Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 66 (2008) 53–67
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

# Improving the performance of question answering with semantically equivalent answer patterns

Leila Kosseim *, Jamileh Yousefi

*CLaC Laboratory, Department of Computer Science and Software Engineering, Concordia University, 1400 de Maisonneuve Blvd., West Montreal, Quebec, Canada H3G 1M8*

Available online 11 September 2007

**Abstract**

In this paper, we discuss a novel technique based on semantic constraints to improve the performance and portability of a reformulation-based question answering system. First, we present a method for acquiring semantic-based reformulations automatically. The goal is to generate patterns from sentences retrieved from the Web based on lexical, syntactic and semantic constraints. Once these constraints have been defined, we present a method to evaluate and re-rank candidate answers that satisfy these constraints using redundancy. The two approaches have been evaluated independently and in combination. The evaluation on 493 questions from TREC-11 shows that the automatically acquired semantic patterns increase the MRR by 26%, the re-ranking using semantic redundancy increases the MRR by 67%, and the two approaches combined increase the MRR by 73%. This new technique allows us to avoid the manual work of formulating semantically equivalent reformulations; while still increasing performance.


*Keywords:* Natural language processing; Question answering; Semantic relations; Question reformulations

## 1. Introduction

Today, an overwhelming quantity of textual information is available in electronic form. This makes the development of semi-automatic tools to mine the content of such documents a necessity. To answer this need, the natural language processing (NLP) community has harnessed together formerly independent technologies in order to build more sophisticated content-based systems such as information retrieval engines, text summarizers, and question answering systems.

In this paper, we present a technique that uses semantic constraints to improve a reformulation-based question answering system. The objective of the research was to automatically acquire answer extraction patterns with semantic constraints that perform as well as manually created ones. We show how we generate these patterns from sentences retrieved from the Web based on lexical, syntactic and semantic constraints. Once these

* Corresponding author.
*E-mail addresses:* kosseim@cse.concordia.ca (L. Kosseim), j_yousef@cse.concordia.ca (J. Yousefi).
*URLs:* www.cse.concordi.ca/~kosseim (L. Kosseim), www.cse.concordia.ca/~j_yousef (J. Yousefi).

0169-023X/$ - see front matter 
doi:10.1016/j.datak.2007.07.010

Table 1
Examples of factoid questions from the TREC–QA track (TREC-8–11)

| TREC # | Q # | Question |
|---|---|---|
| TREC-8 | Q-1 | *Who is the author of the book, ''The Iron Lady: A Biography of Margaret Thatcher''?* |
| TREC-8 | Q-2 | *What was the monetary value of the Nobel Peace Prize in 1989?* |
| TREC-8 | Q-3 | *What does the Peugeot company manufacture?* |
| TREC-9 | Q-201 | *What was the name of the first Russian astronaut to do a spacewalk?* |
| TREC-9 | Q-202 | *Where is Belize located?* |
| TREC-9 | Q-203 | *How much folic acid should an expectant mother get daily?* |
| TREC-10 | Q-894 | *How far is it from Denver to Aspen?* |
| TREC-10 | Q-895 | *What county is Modesto, California in?* |
| TREC-10 | Q-896 | *Who was Galileo?* |
| TREC-11 | Q-1394 | *In what country did the game of croquet originate?* |
| TREC-11 | Q-1395 | *Who is Tom Cruise married to?* |
| TREC-11 | Q-1396 | *What is the name of the volcano that destroyed the ancient city of Pompeii?* |

constraints have been defined, we present a method to evaluate and re-rank candidate answers that satisfy these constraints using redundancy. The two approaches have been evaluated independently and in combination. The evaluation on 493 questions from TREC-11 shows that the automatically acquired patterns increase the precision by 16% and the MRR by 26%, the re-ranking using semantic redundancy increases the MRR by 67%, and the two approaches combined increase the precision by 28% and the MRR by 73%. This new technique allows us to avoid the manual work of formulating semantically equivalent reformulations; while still increasing performance.

This paper is organized as follows: Section 2 reviews previous work in the area. We present both general issues in question answering, and specific work on reformulation-based question answering. Section 3 presents the manual formulations that we used for comparison, then Section 4 presents in detail how we generate the semantic patterns and how we evaluated them. Section 5 then discusses how the patterns can be used to re-rank the candidate answers and again, improve the performance of the system. Section 6 finally evaluates the use of the generated patterns and the semantic re-ranking together. A discussion and a conclusion are then presented in Section 7.

## 2. Related work

### 2.1. Question answering

Information Retrieval (IR) systems are designed to find documents that satisfy user's information needs in large document collections (e.g. the Web). Given a bag of keywords, an IR system will retrieve the most relevant documents. However, as [1] puts it, IR is better described as *document retrieval*, rather than *information retrieval*. To approach *true* information retrieval, in 1998, the National Institute of Standards and Technology (NIST) have added a new Question Answering (QA) track to their competition-style Text REtrieval Conference (TREC) [2]. In this task, systems are given a set of factoid questions (see Table 1 for some examples) and a 3 GB document collection and are required to return anything from a 250-byte text snippet to the exact answer to the question extracted from the document collection. For example, given the question, *Who founded American Red Cross?* a QA system will search the document collection and extract the answer *Clara Barton*, instead of returning an entire document. Thus, the user is not responsible anymore of analyzing the content of returned documents to find an answer; it is the QA system's responsibility to extract the correct answer from the retrieved documents.

Following TREC–QA, standard measure to evaluate QA systems is the mean reciprocal rank (MRR).[1] For each question, a QA system will return a ranked list of candidate answers. The score of the system for a question $q$ is computed as the reciprocal rank of the first correct answer ($RR(q)$). If the candidate list contains the

[1] Other measures include the confidence weighted score and the accuracy.

correct answer, the score is equal to the reciprocal of its rank; otherwise, the score is zero. For example, if a question generates a list of five candidates and the first correct answer is at rank 3, the question receives a $RR(q)$ score of $\frac{1}{3}$. The overall system score is the mean of $RR(q)$ for all questions $q$ [3].

