module de recherche d'information a donné de bons résultats.

Mots-clés : information retrieval, question-answering, recherche d'information, système de question-réponse.

1. Introduction

Ce papier présente nos expériences dans le développement d'un système de question-réponse (QR) (anglais, *question-answering*) qui vise à répondre à des questions d'un client sur les services offerts par une grande compagnie, ici Bell Canada. Cette sorte de système est un exemple de la QR de *domaine fermé* (*closed-domain QA*), qui travaille sur une collection de documents restreints quant au sujet et à la quantité. Le domaine a quelques caractéristiques différentes à la QR de domaine ouvert (*open-domain QA*) qui travaille sur une grande collection de documents de sujets divers, y compris le WWW.

Dans la QR de domaine fermé, les réponses correctes ne peuvent normalement être trouvées que dans très peu de documents. Le système n'a donc pas beaucoup de chance, comme dans le cas de la QR de domaine ouvert, de faire la choix dans un ensemble de candidats abondant de réponses correctes. De plus, si le système vise à répondre aux questions d'un client d'une compagnie, il doit être capable d'accepter des questions complexes, et de formes et de styles librement variés. Il doit aussi retourner des réponses complètes, qui peuvent être très longues et complexes, parce qu'il doit clarifier le contexte du problème posé dans la question, expliquer des options de services, donner des instructions ou des procédures, etc. Ces caractéristiques rendent moins utilisables les techniques développées récemment pour la QR de

domaine ouvert, en particulier, avec les compétitions de TREC (Text REtrieval Conference, par exemple (TREC, 2002)). Ces techniques, qui visent à trouver une réponse courte et précise, se base souvent sur l'hypothèse que les questions sont limitées à des mono-phrases, et peuvent être catégorisées (par exemple en PERSONNE, TEMPS, LIEU, etc.).

La QR de domaine fermé a une longue histoire, qui se commence avec des systèmes travaillant sur une base de données (par exemple, BASEBALL (Green *et al.*, 1961) et LUNAR (Wood, 1973)). Une technique très connue à cause de son originalité était les *semantic grammars* (Brown et Burton, 1975), qui préparent *a priori* des patrons de questions d'une tâche spécifique. Malgré sa simplicité, cette technique ne marche que avec des tâches très petites, et un ensemble limité de questions.

Récemment, la recherche de QR fait attention surtout aux problèmes de la QR de domaine ouvert, particulièrement au problème de trouver des réponses très précises et courtes. Pourtant, on commence à reconnaître la nécessité des réponses longues et complexes. Lin *et al.* (2003) font des expériences prouvant que l'utilisateur préfère une réponse dans contexte, par exemple, un paragraphe qui contient la réponse, sans se soucier de la fiabilité de la source. Buchholz et Daelemans (2001) définissent quelques types de réponses complexes, et proposent une solution où le système présente à l'utilisateur une liste de bons candidats et lui laisse la construction de la réponse. Harabagiu *et al.* (2001) abordent les questions qui demandent une réponse en forme d'une énumération (*listing answer*).

Dans ce qui suit, nous présenterons nos expériences dans la construction d'un système de QR pour Bell Canada, en particulier, les efforts pour améliorer la précision du système.

2. Le corpus et l'ensemble de questions

Le corpus sur lequel nous travaillons contient des documents en anglais qui décrivent les services que Bell Canada offre à la clientèle individuelle et d'entreprise. Ils décrivent les services de téléphone standard, de téléphone sans-fil, d'Internet, etc. La collection de documents a été dérivée d'une version Web de Bell Canada (www.bell.ca). Elle comprend plus de 220 fichiers de texte pur, sans balises à l'intérieur. En général, chaque fichier correspond à une page Web, mais certains fichiers correspondent à plusieurs pages. La plupart des documents sont courts, de 1K à 5K caractères, mais il y a aussi de longs documents (max. 24K caractères). Au total, le corpus comprend environ 560K caractères. Comme le corpus était la dérivation au format texte pur des documents formatés (HTML et PDF), beaucoup d'informations importantes sont perdues, comme les titres, sous-titres, tableaux, énumérations, etc. Dans quelques documents, il reste des informations « bruit », par exemple, des titres des liens de navigation génériques du site Web.

