TALN

19-22 AVRIL 2004 FÈS (MAROC)

XI[e] Conférence sur le Traitement Automatique des Langues Naturelles

11th Conference on Natural Language Processing

Using Semantic Information to Improve the Performance of a Restricted-Domain Question-Answering System

Hai Doan-Nguyen and Leila Kosseim

CLaC Laboratory – Concordia University
Montreal, Quebec, Canada, H3G-1M8
{haidoan, kosseim}@cs.concordia.ca

Résumé – Abstract

Ce papier présente nos expériences d'utilisation d'informations sémantiques pour améliorer la précision du module de recherche d'information d'un système de question-réponse à domaine restreint. Ce système vise à répondre à des questions sur les services offerts par une grande compagnie. Notre approche consiste à découvrir un ensemble de termes spéciaux et à construire une hiérarchie de concepts, qui peuvent efficacement caractériser la pertinence d'un candidat à la question correspondante. La combinaison de ces informations sémantiques avec le module de recherche d'information a donné de très bons résultats.

This paper presents our experiments in using semantic information to improve the precision of the information retrieval module of a restricted-domain question-answering system. The system aims at replying questions on services offered by a large company. Our approach consists in finding a set of special terms and building a concept hierarchy which can effectively characterize the relevance of a retrieved candidate to its corresponding question. Combining these two kinds of semantic information with the information retrieval module has resulted in very good improvements.

Mots Clés – Keywords

Systèmes de question-réponse, recherche d'information, informations sémantiques.
Question-answering, information retrieval, semantic information.

1 Introduction

This paper presents our experiments in developing a question-answering (QA) system which aims at replying clients' questions on services offered by a large company, here Bell Canada. As it is well-known, the precision of the document retrieval stage is essential to the success of a QA system, because it places an upper bound on the precision of the entire system. Our

approach is to find kinds of semantic information which can effectively characterize the relevance of a retrieved candidate to its corresponding question.

1.1 Restricted-domain QA

Our system is an instance of restricted-domain QA, which works on a document collection restricted in subject and volume. This kind of QA has some characteristics that make it different from open-domain QA, which works over a large document collection, including the WWW. Techniques developed recently for open-domain QA, particularly those within the TREC competitions (e.g. TREC, 2002), may become less helpful to restricted-domain QA. First, in restricted-domain QA, correct answers to a question may often be found in only very few documents; the system does not have a large retrieval set abundant of good candidates for selection. Light & al (2001) give evidences that the performance of a system depends greatly on the redundancy of answer occurrences in the document collection. Second, a restricted-domain QA system often has to work with domain-specific terminology, including domain-specific word meaning. Lexical and semantic techniques based on general lexicons and thesauri, such as WordNet, may not apply well here. Third, if the QA system is to be used for answering questions from a company's clients, it should accept complex questions, of various forms and styles. The system should then return a complete answer, which can be long and complex, because it has to, e.g., clarify the context of the problem posed in the question, explain the options of a service, give instructions or procedures, etc. Contrarily, techniques generally used in TREC competitions, aiming at finding short and precise answers, are often based on the hypothesis that the questions are constituted by a single constituent, and can be categorized into a well-defined and simple semantic classification (e.g. PERSON, TIME, LOCATION, QUANTITY, etc.).

Restricted-domain QA has a long history, beginning with systems working over databases (e.g., BASEBALL (Green et al, 1961) and LUNAR (Woods, 1973)). Recently, research in QA has concentrated mostly on problems of open-domain QA, in particular on how to find a very precise and short answer. Nonetheless, restricted-domain QA seems to regain attention, as shown by a dedicated ACL workshop to be held this year. Researchers also begin to recognize the importance of long and complete answers. Lin & al (2003) carried out experiments showing that users prefer an answer within context, e.g., an answer within its containing paragraph. Buchholz & Daelemans (2001) define some types of complex answers, and propose that the system presents a list of good candidates to the user, and let him construct the reply by himself. Harabagiu & al (2001) mention the class of questions that need an answer in form of an enumeration (*listing answer*).

1.2 Semantic approaches in QA

Use of semantic information in QA systems, or more generally, in any Natural Language Processing applications, has long been studied and continues to be an important research direction. One well-known approach was *semantic grammars* (Brown & Burton, 1975), which build pre-defined patterns of questions for a specific task. Simple and easy to implement, this approach can only deal with very small tasks, and a restricted set of questions. *Latent Semantic Indexing* (Deerwester & al, 1995), a kind of concept indexing,

tries to find underlying or "latent" concepts from interrelationships between words. This technique needs a large volume of data for a machine learning process, hence is not applicable for our project. The most popular class of techniques includes using thesauri and lexicons, classifying documents, and categorizing the questions as mentioned in Section 1.1 above. Harabagiu & al (2000), for example, use WordNet extensively to generate keyword alternations and infer the expected answer category of a question.