Since 1998, the field of QA has received a lot of interest both in the research community and in the industry; and today QA has become a major application of NLP. Over the years, the TREC–QA task has attracted more and more teams and the requirements have become more and more challenging. Novel techniques have been developed to answer these new challenges, ranging from purely statistical techniques using redundancy of answers from the Web (e.g. [4]), to syntactic-based analysis of the documents (e.g. [5]), to attempts at semantic representations (e.g. [6]). Harabagiu et al. [1] offers an up-to-date overview of the techniques used in open-domain QA and Voorhees and Harman [7] presents a complete historical perspective of the TREC competitions.

### 2.2. Reformulation-based QA

A standard technique used in QA working on large document collections such as the Web is to use question reformulations (also referred to as *surface patterns*, *paraphrases*, *re-write rules*, and *answer patterns*). The technique is based on identifying various ways of expressing an answer context given a natural language question. For example given the question *Who founded American Red Cross?*, a reformulation-based QA system will search for formulations like *the founder of the American Red Cross is* ⟨NP⟩ or ⟨NP⟩, *the founder of the American Red Cross* in the document collection and will instantiate ⟨NP⟩ with the matching noun phrase. The ideal reformulation should impose constraints on the answer so as not to retrieve incorrect answers (e.g. *the founder of the American Red Cross is a nut lover*) but should also identify many candidate answers to increase the system's confidence in them.

Most work on reformulations has used patterns based on lexical, syntactic or named entity features (e.g. person-name, organization, etc.). However, only a few studies have worked on semantically equivalent reformulations such as ⟨NP⟩, *also known as the founder of the American Red Cross* or *the creator of the American chapter of the Red Cross organization is* ⟨NP⟩. We believe that stronger semantic constraints can be beneficial in finding a more precise set of candidate answers. However writing semantic reformulations by hand is a labor-intensive and tedious task. Our goal is to learn semantically equivalent reformulation patterns automatically from natural language questions and use these constraints to re-rank our candidate answers to improve the performance of a QA system.

### 2.3. Related work on reformulation acquisition

In 2001, Soubbotin et al. [8] along with Brill et al. [9] were among the first to use reformulation patterns as the core of their QA system. This approach searches the document collection for predefined surface patterns or exact sentences that could be the formulation of the potential answer. Soubbotin et al. [8] wrote their patterns by hand and were among the best scoring team at the TREC-10 QA track [10]. Their work shows that if enough human resources are available, hand-crafted rules can produce excellent results. On the other hand, Brill et al. [9] generated patterns automatically by using simple word permutations to produce paraphrases of the question. By permutating words of the questions like simple tokens, they generated a large set of reformulations. Most were ungrammatical, but since they were mapped onto a large document collection (the Web), ungrammatical permutations will, in principle, retrieve nothing, and grammatical and fluent contexts will retrieve candidates that can be ranked by frequency.

Given the success of these first attempts, much progress has then been made to acquire reformulations automatically; and to this day, the use of reformulations is still a important technique used in QA (e.g. [11,12]) whether they are created manually or learned automatically. More recently, Aceves-Prez et al. [13] tried to use simple word permutations and verb movements to generate paraphrases for their multilingual QA system. In the work of [14–16], answer formulations are produced for query expansion to improve information retrieval. While in [16] reformulation rules to transform a question of the form *What is X?* into *X is* or *X refers to* are built by hand, Agichtein et al. [15] and Agichtein and Gravano [14] learns to transform natural language questions into sets of effective search engine queries, optimized specifically for each search engine.

Ravichandran et al. [17] use a machine learning technique and a few hand-crafted examples of question–answer pairs to automatically learn patterns along with a confidence score. However, the patterns do not contain semantic information. They include specific strings of words such as *was born on*, *was born in*, ... with no generalization of the `is-born` relation. Hermjakob et al. [18] do use semantic paraphrases, called *phrasal synonyms*, to enhance their TextMap QA system. However, many of these patterns are manual generalizations of patterns derived automatically by [17].

Kwok et al. [19] use transformational grammar to perform syntactic modifications such as Subject–Aux and Subject–Verb movements. Radev et al. [20] learns the best query reformulations (or paraphrases) for their probabilistic QA system. Here again, the paraphrases are syntactic variations of the original question.

Duclaye et al. [21], however, do try to learn semantically equivalent reformulations by using the web as a linguistic resource. They start with one single prototypical argument tuple of a given semantic relation and search for potential alternative formulations of the relation, then find new potential argument tuples and iterate this process to progressively validate the candidate formulations.

More recently, Mollà [22] tries to learn how to answer a question based on matching a graph representation of the question with a graph representation of the answer. His system translates questions and potential answer sentences into a graph-based logical form representation (inspired by Sowa [23] conceptual graphs) and applies operations based on graph overlaps to compute their similarity. Because a logical form is used, the semantics of the questions and the candidate sentences are taken into account, as opposed to only lexical or syntactic patterns. However, the method does not generalise over relations, because relations express syntactic or semantic relations that they prefer not to over-generalise. In our approach, we do not attempt to form a proper semantic representation of the questions and the candidate sentences. Because generating such a representation (e.g. [22]'s graphs) is a difficult and error-prone task, instead, we generate patterns that include lexical and syntactic constraints, but impose a semantic constraint on the main verbs of the questions and the candidate answers. To avoid over-generalization, we weigh each pattern according to its semantic similarity to the semantic relation that it carries as computed by WordNet.