L'ensemble de questions dont nous disposons comprend 120 questions. Il est assuré qu'on peut trouver une réponse dans le corpus pour toute question. La forme et le style des questions varient librement. La plupart des questions sont composées d'une phrase, mais il y en a d'autres qui sont composées de plusieurs. Il y a des "Wh-questions", "Yes-No questions", questions sous forme impérative, etc. Les questions peuvent être pour demander la définition ou un détail d'un service, l'existence d'une sorte de service pour un certain besoin, l'instruction pour une opération concernant un service, etc. Voici quelques exemples de questions :

- *What is Business Internet Dial?*
- *Do I have a customized domain name even with the Occasional Plan of Business Internet Dial?*

- *How can our company separate business mobile calls from personal calls?*
- *Please tell me how I can make an auto reply message when using Bell IP Messaging Webmail.*
- *With the Web Live Voice service, is it possible that a visitor activates a call to our company from our web pages, but then the call is connected over normal phone line?*
- *It seems that the First Rate Plan is only good if most of my calls are in the evenings or weekends. If so, is there another plan for long distance calls anytime during the day?*

L'ensemble de questions a été divisé au hasard en deux parties. La première partie de 80 questions serait utilisée pour le but de développement (*training*). La deuxième de 40 questions serait pour le test final.

3. Recherche d'information

Bien que notre corpus ne soit pas volumineux, il n'est pas non plus si petit pour qu'une recherche directe de réponses des questions dans le corpus soit évidente. De plus, la perte des informations formatées comme les titres, sous-titres, etc., rendrait une telle approche plus difficile. Nous concevons donc notre système suivant la stratégie classique de la QR avec deux étapes : (1) recherche d'information, et (2) sélection de candidats et extraction de réponses.

Pour la première étape, nous utilisons Okapi, un moteur de recherche d'information générique bien connu (www.soi.city.ac.uk/~andym/OKAPI-PACK/, aussi Beaulieu *et al.* (1995)). Après l'indexage du corpus, nous passons les questions de l'ensemble de développement à Okapi. Pour chaque question, nous recevons d'Okapi un ensemble de passages extraits des documents du corpus. Okapi donne aussi un point d'évaluation de la pertinence d'un passage à la question, et le nom du fichier qui le contient – chaque fichier de la collection contribue au maximum un passage pour une question donnée. Les points pour les questions de développement se trouvent entre 1 et 40, et sont précises jusqu'au millième de point, par exemple, 10.741.

Les passages retournés par Okapi sont jugés par un humain d'une manière booléenne : correct ou incorrect. Cette sorte de jugement est recommandée dans le contexte de communications entre une compagnie et sa clientèle. Les conditions et les détails techniques de services de la compagnie devraient être rédigés le plus clairement possible dans la réponse au client. Pourtant, nous acceptons quelques tolérances en jugement. Si une question est ambiguë, par exemple, une question concernant des appels téléphoniques qui ne spécifie pas s'il s'agit du service de téléphone standard ou de téléphone sans-fil, toute réponse correcte correspondant à un cas spécifique serait acceptée. Si un passage est bon mais incomplet en tant que réponse, il sera jugé correct si il contient le thème principal de la réponse souhaitable et que l'humain peut trouver des informations manquantes autour de ce passage dans le fichier contenant.

Le tableau 1 donne des statistiques sur la performance d'Okapi exécuté sur l'ensemble de questions de développement. C(n) est le nombre des passages au rang n qui sont jugés corrects. Q(n) est le nombre des questions dans l'ensemble de développement qui ont au moins un passage jugé correct parmi les premiers n passages. En fait, nous n'avons retenu que 10 premiers passages au maximum pour chaque question, parce que après le rang 10, un bon passage semble très rare.

n	1	2	3	4	5	6	7	8	9	10
C(n)	20	11	5	4	9	3	1	1	4	1
%C(n)	25%	13.8%	6.3%	5%	11.3%	3.8%	1.3%	1.3%	5%	1.3%
Q(n)	20	26	28	32	39	41	42	43	44	45
%Q(n)	25%	32.5%	35%	40%	48.8%	51.3%	52.5%	53.8%	55%	56.3%

Tableau 1. Performance d'Okapi sur l'ensemble des questions de développement (80 questions).