Our approach for the Bell Canada QA system consists in finding a set of special terms which characterizes the "working language" of the current task, and building an ontological concept hierarchy which can help better map a question to relevant documents.

2 Corpus and question set

Our working corpus contains documents presenting Bell Canada's wide-range services to personal and enterprise clients: telephones, wireless, Internet, etc. The corpus was constructed by crawling on the company's website (www.bell.ca) and creating one document per Web page. Markups were then removed in order to concentrate our efforts on pure text only. The resulting corpus contains about 560K characters, and comprises more than 220 text documents. Most documents are short, of 1K to 5K characters, some are long, up to 24K. As the corpus is a pure text derivation of formatted documents (HTML and PDF), a lot of important information was missing, like titles, subtitles, listings, tables, etc. In several documents, there is "noisy" information, such as general navigation links of the website.

The available question set has 120 questions. It was assured that every question has an answer from the contents of the corpus. The form and style of the questions vary freely and the questions are rather long (11.3 words on average, compared to 7.3 words for TREC questions). Most questions are composed of one sentence, but 2 questions are other composed of two sentences. There are "Wh-questions", "Yes-No questions", questions in form of an imperative (e.g. "Please tell me how..."), etc. The questions ask about what a service is, its details, whether a service exists for a certain need, how to do something with a service, etc. Below are some examples of questions:

- *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?*
- *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?*

For the project, we divided the question set at random into 80 questions for training and 40 questions for testing. The training set was used to develop the approach and to find the best values of different parameters (see next sections) and the test set was used for the final evaluation.

3 Information retrieval module

Although our corpus is not very large, it is not so small either so that a strategy of searching the answers directly in the corpus could be obvious. Moreover, the missing of formatting information mentioned above would make such an approach more difficult. We therefore design our system following the classic strategy of QA systems, with two steps: (1) information retrieval (IR); and (2) candidate selection and answer extraction.

For the first step, we use Okapi, a well-known generic IR engine (www.soi.city.ac.uk/~andym/OKAPI-PACK/, also Beaulieu & al (1995)). For each question, Okapi returns an ordered list of answer candidates, each of which is composed of one or several consecutive paragraphs of a document. Okapi also gives a relevance score for each candidate and the file name of the document containing it. The candidates are then evaluated by a competent person using a binary scale: correct or incorrect. This kind of judgment is recommended in the context of communications between a company and its clients, because the conditions and technical details of a service should be edited as clearly as possible in the reply to the client. However we did also accept some tolerance in the evaluation. If a question is ambiguous, e.g., it asks about phones but does not specify whether it pertains to wired phones or wireless phones, all correct candidates of either case will be accepted. If a candidate is good but incomplete as a reply, it will be judged correct if it contains the principal theme of the supposed answer, and if missing information can be found in paragraphs around the candidate's text in the containing document.

The table in Figure 1 gives statistics of Okapi's performance on the training question set. C(n) is the number of candidates at rank n which are judged correct. Q(n) is the number of questions in the training set which have at least one correct answer among the first n ranks. We kept only 10 best candidates for each question, because after rank 10 a good answer is very rare (i.e. %C(n) < 5%).

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%

Figure 1 : Performance of Okapi on the training question set (80 questions).

As for answer redundancy, among the 45 questions having at least a correct answer (see Q(10)), there are 33 questions (= 41.3% of the entire training set) having exactly 1 correct answer, 10 questions (12.5%) having 2 , and 2 questions (2.5%) having 3 correct answers.

The results show that Okapi's performance is not satisfying, conforming to our discussion about characteristics of restricted-domain QA above. The system's precision is particularly weak for n's from 1 to 5. Unfortunately, these are cases that the system aims at. n=1 means that only one answer will be returned to the user – this corresponds to a totally automatic system. n=2 to 5 correspond to more practical scenarios of a semi-automatic system, where

an agent of the company chooses the best one among the n candidates, edits it, and sends it to the client. We stop at n=5 because a greater number of candidates seems too heavy psychologically to the human agent. Also note that the rank of the returned candidates is not important here, because they would be equally examined by the human agent. This explains why we use Q(n) to measure the precision performance of the system rather than other well-known scoring such as mean reciprocal rank (MRR).