In addition to work in QA, much work in the automatic acquisition of reformulation patterns has been done in the context of information extractions. However, here again, semantic features are often reduced to named entities. Stevenson and Greenwood [24], for example, learns patterns and ranks them using the standard vector space model. Patterns consist of predicate–argument structures. The subject, verb, object (SVO) structure of a clause is mapped to a predicate–argument structure and the pattern fillers can be either a lexical item or semantic category (named entity).

### 2.4. Summary of previous work

As we presented in the previous paragraphs, pattern induction is typically based on lexical or syntactic features. When searching a huge document collection such as the Web, having only lexical and syntactic reformulations may be enough because the collection exhibits a lot of redundancy. However, we believe that in a smaller collection, semantic reformulations are necessary.

Work in pattern acquisition that includes semantic features, usually only takes into account named entities. Mollà [22] does try to learn semantic reformations, using a more complete semantic representation of the questions and answer sentences, but does not impose semantic constraints on the relations. Compared to Mollà [22], our work uses a more flexible and easy to compute question and answer sentence representations, yet imposes semantic constraints on the main verbs. We do not force a semantic match on the concepts of the sentences (only lexical, syntactic and named entity tags are used), but we do force a semantic match on the relations using WordNet's hyponym/hypernym hierarchy.

## 3. The manual patterns

Our work builds on our current reformulation-based QA system called QUANTUM [25,26], where reformulations were hand-crafted and only relied on named entities for semantic constraints.

Given a question, QUANTUM needs to identify which answer pattern to look for. It therefore uses two types of patterns: *a question pattern* that defines what the question must look like, and a set of *answer patterns* to be looked

for in the document collection. An answer pattern specifies the form of sentences that may contain a possible candidate answer. For example, the question *Who is George Bush?* will be matched to the question pattern `Who Vsf PERSON?` which will trigger the search for any one of these answer patterns in the document collection:

⟨QT⟩ ⟨Vsf⟩ ⟨ANSWER⟩
⟨ANSWER⟩ ⟨Vsf⟩ by ⟨QT⟩

where ⟨ANSWER⟩ is the candidate answer, ⟨QT⟩ is the question term (i.e. *George Bush*), and ⟨Vsf⟩ is the verb in simple form.

To develop the patterns, we used the 893 questions of TREC-8 & 9 as training set. In total, 77 formulation templates were created manually, covering 90% of the questions of the training set. By coverage, we mean that at least one formulation template is applicable for a question. In the current implementation, both question and answer patterns are based on named entity tags (e.g. PERSON), part-of-speech tags (e.g. Vsf), tags on strings (e.g. QT, ANY-SEQUENCE-WORDS) and specific keywords (e.g. Who, by). The templates generate 1638 actual answer formulations for the TREC-8 & 9 questions that are covered. So, on average, two answer formulations are produced per question.

These hand-made patterns will be used later for comparison in order to evaluate the patterns that we learned automatically.

## 4. Generating semantically equivalent patterns

### 4.1. Overall methodology

To generate semantically equivalent answer contexts, we try to find sentences from the Web that contain the correct answer and try to generalize them into syntactico–semantic patterns. To do so, we first use a training corpus of question–answer pairs from which we learn how to generalize each type of question. Each question–answer pair is analyzed to extract its expected answer type, its arguments and its semantic relation. We then search the Web for sentences containing the arguments and the semantic relation and finally, we pass the sentences through a part-of-speech tagger and a noun phrase chunker to generalize them.

More specifically, each question–answer pair of the training corpus is processed to:

(1) Extract the main arguments from the question.
(2) Extract the main arguments from the answer.
(3) Extract the relation between the question arguments and the answer arguments. The semantic constraint will be defined on this relation.
(4) Generate syntactico–semantic patterns for the answer sentences.
    (a) Construct a search query using the question and answer arguments.
    (b) Submit the query to a Web search engine.
    (c) Filter sentences from the retrieved documents containing the question and answer arguments.
    (d) Filter the remaining sentences using the semantic constraints.
    (e) Eliminate redundant sentences.
    (f) Identify the main syntactic constituents of the sentences.
    (g) Construct a corresponding syntactico–semantic pattern.
    (h) Weigh these patterns based on their frequency, their semantic similarity to the question and other features.

Let us now explain each step in details.

### 4.2. The training corpus

The training corpus consists of 1343 question–answer pairs taken from the TREC-8, TREC-9, and TREC-10 collection data [2,27,10]. Each question–answer pair is composed of one question and its corresponding answer. For example:

Q: *Where is the actress, Marion Davies, buried?* A: *Hollywood Memorial Park*
Q: *When did Nixon die?* A: *April 22, 1994*
Q: *Who is the prime minister of Australia?* A: *Paul Keating*

We divided the training corpus according to the question type. We used the classification used in [28] to categorize questions into seven main classes (what, who, how, where, when, which, why) and 20 subclasses (e.g. what-who, who-person, how-many, how-long, etc.). Fig. 1 shows the proportion of each type of questions in the training set.

### 4.3. Sentence retrieval

To generate semantic contexts, for each question–answer pair, we define an argument set as the set of terms which a relevant document should contain. For example, consider the question–answer pair:

Q: *Who provides telephone service in Orange County, California?*
A: *Pacific Bell*

Any relevant document to this question–answer pair must contain the terms "*telephone service*", "*Orange County, California*", and "*Pacific Bell*". Therefore, to search documents on the Web, we formulate a query made up of all the arguments found in the question–answer pair. The argument set is made up of all the base noun phrases in the question (found using the BaseNP chunker [29]).