L'estimation ci-dessus a montré que les résultats rendus par Okapi ne sont pas satisfaisants. Bien que le taux de 56.3% de questions ayant une réponse correcte parmi les 10 premiers candidats soit acceptable dans l'état de l'art de QR, les taux pour n de 1 à 5 sont un peu faibles. Malheureusement, ce sont des cas qui correspondent aux buts d'application du système. n=1 signifie qu'une seule réponse serait retournée à l'utilisateur – cela correspond à un système QR totalement automatique. n=2 à 5 correspondent aux scénarios plus réalistes des systèmes semi-automatiques, où un agent de la compagnie choisirait une bonne réponse parmi les n passages retournés par le système, le rédigerait et l'enverrait au client. Nous nous arrêtons à n=5, parce que un nombre plus grand de candidats semble psychologiquement trop pour l'agent humain.

Examinant les passages corrects, nous trouvons qu'ils sont en général assez bons pour envoyer à l'utilisateur comme une réponse compréhensible. Environ 25% des passages corrects contiennent des informations superflues à la question correspondante, tandis que 15% manquent des informations. Pourtant, seulement 2/3 parmi ces derniers (soit 10% du total) se montrent difficiles à être complémentés automatiquement. L'extraction d'une réponse d'un bon passage semble alors moins importante que la tâche d'améliorer la précision du module de recherche d'information du système. Dans ce qui suit, nous nous concentrons sur le problème d'augmenter les Q(n), n=1 à 5, du système.

4. Amélioration de la précision du système

L'idée principale ici est de chercher à pousser le plus possible vers les premiers rangs les passages corrects parmi les meilleurs 10 passages que Okapi retourne pour chaque question. Pour ce faire, il faut trouver une information dans les passages qui a la capacité de caractériser la pertinence d'un passage à la question correspondante. Nous notons que les noms des services spécifiques de Bell Canada, comme *'Business Internet Dial'*, *'Web Live Voice'*, etc., peuvent jouer ce rôle. En fait, ces noms de services apparaissent très régulièrement dans presque tous les documents et questions. De plus, un service est souvent présenté ou mentionné dans très peu de documents, ce qui rend ces termes très discriminatoires. Pour être générique, nous les appellerons les *« termes spéciaux »*. Enfin, dans le corpus, ces termes apparaissent sous forme capitalisée, ce qui nous a permit une extraction automatique facile suivie par un filtrage manuel. Nous avons ainsi obtenu plus de 450 termes spéciaux.[1]

4.1. *Première expérience : le rôle des termes spéciaux*

Pour démontrer que les termes spéciaux peuvent aider à améliorer la performance du système, nous avons conçu une série d'expériences. Premièrement, nous cherchons à savoir si ces termes sont capables d'indiquer la pertinence d'un passage à sa question. Un exemple servira la

[1] Okapi permet un type d'indexage où on peut lui fournir une liste de groupes de mots (deux mots ou plus) comme indices, en plus des indices créés automatiquement à partir de mots seuls. En fait, les résultats présentés dans le tableau 1 correspondent à ce type d'indexage, avec ces termes spéciaux fournis à Okapi. Ces résultats sont bien meilleurs que ceux de l'indexage normal.

présentation de l'idée. Pour la question *'Please explain about Web Hosting Managed Services.'*, le passage 4 est le seul passage qui contient le terme spécial *'Web Hosting Managed'* et, en même temps, le seul passage correct.

Nous concevons donc un système de points basé uniquement sur les termes spéciaux et les mots de catégories grammaticales ouvertes (noms, verbes, adjectifs, adverbes) qui apparaissent en commun dans la question et dans le passage. Nous éviterons de présenter les formules compliquées ici, mais le principe est que plus un passage contient des termes – y compris des termes spéciaux, des groupes nominaux, et des mots de classes verbes, adjectifs, et adverbes – en commun avec la question, plus son point est élevé. Les termes spéciaux contribuent des points plus importants que d'autres termes. Si le passage contient plusieurs fois un terme dans la question, ou deux termes différents dans la question, il recevra une prime (lois de synergie). Nous appellerons ce système de points les *points à termes*. Les passages d'Okapi sont maintenant triés par leurs points à termes, et non pas par les points d'Okapi. Les points à termes donnés par la meilleure formule varient entre 0 et 400.