Examining the correct candidates, we find that they are generally good enough to be sent to the user as an understandable reply. About 25% of them contain superfluous information for the corresponding question, while 15% are lacking of information. However, only 2/3 of the latter (that is 10% of all) look difficult to be completed automatically. Extracting the answer from a good candidate therefore seems less important than improving the precision of the IR module. In the following, we concentrate on how to improve Q(n), n= 1 to 5, of the system.

4 Improving the precision of the system with special terms

Our main idea here is to try to push the correct candidates among the 10 best candidates kept for a question to the first ranks as much as possible. To do this, it is necessary to find some kind of information which could effectively characterize the relevance of a candidate to its corresponding question. We note that the names of specific Bell services could be helpful here, because they occur very often in almost every document and question, and a service is often presented or mentioned in only one or a few documents – making these terms very discriminating. To have a generic concept, we will call these names *'special terms'*. Finally, in the corpus, these special terms occur normally in capital letters, e.g. *'Business Internet Dial'*, *'Web Live Voice*', etc., and can easily be extracted automatically. After a manual filtering, we obtained more than 450 special terms.

In (Doan-Nguyen & Kosseim, 2004), we obtained very good improvements by re-ranking Okapi's candidates using a new scoring system which raises the score of candidates containing occurences of special terms found in the corresponding question, as follows:

(1) Score_of_candidate[i] = DC * (OW * Okapi_score + RC[i] * Term_score)

According to the formula, the score of candidate i in the ranked list returned by Okapi is a function of 3 factors: (1) The original **Okapi_score** given by Okapi, weighted by some integer value **OW**. (2) A **Term_score** that measures the importance of common occurrences of special terms in the question and the candidate. It is weighted by some integer value **RC[i]** (for *rank coefficient*) that represents the role of the relative ranking of Okapi. (3) A *document coefficient* **DC** that indicates the relative importance of a candidate i coming or not coming from a document which contains a special term occurring in the question. DC is thus represented by a 2-value pair; e.g., the pair (1, 0) corresponds to the extreme case of keeping only candidates coming from a document which contains at least one special term in the question, and throwing all others.

The table in Figure 2 gives the best results obtained when running the system with 20 different values of **DC**, 50 of **RC**, and **OW** from 0 to 60, on the training question set. See (Doan-Nguyen & Kosseim, 2004) for how to design these values. The table in Figure 3 gives the results of running the system with optimal training parameters on the test question sets.

n	1	2	3	4	5
Q(n) of Okapi	20	26	28	32	39
Q(n) of system	30	38	41	43	44
Improvement	10	12	13	11	5
%Improvement	50%	46.2%	46.4%	34.4%	12.8%

Figure 2: Best results of scoring function (1) on the training set (80 questions).

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) of Okapi	10	14	19	20	22	22	24	25	25	25
%Q(n) of Okapi	25%	35%	47.5%	50%	55%	55%	60%	62.5%	62.5%	62.5%
Q(n) of system	15	19	22	23	23					
Improvement	5	5	3	3	1					
%Improvement	50%	36%	16%	15%	5%					

Figure 3: Results of scoring function (1) on the test set (40 questions).

5 Improving precision with a concept hierarchy

In another effort to find a semantic characteristic of a document, we try to map the documents into a hierarchy of concepts. Each document corresponds to a set of concepts, and a concept corresponds to a set of documents. Building such a concept hierarchy seems feasible within closed-domain applications, because the domain of the document collection is pre-defined, the number of documents is in a controlled range, and the documents are often already classified topically, e.g. by its creator. If no such classification existed, one can make use of techniques of building hierarchies of clusters (e.g. those summarized in Kowalski (1997)). However they are painstaking in programming and may not be very precise.

5.1 Building the concept hierarchy

We are lucky finding that the web page URL of a document can help here. Although the URLs are complex, they contain a portion which can be used to construct the concept hierarchy and the mapping. For example, below are the URLs of documents presenting the long distance (wired) phone service, and its 'First Rate' plans, respectively. The portions in bold characters are useful portions:

– *http://www.bell.ca/shop/application/commercewf?origin=noorigin.jsp&event=link(goto)&**content=/jsp/content/personal/catalog/phoneservices/long_distance/index.jsp**&REF=HP_PERS*

– *http://www.bell.ca/shop/application/commercewf?origin=noorigin.jsp&event=link(goto)&**content=/jsp/content/personal/catalog/phoneservices/long_distance/first_rate/index.jsp***