In the TREC-8–11 collections, the answers are typically constituted of a noun phrase. However, some supporting documents may only contain part of this noun phrase. To increase the recall of document retrieval, we search for a combination of question arguments and each sub-phrase of the answer. We restrict each sub-phrase to contain less than four words and to contain no stop word. Finally, we assign a score to each sub-phrase according to its length (measured in words) relative the length of the candidate answer. For example, the sub-phrases and the score assigned for the previous question–answer pair are: {`Pacific Bell` 1, `Pacific` $\frac{1}{2}$, `Bell` $\frac{1}{2}$}. The sub-phrase score will be used later to rank the extracted candidate answers from the retrieved sentences.

Once the argument set is built, we construct a query using all the arguments extracted from the question, and the original candidate answer or one of its sub-phrases. We send the query to Google and then we scan the first 500 retrieved documents to identify sentences that contain all of the question arguments and at least one answer argument.

### 4.4. Semantic filtering of sentences

The set of sentences retrieved by Google are then filtered, according to the validity of the semantic relation that they contain. To do this, we need to find sentences that contain equivalent semantic relations holding

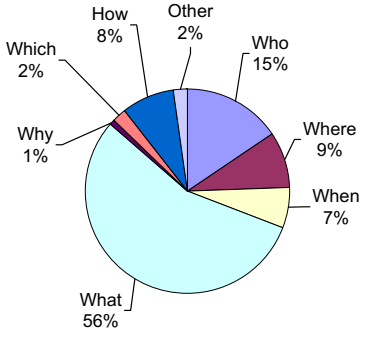


Fig. 1. Distribution of question types in the training corpus (TREC-8, 9, 10).

between question arguments and the answer. We assume that the semantic relation generally appears as the main verb of the question. For example, the verb '*provide*' is considered as the semantic relation in the following question–answer pair:

Q: *Who provides telephone service in Orange County, California?*
A: *Pacific Bell*

To check semantic equivalence, we examine all verbs in the selected sentences for a possible semantic equivalence using WordNet. We check if the main verb of the sentence is a synonym, hypernym, or hyponym of the original verb in the question.

Initially, we only attempt to validate verbs but if the semantic relation is not found through the verbs, then we also validate nouns and adjectives because the semantic relation may occur as a nominalization or another syntactic construction. For this, we use the Porter stemmer [30] to find the stem of the adjectives and nouns and then we check if it is equivalent to the stem of the original verb or one of its synonyms, hypernyms, or hyponyms.[2]

For example, with our running example, both these sentences will be retained:

Sentence 1 *Pacific Bell, major provider of telephone service in Orange County, California...*

Sentence 2 *Pacific Bell Telephone Services today offers the best long distance rate in Orange County, California.*

### 4.5. Generating the answer contexts

Once we have identified a set of semantically equivalent sentences, we try to generalize them into a pattern using both syntactic and semantic features. Each sentence is tagged and syntactically chunked (with [29]) to identify POS tags and base noun phrases. To construct a general form for answer patterns, we replace the noun phrase corresponding to the argument in the answer by the corresponding named entity tag (e.g ⟨ORGANIZATION⟩) and the noun phrases corresponding to the question arguments by the tag ⟨QARGx⟩ where x is the argument identifier. We replace the other noun phrases that are neither question arguments nor answer arguments with the syntactic tag ⟨NPx⟩, where x is the noun phrase identifier. To achieve a more general form of the answer pattern, all other words except prepositions are removed. For example, the following sentence chunked with NPs:

```
[California's/NNP Baby/NNP Bell,/NNP SBC/NNP Pacific/NNP Bell,/NNP]/NP
still/RB
provides/VBZ
nearly/RB
all/DT
of/IN
[the/DT local/JJ phone/NN service/NN]/NP]/NP
in/IN
[Orange/NNP County,/NNP California./NNP]/NP
```

will generate the following pattern:

```
⟨ORGANIZATION⟩ ⟨VERB⟩ ⟨QARG1⟩ in ⟨QARG2⟩ | senseOf(provide)
```

The constraint `senseOf(provide)` indicates the semantic relation to be found in the candidate sentences through a verb, a noun or an adjective.

[2] A more proper method to deal with nominalizations would be to use a lexical resource such as NOMLEX [31] (as in the work of [32]). We used the Porter stemmer, however, because it is faster and more robust to words that are absent in the lexical resource.

### 4.6. Weighting the patterns

As one pattern may be more reliable than another, the last challenge is to assign a weight to each candidate pattern. This helps us to better rank the pattern list, and ultimately the answer extracted from them, by their quality and precision. From our experiments, we found that the frequency of a pattern, its length, the answer sub-phrase score, and the level of semantic similarity are the most indicative factors in the quality of each pattern. We set up a function to produce a weight for each pattern over the above major factors; these weights are defined to have values between 0 and 1.

More formally, let $P_i$ be the $i$th pattern of the pattern set $P$ extracted for a question–answeranswer pair; we compute each of the following factors:

- $Count(P_i)$ is the number of times pattern $P_i$ was extracted for a given question pattern. The most frequent the pattern, the more confidence we have in it and the better we rank it.
- *Distance* measures the distance (in number of words) between the answer and the closest term from the question arguments in the pattern. The smallest the distance, the more confidence we have in the pattern.
- $Length(P_i)$ is the length of the pattern $P_i$ measured in words. A shorter pattern will be given a better rank.
- *Sub_phrase_score* is the score of the candidate answer sub-phrase. The score of each answer sub-phrase depends on its similarity to the full candidate answer. Here we have used the simple heuristic method to score a sub-phrase by its length as $\frac{number\ of\ words\ present\ in\ both\ p_i\ and\ candidate\ answer}{total\ number\ of\ words\ in\ the\ candidate\ answer}$.
- $Sem\_sim(V_Q, S_{P_i})$ measures the similarity between the sense expressed in the candidate pattern ($S_{P_i}$) (through a verb, a noun or an adjective) and the original verb in the question ($V_Q$). We want to estimate the likelihood that the two words actually refer to the same fact or event. This weight is based on the type of semantic relation between the terms and $V_Q$ as specified in WordNet:
  - 1 pt for the original verb in the question;
  - $\frac{1}{2}$ pt for strict synonyms of the question verb, and
  - $\frac{1}{8}$ pt for hyponyms and hypernyms of the question verb.