Nous avons essayé quelques formules différentes, et avec des ensembles de paramètres différents. Les résultats semblent meilleurs qu'Okapi, non remarquablement, mais d'une manière encourageante. Le tableau 2 donne les meilleurs résultats que nous avons obtenus.

n	1	2	3	4	5
Q(n) d'Okapi	20	26	28	32	39
Q(n) du système	24	29	32	34	37
Amélioration	4	3	4	2	-2

Tableau 2. Résultats avec les points à termes.

Une analyse des résultats montre que le système de points à termes a en fait poussé beaucoup de passages corrects vers les premiers rangs, par exemple, il a mis 12 nouveaux passages corrects au rang 1. Pourtant, il a aussi abaissé le rang de beaucoup de passages corrects, par exemple, il en a déplacé 8 hors du rang 1, ce qui a rendu une amélioration totale de 4. Ce mauvais réarrangement sont de deux causes : (1) La longueur des passages qu'Okapi retourne n'est pas homogène : beaucoup de passages corrects sont très courts (par exemple, une ligne), tandis qu'il y a de très longs passages incorrects; et (2) Ce système de points ignore totalement les calculs implicitement effectuées par Okapi. Okapi a peut-être choisi un bon document et extrait un bon passage mais qui ne contient pas beaucoup de termes en commun avec la question correspondante.

4.2. *Deuxième expérience : combinaison de points à termes et points d'Okapi*

Nous nous attendions à ce qu'une combinaison de points à termes et points d'Okapi donne une meilleure performance, c'est pourquoi, nous avons utilisé la formule suivante dans la seconde expérience :

(1) Points du passage = Points à termes + CO * Points d'Okapi

où CO *(coefficient pour Okapi)* est un nombre entier qui prend valeur de 0, 1, 2,... jusqu'à MaxCO. Nous avons choisi MaxCO=60, comme à cet ordre, la partie CO * Points d'Okapi est beaucoup plus grand que la partie Points à termes, et les résultats sont pareils aux ceux d'Okapi originalement.

n	1	2	3	4	5
Q(n) d'Okapi	20	26	28	32	39
Q(n) du système	25	31	36	40	43
Amélioration	5	5	8	8	4
%Amélioration	25%	19.2%	28.6%	25%	10.3%

Tableau 3. Résultats avec les points combinés.

Les résultats présentés dans le tableau 3 ont montré une meilleure amélioration en comparaison avec la première expérience, en particulier pour les n=2 à 5. On remarquera que les résultats pour chaque n correspondent à différentes valeurs de CO.

4.3. *Troisième expérience : rôle de rangs de passages*

Analysant les résultats de la seconde expérience, qui est un peu faible pour n=1, nous trouvons que la partie CO * Points d'Okapi n'a pas bien résolu le problème de mauvais réarrangements dans la première expérience. C'est parce que, dans plusieurs cas, les points d'Okapi pour les passages retournés pour une question ne sont pas très distinctifs, par exemple, ils peuvent être différents l'un de l'autre d'à peine quelques dixièmes de points. Dans ces cas-là, même avec un grand CO, c'est la partie de points à termes qui joue le rôle principal, et le système se comporte de façon identique à celui dans la première expérience. Cette analyse nous conduit à donner au rang des passages dans l'ordre retourné par Okapi un rôle dans le calcul de points. Pour ce faire, nous avons modifié la formule (1) en :

(2) Points du passage[i] = CR[i] * Points à termes + CO * Points d'Okapi

où i est le rang du passage dans l'ordre retourné par Okapi, et CR[i] un *coefficient correspondant au rang*. Le problème maintenant est de trouver de bonnes valeurs pour le vecteur CR.