After some string manipulations, including removing parts like '/jsp/content/', 'catalog', 'index.jsp', etc., we obtain labels which can be used as the characterizing topic or *concept* of a document, e.g., `Personal-Phone-LongDistance`, `Personal-Phone-LongDistance-FirstRate`, etc. Note that although many concepts coincide in fact with a special term, e.g. 'First Rate', many other are not special terms, e.g. 'phone', 'wireless', 'long distance', etc. Now it is easy to build a concept hierarchy from the labels: one which is a prefix of another will be its parent concept. After some manual editing, we achieve a nice concept hierarchy and mapping between it and the document collection. Below is a small excerpt from the hierarchy:

```
!Bellroot!
      Personal
            Personal-Phone
                  Personal-Phone-LongDistance
                        Personal-Phone-LongDistance-BasicRate
                        Personal-Phone-LongDistance-FirstRate
```

5.2 Mapping from questions to documents via concepts

The use of the concept hierarchy in the QA system is based on the following assumption: *A question can be well understood only when we can recognize the concepts implicit in it*. For example, the concepts in the question:

> *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?*

include `Personal-Phone-LongDistance` and `Personal-Phone-LongDistance-FirstRate`. Once the concepts are recognized, we can determine a small set of documents relevant to these concepts, and carry out the search of answers in this set.

To map a question to the concept hierarchy, we postulate another assumption, *that the question should contain words expressing the concepts*. These words may be those constituting the concepts, e.g., "long", "distance", "first", "rate", etc., or synonyms/near synonyms of them, e.g., "telephone" to "phone"; "mobile", "cellphone" to "wireless". Because the questions are very varying in form, we used a bag of word approach. For every concept, we build a bag of words which make up the concept, e.g., for the concept `Personal-Phone-LongDistance-FirstRate`, the word bag is {"personal", "phone", "long", "distance", "first", "rate"}. We also build manually a small lexicon of (near) synonyms as mentioned above.

Now, a question will be analyzed into separate words, and we look for concepts whose word bags have elements in common with them. A concept is judged more relevant to a question if: (1) its word bag has more elements (in stemmed form) in common with the question's words; (2) the (percentage) quotient of the subset in (1) over the entire word bag is higher; and (3) there are more occurrences of words in that subset in the question. For the example question above, here are the first 6 relevant concepts in descending order (the numbers indicate their rank):

```
Personal-Phone-LongDistance-FirstRate             1
Personal-Phone-LongDistance-FirstRateOverseas     2
Personal-Phone-LongDistance-FirstRate24           2
Business-Voice-LongDistance-CallingCard           4
Business-Voice-LongDistance-SavingsPlan           4
Business-Voice-LongDistance-PerCall               4
```

From the relevant concept sets, it is straightforward to derive the relevant document set for a given question. The documents will also be ranked according to the order of the deriving concepts. (If a document is derived from several concepts, the highest rank will be used.) As for the coverage of the mapping, there are only 4 questions in the training set and 3 in the test set having an empty relevant document set. In fact, these questions seem to need a context to be understood, e.g., a question like *'What does Dot org mean?'* should be posed in a conversation about Internet services. Roughly speaking, the mapping can cover about 94% (=113/120) of the questions.

5.3 Improving precision via concepts

To determine whether the list of documents derived for a question as described above makes any difference, we use a formula similar to (1), but rather than using **DC**, we use **CC**, a *concept-related coefficient* that measures to what degree a document occurs in the concept-derived list. This can be formulated as:

(2) Score of candidate[i] = CC * (OW * Okapi_score + RC[i] * Term_score + 1)

The value of **CC** depends on the document that provides the candidate[i]. **CC** should be high if the rank of the document is high, e.g. **CC**=1 if rank=1, **CC**=0.9 if rank=2, **CC**=0.8 if rank=3, etc. If the document does not occur in the concept-derived list, its **CC** should be small, e.g. 0. We run the system with 14 different values of the **CC** vector, with **CC** for rank 1 varying from 1 to 7, and **CC** for other ranks decreasing accordingly. Values for other coefficients are the same as in the previous experiment using (1). We obtain the results shown in Figure 4, which are slightly better than the previous experiment (cf. Figure 2). This shows that our approach of using concepts is appropriate. It also suggests us to combine **CC** and **DC** in the next experiment.

n	**1**	**2**	**3**	**4**	**5**
Q(n) of Okapi	20	26	28	32	39
Q(n) of system	31	39	41	41	42
Improvement	11	13	13	9	3
%Improvement	55%	50%	46.4%	28.1%	7.7%

Figure 4: Best results with concept-related coefficient **CC** on the training set

5.4 Combining document-related scoring and concept-related scoring

In the last experiment, we try to combine the document coefficient used in the scoring function (1) and the concept coefficient of function (2). We make a simple combination of **CC** and **DC** as below:

(3) Score of candidate[i] = (CC + DC) * (OW * Okapi_score + RC[i] * Term_score + 1)

This time, the results (Figure 5) are better than when using **DC** or **CC** alone. This shows that special terms and concepts are complementary methods, which yield very good improvements when combined.

n	1	2	3	4	5
Q(n) of Okapi	20	26	28	32	39
Q(n) of system	33	42	43	44	44
Improvement	13	16	15	12	5
%Improvement	65%	61.5%	53.6%	37.5%	12.8%

Figure 5: Best results with **DC** and **CC** combined on the training set.