The final weight of a pattern is based on the combined score of the previous four factors computed as:

$$Weight(P_i) = \frac{count(P_i)}{count(P)} \times \frac{1}{length(P_i)} \times \frac{1}{distance} \times sub\_phrase\_score \times sem\_sim(V_Q, S_{P_i})$$

For the TREC-8, 9, 10 question sets, a list of 98 ranked patterns were created automatically by the system. This can be compared to the bag of 77 hand-made patterns in the original system created using the TREC-8, 9 question set.

### 4.7. Evaluation

The system was implemented as a cascade of Perl scripts using the Google search engine. It was developed for experimental purposes only and very little effort was made to make the code efficient, easy to use or modular. Our main goal was to evaluate the quality of the results.

We evaluated the automatically created patterns using the 493 questions-answers of the TREC-11 collection data [33]. The use of only the TREC question set for training and evaluating does create a bias in the system. The TREC–QA question set contains factoid questions that are mostly answered by short noun phrases. Our patterns (both manual and induced) were developed using the TREC questions and are thus biased towards these types of questions that can be answered by stereotypical answer contexts. The evaluation with a different corpus would probably lead to a lower MRR, especially if the questions are not factoid in nature. For example, more explanation-based or *why*/*how* questions as in closed-domain QA would have been harder to deal with.

Table 2
Results of the generated patterns compared with the original hand-crafted patterns (TREC-11 data)

| System | #Q with a correct answer in top five candidates | Δ (%) | Precision of candidate list | Δ (%) | MRR | Δ (%) |
|---|---|---|---|---|---|---|
| Hand-crafted patterns | 86 | | 0.50 | | 0.32 | |
| Generated patterns | 101 | 17 | 0.58 | 16 | 0.40 | 26 |

The system was evaluated with the original 77 hand-crafted patterns and with the 98 learned ones; then the answers from both runs were compared. Table 2 shows the result of this comparison based on precision of the candidate list, number of questions with at least one correct answer in the top five candidates and mean reciprocal rank (MRR). The evaluation shows an increase in precision of about 16% with the generated patterns (from 0.50 to 0.58). This shows that the semantic constraints have filtered out some bad candidates that the original patterns accepted. The MRR, which takes the order of the candidates into account, increased by 26% from 0.32 to 0.40. In addition, since the patterns are generated automatically, no manual work is now necessary.

## 5. Semantic candidate re-ranking

A further analysis of the results, however, showed that although the semantic constraints imposed by the new patterns filtered out noisy candidates, quite a few bad answers still remained. This is because at least one document contained the semantic relation and the question arguments in the same sentence. Our next goal was then to improve these results by filtering out noisy candidates and re-rank the remaining candidates better.

To re-rank the candidates, we used a redundancy technique, but this time, based on the satisfaction of the semantic constraints. That is, we evaluate how many times the candidate answer satisfies the semantic constraint then re-rank the list of candidates according to this proportion. If the semantic relation appears in the same sentence as the question arguments by chance, it should thus be given a lower rank or be removed completely. Let us describe this process in detail.

### 5.1. Sentence retrieval

We first run the QA system on the Web and retrieve its top 200 answer candidates. This first run can be done with the newly acquired semantic patterns or the original hand-crafted ones. In fact, Section 5.4 presents the results for both methods. For example, with our question *Who provides telephone service in Orange County, California*, the system retrieves the following candidates:

```
Southwestern Bell
Pacific Bell
```

Similarly to our approach for learning reformulations, we build a set of argument tuples composed of the candidate answers and the argument expressed in the question. In order to achieve this task, we decompose the original question into two parts: the main semantic relation expressed (e.g. *provides*) and the argument(s) of the relation (e.g. *telephone service* and *Orange County, California*). A set of argument tuples is then created from the noun phrases of the question and the candidate found by the QA system. In our example, the following tuples are created:

```
('telephone service', 'Orange County, California', 'Southwestern Bell')
('telephone service', 'Orange County, California', 'Pacific Bell')
```

Once we have built the set of argument tuples, we search for them in the document collection to identify the possible semantic relations relating them, and make sure that the relation that relates them in the documents is equivalent to what we were originally looking for in the question (`senseOf(provide)`).

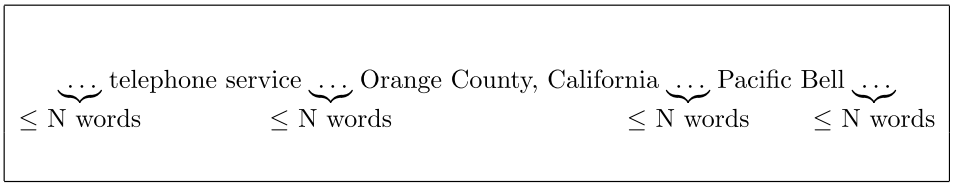


Fig. 2. Example of a context window.

In our experiment, we submitted all the tuples to the Web to find paragraphs that contained these tuples. Then we extracted only the paragraphs where both tuple elements are at a distance of $N$ words or less (in our experiment, $N = 5$). We used a context window size of $N$ words between the tuple elements and $N$ words on each side of them in the extracted paragraphs and then examined the words in these context windows for a possible similar semantic relation. This is shown in Fig. 2.

### 5.2. Evaluating the semantic relation

Finally, we evaluate the relations expressed in the context windows to identify if at least one is semantically equivalent to the original semantic relation expressed in the question. To verify the semantic relation, we use the same procedure as for learning patterns (see Section 4.4). We first check if any verb found in any context window is a synonym, a hypernym or a hyponym of the original verb in the question. If no verb has an equivalent semantic relation, we then back-off to validating nouns and adjectives. Any tuple that does not have a similar semantic relation in the question and in the documents is discarded. Thus if a candidate had been selected in the first QA run, but no further evidence is found in the re-ranking phase, it is filtered out.