Plusieurs possibilités sont disponibles. Le cas de distribution normales est CR=(1, 1, 1,..., 1). En notant que la distribution de passages correctes (C(n) dans le tableau 1) se concentre dans les premiers rangs (n=1 à 6), on peut penser à un CR=(1, 1, 1, 1, 1, 1, 0.5, 0.5, 0.5, 0.5). Si on veut pousser plus les passages corrects de rang 2 à 6, on peut utiliser CR=(1, 1.5, 1.5, 1.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5). Si on fait attention à la diminution des C(n), on peut proposer CR=(1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1). Si on tient compte de la grandeur relative des C(n) (une perspective probabiliste), on peut penser à CR=(1, 0.55, 0.25, 0.20, 0.45, 0.15, 0.05, 0.05, 0.20, 0.05)[2], etc. En général, on peut construire des CR selon les formes (a), (b), (c) dans la figure 1. La forme (d) donne de mauvais résultats, car elle amplifie les points de passages de bas rangs (de 7 à 10).

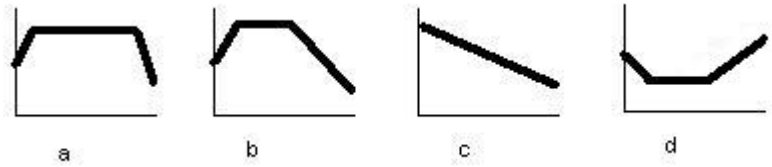

Figure 1. Les distributions possibles pour construire les vecteurs CR.

[2] CR[i]=C(i)/C(1).

Nous avons ainsi construit environ 50 valeurs de CR, et fait fonctionner le système avec ces valeurs et les CO de 0 à 60. Les résultats sont encore meilleurs (tableau 4). Il est intéressant de noter que la meilleur amélioration pour n=1 correspond principalement à la forme (a) de la figure 1 (par exemple, avec CR=(1, 1.2, 1.2,..., 1.2, 0.1), CR=(1, 1.8, 1.8,..., 1.8, 0.1), CR=(1, 3, 3,...,3, 0)). Pour n=2, c'est la forme (b); n=3, la forme (a); n=4, la forme (b); et n=5, toutes les formes (a), (b), (c).

n	1	2	3	4	5
Q(n) d'Okapi	20	26	28	32	39
Q(n) du système	26	33	37	43	43
Amélioration	6	7	9	11	4
%Amélioration	30%	26.9%	32.1%	34.4%	10.3%

Tableau 4. Résultats avec les coefficients de rangs.

4.4. *Quatrième expérience : relation entre la question et le fichier contenant le passage via les termes spéciaux*

Comme la formule (2) n'a pas encore pu résoudre totalement le problème de mauvais réarrangement, dans la quatrième expérience, nous testons la spéculation que si un passage vient d'un fichier qui ne contient aucun terme spécial apparaissant dans la question, ce passage devrait être un « mauvais » candidat, et qu'il faudrait donc le pousser vers les bas rangs, ou même l'éliminer. Cela peut être modélisé en modifiant la formule (2) en :

(3) Points du passage[i] = CT * (CR[i] * Points à termes + CO * Points d'Okapi)

où CT, *coefficient avec termes spéciaux*, sera petit si le passage[i] est « mauvais », et grand si il est « bon ». Nous construisons 20 tels paires de CT pour le test, par exemple (0, 1), (1, 1), (1, 1.5), (1, 2), (1, 2.5), (1, 20), etc. La paire (0, 1) correspond à l'extrémité où tous les mauvais passages sont jetés. Cette fois, nous avons obtenu de très bons résultats (tableau 5). En particulier, les meilleures améliorations pour n=1 et n=2 sont réalisées seulement avec la paire (0, 1). On remarquera que ces meilleures améliorations correspondent aux valeurs non triviales de CO et de CT, ce qui montre que les arguments pour les expériences précédentes restent encore valables. Pour n=3 et n=4, les améliorations ne sont pas les meilleures à la paire (0, 1), mais encore très bonnes à cette paire.

n	1	2	3	4	5
Q(n) d'Okapi	20	26	28	32	39
Q(n) du système	30	38	41	43	44
Amélioration	10	12	13	11	5
%Amélioration	50%	46.2%	46.4%	34.4%	12.8%

Tableau 5. Résultats avec les CT.