5.5 Final test

Finally, we carry out the final test with the optimal values of **CC**, **DC**, **RC**, and **OW** on the test question set. The results in Figure 6 show that the current system outperforms Okapi, and is also uniformly better than when we use only special terms (see Figure 3).

n	1	2	3	4	5
Q(n) of Okapi	10	14	19	20	22
%Q(n) of Okapi	25%	35%	47.5%	50%	55%
Q(n) of system	21	23	23	24	24
Improvement	11	9	4	4	2
%Improvement	110%	64.3%	21%	20%	9%

Figure 6: Results of the final test using formula (3) on the test set (40 questions).

6 Discussion and Conclusions

Precision performance is a hard issue for restricted-domain QA, due to data scarcity. In this work, we have made considerable improvements on the precision of the IR module of the QA system. Special terms, constituted by service names of Bell Canada, and the concept hierarchy, built from the company's original document classification, play a major role in these improvements. They are complementary information sources which make the QA system much more sensitive to the working language of the application than a general IR approach.

We believe that the methodology and conclusions drawn in this study are transferable to other domains and corpora. The main idea here is to find some kinds of semantic information which can efficiently characterize the relevance of a candidate to the corresponding question. One such kind of information can be the essential terminological set used in the interested task In our case, the extraction of special terms was simplified thanks to the systematic use of capital letters in the corpus. In another application, these resources can be constructed by hand or (semi-)automatically using different term extraction techniques. As for the concept hierarchy, it is reasonable to hypothesize that some classification exists for a domain-specific

document set. However, our work is not about how to construct such a concept hierarchy, but rather how to apply it in finding correct answers.

Acknowledgements

This project was funded by Bell University Laboratories and NSERC. The authors would like to thank the anonymous referees for their comments.

References

BEAULIEU M. ET AL.. (1995). Okapi at TREC-3. In: *Overview of the Third Text REtrieval Conference (TREC-3)*. Edited by D. K. Harman. Gaithersburg, MD: NIST, April 1995.

BROWN J., BURTON R. (1975). Multiple representations of knowledge for tutorial reasoning. In Bobrow & Collins (Eds), *Representation and Understanding*. Academic Press, New York.

BUCHHOLZ S., DAELEMANS W. (2001). Complex Answers: A Case Study using a WWW Question Answering System. *Natural Language Engineering*, 7(4), 2001.

DEERWESTER S., DUMAIS S., FURNAS G., LANDAUER T., HARSHMAN R. (1990). Indexing by latent semantic analysis. *Journal of American Society of Information Science, 41,* 391-407.

DOAN-NGUYEN H., KOSSEIM L. (2004). Amélioration de la précision dans un système de question-réponse de domaine fermé. To appear. *Proceedings of 7es Journées internationales d'Analyse statistique des Données Textuelles (JADT)*. Louvain-la-Neuve, Belgium.

GREEN W., CHOMSKY C., LAUGHERTY K. (1961). BASEBALL: An automatic question answerer. *Proceedings of the Western Joint Computer Conference*, pp. 219-224.

HARABAGIU S. ET AL. (2000). FALCON: Boosting Knowledge for Answer Engines. *Proceedings of the Ninth Text REtrieval Conference (TREC).*

HARABAGIU S. ET AL. (2001). Answering Complex, List and Context Questions with LCC's Question-Answering Server. *Proceedings of the 10th Text REtrieval Conference (TREC).*

KOWALSKI G. (1997). *Information Retrieval Systems – Theory and Implementation*. Kluwer Academic Publishers, Boston/Dordrecht/London.

LIGHT M., MANN G., RILOFF E., BRECK E. (2001). Analyses for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4), 2001.

LIN J., QUAN D., SINHA V., BAKSHI K., HUYNH D., KATZ B., KARGER D. (2003). The Role of Context in Question Answering Systems. *Proceedings of the 2003 Conference on Human Factors in Computing Systems (CHI 2003),* April 2003, Fort Lauderdale, Florida.

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

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