### 5.3. Re-ranking candidates

The remaining candidates are re-ranked according to the proportion of passages in the collection containing the same relation. For example, when we submitted the tuple (`'telephone service'`, `'Orange County`, `California'`, `'Pacific Bell'`), we found 110 passages containing the elements of the tuple. Among these, only 24 contained the tuples and the relation `senseOf (provide)` within five words of each other. We therefore gave a rank of (24/110) to the candidate `Pacific Bell`. By applying this procedure to all the argument tuples, all candidates can be easily re-ranked.

### 5.4. Evaluation

We evaluated the semantic re-ranking alone again with the TREC-11 data. Table 3 shows the results. Here again, the MRR improved (by 67%). This means that the candidates found are better ordered in the list so as to move the correct answers up in the list. In fact, with the TREC-11 data, 42% of correct answers were moved up in the candidate list by 3.8 positions on average while 4% were actually ranked worse by 5.7 positions.

Table 3
Results of the semantic re-ranking (TREC-11 data)

| System | #Q | MRR | $\Delta$ (%) |
|---|---|---|---|
| Hand-crafted patterns | 493 | 0.321 | |
| Semantic re-ranking | 493 | 0.537 | 67 |

## 6. Evaluation of the combined approach

### 6.1. Results

Finally, we evaluated the combined approach: automatically acquired semantic patterns (Section 4) and semantic re-ranking (Section 5). Again, we used the TREC-11 collection for testing. The results are reported in Tables 4 and 5.

As Table 4 shows, with the combined approach ($A + B$), the MRR increased by 73% compared to the original system. The precision is also higher than the original system (0.64 versus 0.50) yet does not rely on manual expertise to hand-craft patterns. It is therefore more interesting when portability is an issue, for example, when doing QA on a different domain or a new language.

Table 5 shows the details of the evaluation for each type of question. The proportion of each type of question is similar in the testing corpus (see Fig. 3) than in the training corpus (see Fig. 1), hence the training corpus is representative in this respect and each question type benefits from this approach. The results of *what* and *how* questions are particularly higher (the MRR more than doubled). We speculate that these types of questions are answered by sentences that are more stereotypical; thus a small number of reformulation patterns are sufficient to cover a larger number of answers.

Table 4
Results of each type of approach (TREC-11 data)

| | #Q with a correct answer in top five | Precision | $\Delta$ (%) | MRR | $\Delta$ (%) |
|---|---|---|---|---|---|
| Hand-crafted patterns | 86 | 0.50 | | 0.32 | |
| Generated patterns ($A$) | 101 | 0.58 | 16 | 0.40 | 26 |
| Semantic re-ranking ($B$) | 86 | | | 0.54 | 67 |
| Combined ($A + B$) | 99 | 0.64 | 28 | 0.55 | 73 |

Table 5
Results of the combined approach based on question categories (TREC-11 data)

| Question type | Hand-crafted patterns | | Combined ($A + B$) | | | |
|---|---|---|---|---|---|---|
| | Precision | MRR | Precision | $\Delta$ (%) | MRR | $\Delta$ (%) |
| Who | 0.57 | 0.30 | 0.71 | 24 | 0.45 | 49 |
| What | 0.50 | 0.23 | 0.61 | 12 | 0.55 | 138 |
| Where | 0.53 | 0.50 | 0.79 | 47 | 0.79 | 57 |
| When | 0.69 | 0.55 | 0.72 | 5 | 0.62 | 89 |
| How | 0.28 | 0.19 | 0.55 | 97 | 0.48 | 130 |
| Which | 0 | 0 | 0 | 0 | 0 | 0 |
| Total | 0.50 | 0.32 | 0.638 | 28 | 0.55 | 73 |

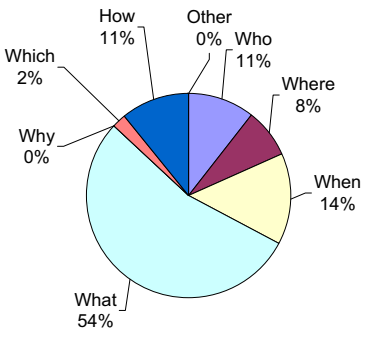


Fig. 3. Distribution of question types in the testing corpus (TREC-11).

### 6.2. Discussion

The evaluation of the QA system presented in the previous section shows that the acquired patterns improve the performance of the system. However, other more qualitative evaluations can be done. In the following paragraphs, we will discuss a few interesting issues.

*Expressivity and ease of maintenance:*

The generated patterns use the same formalism and vocabulary as the manual ones. In fact, this allows us to turn the automatic patterns on and off, and use each type interchangeably in the system. The automatic patterns are therefore as easy (or as hard) to read and modify as the manual ones.

*Complementarity of the manual and the induced patterns:*

An interesting question is to determine if the manual patterns and the induced ones cover the same questions or if they complement each other. Table 6 shows the evaluation of the two types of patterns compared to using no reformulation at all. As the table shows all types of questions (except *which*), benefit from reformulations (whether they are hand-made or generated).

It is interesting to note however, that the manual patterns are most beneficial for *who*, *how* and *where* questions; whereas the induced patterns improve *how*, *what* and *who* questions the most. As mentioned earlier, we suspect that *who* and *how* (e.g. *how many*, *how much*, etc.) questions are answered by sentences that are more stereotypical; thus the reformulation patterns are sufficient to cover a larger number of answers. The induced patterns do not answer *where* questions as one would have expected from the results of the hand-crafted patterns; a more detailed failure analysis is needed to understand why this is so.