4.5. *Test final*

Enfin, nous avons effectué le test final avec les valeurs optimales de CT, CR et CO sur l'ensemble de questions de test (40 questions). Le résultats donnés dans le tableau 6 montrent que le système a fait de très bonnes améliorations avec n=1 et 2, mais moins excellentes avec n=3 à 5. Cela peut s'expliquer par le fait que les passages corrects dans les rangs de 6 à 10 ne

contribuent pas beaucoup à Q(n) : il n'y a que 25-22=3 questions de plus qui reçoivent un passage correcte si on étend n de 5 à 10.

n	1	2	3	4	5	6	7	8	9	10
C(n)	10	6	7	2	3	2	3	2	1	0
Q(n) d'Okapi	10	14	19	20	22	22	24	25	25	25
%Q(n) d'Okapi	25%	35%	47.5%	50%	55%	55%	60%	62.5%	62.5%	62.5%
Q(n) du système	15	19	22	23	23					
Amélioration	5	5	3	3	1					
%Amélioration	50%	36%	16%	15%	5%					

Tableau 6. Résultats du test final.

5. Discussions et conclusions

Dans ce travail, nous avons réalisé des améliorations considérables sur la précision du module de recherche des candidats pour une question. Les termes spéciaux, constitués par les noms de services de Bell Canada, ont joué le rôle principal dans ces améliorations. On peut se demander, pourquoi une telle performance n'avais pas été atteinte par le moteur de recherche d'information (Okapi ici), même si ces termes avaient été entrés au moteur comme termes d'indexage (voir la note en bas de page 1). La raison est peut-être parce que le moteur a traité les termes également comme d'autres termes. En donnant de grands points aux termes spéciaux, ainsi en concevant les coefficients CO, CR, et CT, nous avons rendu le système plus sensible à l'ensemble de termes de travail de l'application.

Le fait que les termes spéciaux ont été extraits facilement dans notre cas, parce qu'ils apparaissent en lettres majuscules dans la collection de documents, ne diminue pas la généralité de l'approche. L'idée principale ici est de trouver une sorte d'information qui peut efficacement caractériser la pertinence d'un candidat à la question correspondante. Cette sorte d'information peut être un ensemble terminologique le plus utilisé dans une application. Il peut être construit manuellement ou (semi-)automatiquement en utilisant de diverses techniques d'extraction.

Pour l'avenir, nous considérerons les approches sémantiques pour notre problème, en remarquant que les noms de services de Bell Canada sont en fait une sorte d'information sémantique. Nous chercherons à savoir si il y a d'autres sortes d'informations sémantiques qui peuvent être utiles, par exemple, le thème de la question, la représentation ontologique des documents, etc.

Remerciements

Ce projet a été financé par les Laboratoires Universitaires Bell (LUB) et le Conseil de Recherche en Sciences Naturelles et Génie du Canada (CRSNG).

Références

Brown J. et Burton R. (1975). Multiple representations of knowledge for tutorial reasoning. In Bobrow et Collins (Eds), *Representation and Understanding*. Academic Press : 311-350.

Buchholz S. et Daelemans W. (2001). Complex Answers: A Case Study using a WWW Question Answering System. *Natural Language Engineering*, Special Issue on Question Answering.

Green W., Chomsky C. et Laugherty K. (1961). BASEBALL: An automatic question answerer. In *Proceedings of the Western Joint Computer Conference* : 219-224.

Harabagiu S., Moldovan D., Pasca M., Surdeanu M., Mihalcea R., Girju R., Rus V., Lactusu F., Morarescu P. et Bunescu R. (2001). Answering Complex, List and Context Questions with LCC's Question-Answering Server. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.

Lin J., Quan D., Sinha V., Bakshi K., Huynh D., Katz B. et Karger D. (2003). The Role of Context in Question Answering Systems. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems (CHI 2003)*, April 2003, Fort Lauderdale, Florida.

Beaulieu M., Gatford M., Huang X., Robertson S.E., Walker S. et Williams P. (1995). Okapi at TREC-3. In Harman D. (Ed.), *Overview of the Third Text REtrieval Conference (TREC-3)*. Gaithersburg, MD: NIST, April 1995.

TREC (2002). *Proceedings of The Eleventh Text Retrieval Conference (TREC 2002)* - NIST Special Publication: SP 500-251, Voorhees E. M. et Buckland Lori P. (Eds).

Woods W.A. (1973). Progress in natural language understanding: An application to lunar geology. In *AFIPS Conference Proceedings*, vol. (42) : 441-450.