As Table 6 shows, the two types of patterns do improve a few common question types, but they seem to complement each other for other types of questions. For those questions, one would expect that using the two types of patterns together should produce an even higher MRR. However, we have not tested this hypothesis experimentally, and a more detailed analysis would be needed.

*Scalability of the approach:*

The approach presented here has been developed and tested with the TREC open-domain factoid questions. As noted earlier, this has biased the patterns induced to these types of questions that can be answered with short phrases. It is not clear how the approach can be scaled up to handle a wider range of questions. For example with more explanation-based questions, the answer will not necessarily be formed of a short noun-phrase and within a strongly stereotypical answer pattern. Also, the approach needs a large document collection to learn the patterns (e.g. the Web). In the case of restricted domain QA, where the document collection is much smaller [34], it is not clear if the approach will be able to learn useful patterns.

On the other hand, the fact that semantic-based answer patterns can be learned automatically without degrading the system's performance compared to hand-crafted patterns makes the approach portable to other application domains and other languages.

*Limitations of the approach:*

As opposed to several other approaches that use the Web for answer redundancy; our approach is less strict as it looks for reinforcement of the semantic relation between the arguments, rather than looking only for lexically or syntactic similarity. In this respect, our approach is much more tolerant and allows us to find more evidence to support answers. However, as we saw in Section 4, the formalism used to represent patterns is very

Table 6
MRR with no patterns, the original hand-crafted patterns and the new patterns and the semantic re-ranking (combined) and improvement in MRR compared to using no pattern at all

| Question type | MRR (% improvement) with | | |
|---|---|---|---|
| | No pattern | Hand-crafted patterns | Combined approach ($A + B$) |
| When | 0.48 | 0.55 (15%) | 0.62 (29%) |
| Where | 0.37 | 0.50 (35%) | 0.79 (114%) |
| How | 0.13 | 0.19 (46%) | 0.48 (269%) |
| What | 0.18 | 0.23 (28%) | 0.55 (205%) |
| Which | 0 | 0 | 0 |
| Who | 0.16 | 0.30 (88%) | 0.45 (181%) |

crude; we only look for lexical, syntactic and semantic evidence within a window of words. We do not attempt to unify a proper semantic representation of the questions with that of candidate sentences. We chose not to do this, because in our view, creating such semantic representations is a difficult and error-prone task. However, as we look for evidence anywhere in a window of words, we are more sensitive to mistakes. We are only interested in finding a word that carries a similar sense without doing a full semantic parse of the sentence. Negations and other modal words may completely change the sense of the sentence. When looking in a very large corpus such as the Web, this may lead to more noise.

## 7. Conclusion and future work

We presented a novel method for acquiring reformulation patterns automatically based on lexical, syntactic and semantic features then used these semantic constraints to re-rank the list of candidate answers using redundancy.

The experimental evaluation shows that using new semantic patterns increases the performance or our QA system. Together, the automatically acquired semantic patterns and the candidate re-ranking improve the performance significantly (28% for precision and 73% for MRR) and remove the need for human intervention. It is therefore very interesting when porting the system to a new language or a new domain.

The current implementation only looks at semantic relations holding between two or three arguments. It can easily be extended to consider variable–size relations; however, as more constraints are taken into account, the precision of the candidate list is expected to increase, but recall is expected to decrease. A careful evaluation would be necessary to ensure that the approach does not introduce too many constraints and consequently filters out too many candidates.

Another interesting question is to what degree the results are bound to the thresholds we have used. For example, we have arbitrarily taken the first 500 hits from Google to generalize answer patterns. It is not clear if or how changing this value will affect the results.

This work tried to learn answer patterns automatically without degrading the quality of the QA results. We have not, however, looked at performance issues of the system. In a real-time QA system, quality is important but if a question takes too long to be analyzed, the system is practically unusable. Further work is thus necessary to measure such things as response time and scalability to a real application.

## Acknowledgements

This project was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Bell University Laboratories (BUL). The authors would like to thank the anonymous referees for their valuable comments.

## References

[1] S. Harabagiu, S. Maiorano, M. Pasca, Open-domian textual question answering techniques, Natural Language Engineering 1 (2003) 1–38.
[2] E.M. Voorhees, D.K. Harman (Eds.), Proceedings of the Eighth Text REtrieval Conference (TREC-8), Gaithersburg, Maryland, NIST, 1999.
[3] E.M. Voorhees, Overview of the TREC 2001 question answering track, in: Proceedings of the 10th Text REtrieval Conference (TREC 2001), Gaithersburg, Maryland, 2001, pp. 157–165.
[4] C.L.A. Clarke, G.V. Cormack, T.R. Lynam, Exploiting redundancy in question answering, in: Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR-2001), 2001, pp. 358–365.
[5] S.M. Harabagiu, D.I. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R.C. Bunescu, R. Girju, V. Rus, P. Morarescu, Falcon: boosting knowledge for answer engines, in: Proceedings of the the Ninth Text REtrieval Conference (TREC-9), Gaithersburg, Maryland, 2000.
[6] K. Ahn, J. Bos, J.R. Curran, D. Kor, M. Nissim, B. Webber, Question answering with QED at TREC-2005, in: Proceedings of the the 14th Text REtrieval Conference (TREC-2005), Gaithersburg, Maryland, 2005.
[7] E.M. Voorhees, D.K. Harman, TREC: Experiment and Evaluation in Information Retrieval, MIT Press, 2005.
[8] M. Soubbotin, S. Soubbotin, Patterns of potential answer expressions as clues to the right answers, in: Proceedings of the 10th Text REtrieval Conference (TREC 2001), Gaithersburg, Maryland, 2001, pp. 175–182.

[9] E. Brill, J. Lin, M. Banko, S. Dumais, A. Ng, Data-intensive question answering, in: Proceedings of the 10th Text REtrieval Conference (TREC 2001), Gaithersburg, Maryland, 2001, pp. 393–400.
[10] E.M. Voorhees, D.K. Harman (Eds.), Proceedings of the 10th Text REtrieval Conference (TREC 2001), Gaithersburg, Maryland, NIST, 2001.
[11] M. Kaisser, T. Becker, Question answering by searching large corpora with linguistic methods, in: Proceedings of the 13th Text REtrieval Conference (TREC 2004), Gaithersburg, 2004.
[12] S. Cucerzan, E. Agichtein, Factoid question answering over unstructured and structured content on the Web at TREC 2005, in: Proceedings of the 14th Text REtrieval Conference (TREC 2005), Gaithersburg, 2005.
[13] R. Aceves-PTrez, L. Villaseor-Pineda, M.M. yGmez, Towards a multilingual QA system based on the web data redundancy, in: Proceedings of the Third Atlantic Web Intelligence Conference, AWIC 2005, Lecture Notes in Artificial Intelligence, vol. 3528, Springer, Lodz, Poland, 2005.
[14] E. Agichtein, S. Lawrence, L. Gravano, Learning search engine specific query transformations for question answering, in: Proceedings of WWW10, Hong Kong, 2001, pp. 169–178.
[15] E. Agichtein, L. Gravano, Snowball: extracting relations from large plain-text collections, in: Proceedings of the Fifth ACM International Conference on Digital Libraries, 2000.
[16] S. Lawrence, C.L. Giles, Context and page analysis for improved web search, IEEE Internet Computing 2 (1998) 38–46.
[17] D. Ravichandran, E. Hovy, Learning surface text patterns for a question answering system, in: Proceedings of the 40th ACL Conference, Philadelphia, 2002.
[18] U. Hermjakob, A. Echihabi, D. Marcu, Natural language based reformulation resource and wide exploitation for question answering, in: Proceedings of the 11th Text REtrieval Conference (TREC 2002), Gaithersburg, Maryland, 2002.
[19] C.C.T. Kwok, O. Etzioni, D.S. Weld, Scaling question answering to the web, in: World Wide Web, 2001, pp. 150–161.
[20] D.R. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan, J.M. Prager, Mining the web for answers to natural language questions, in: CIKM, 2001, pp. 143–150.
[21] F. Duclaye, F. Yvon, O. Collin, Using the web as a linguistic resource for learning reformulations automatically, in: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02), Las Palmas, Spain, 2002, pp. 390–396.
[22] D. Mollà, Learning of graph-based question answering rules, in: Proceedings of HLT/NAACL 2006 Workshop on Graph Algorithms for Natural Language Processing, 2006, pp. 37–44.
[23] J. Sowa, Semantics of conceptual graphs, in: Proceedings of ACL 1979, 1979, pp. 39–44.
[24] M. Stevenson, M. Greenwood, A semantic approach to IE pattern induction, in: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, Michigan, 2005, pp. 379–386.
[25] L. Kosseim, L. Plamondon, L. Guillemette, Answer formulation for question-answering, in: Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI'2003), Halifax, Canada, AI-2003, 2003.
[26] L. Plamondon, G. Lapalme, L. Kosseim, The QUANTUM question-answering system at TREC-11, in: Proceedings of TREC 2001, Gaithersburg, Maryland, 2002.
[27] NIST, in: Proceedings of TREC-9, Gaithersburg, Maryland, NIST, 2000.
[28] L. Plamondon, G. Lapalme, L. Kosseim, The QUANTUM question answering system, in: Proceedings of the 10th Text Retrieval Conference (TREC 2001), Gaithersburg, Maryland, 2001, pp. 157–165.
[29] L. Ramshaw, M. Marcus, Text chunking using transformation-based learning. In: Proceedings of the Third ACL Workshop on Very Large Corpora, MIT, 1995, pp. 82–94.
[30] M. Porter, An algorithm for suffix stripping, Program 14 (1980) 130–137.
[31] C. Macleod, R. Grishman, A. Meyers, L. Barrett, R. Reeves, Nomlex: a lexicon of nominalizations, in: Proceedings of the EURALEX'98, 1998.
[32] A. Meyers, C. Macleod, R. Yangarber, R. Grishman, L. Barrett, R. Reeves, Using NOMLEX to produce nominalization patterns for information extraction, in: Proceedings of the COLING-ACL'98 Workshop on Computational Treatment of Nominals, Montreal, Canada, 1998.
[33] E.M. Voorhees, D.K. Harman (Eds.), Proceedings of the 11th Text REtrieval Conference (TREC 2002), Gaithersburg, Maryland, NIST, 2002.
[34] F. Rinaldi, M. Hess, J. Dowdall, D. Mollà, R. Schwitter, Question answering in terminology-rich technical domains, in: M. Maybury (Ed.), New Directions in Question Answering, 2004, pp. 71–86.

**Leila Kosseim** is an associate professor in the Department of Computer Science and Software Engineering at Concordia University in Montreal, Canada.

She received her Ph.D. from Université de Montréal in 1995. She received an NSERC industrial post-doctorate fellowship and worked on semantic grammatical correction for 2 years after her Ph.D. She then joined the RALI laboratory at the Université de Montréal as a researcher and worked on information extraction. She finally joined Concordia University as a professor in 2001.

Her research interest is in Natural Language Processing (NLP), specifically: Question Answering, Natural Language Generation and Word Sense Disambiguation. She also has a strong interest for machine learning, data mining, information retrieval and logic programming. She is the author of over 50 papers in NLP.

**Jamileh Yousefi** received her Master in Computer Science from Concordia University (Montréal) in 2005. Her thesis, supervised by Leila Kosseim, is the source of the current paper. She is the author of three papers on question-answering. She now works in Toronto, Canada.