

# Using Semantic Constraints to Improve Question Answering

Jamileh Yousefi and Leila Kosseim

CLaC Laboratory  
Department of Computer Science and Software Engineering  
Concordia University  
1400 de Maisonneuve Blvd. West  
Montreal, Quebec, Canada H3G 1M8  
j\_yousefi@cs.concordia.ca, kosseim@cs.concordia.ca

**Abstract.** In this paper, we discuss our experience in using semantic constraints to improve the precision 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 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 about 500 questions from TREC-11 shows that the acquired semantic patterns increase the precision by 16% and the MRR by 26%, the re-ranking using semantic redundancy as well as the combined approach increase the precision by about 30% and the MRR by 67%. This shows that no manual work is now necessary to build question reformulations; while still increasing performance

## 1 Introduction

Question reformulation (also called *surface pattern*, *paraphrase*, *answer pattern*, ...) tries to identify various ways of expressing an answer given a natural language question. These reformulations are often used in a Question Answering (QA) system to retrieve answers in a large document collection. For example given the question *Who is the president of the U.S.?*, a reformulation-based QA system will search for formulations like *the president of the U.S. is <NP>* or *<NP>, the president of the U.S.* 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 president of the U.S. is a nut lover*) but should also identify many candidate answers to increase the system's confidence in them.

Most work on reformulations have used patterns based on string constraints, syntactic constraints or named entity tags (e.g. person-name, organization, ...). However, only a few have worked on semantically equivalent reformulations such

as  $\langle NP \rangle$ , also known as the leader of the United States or at the top of the US government is  $\langle NP \rangle$ .

We believe that stronger semantic constraints can be beneficial to find 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 performance.

## 2 Related Work

Soubotin et al. [1] along with [2] were among the first to use surface patterns as the core of their QA system. This approach searches the document collection for predefined patterns or exact sentences that could be the formulation of the potential answer. [1] wrote their patterns by hand and were among the best scoring team at the TREC-10 QA track [3]. Their work shows that if enough human resources are available, handcrafted rules can produce excellent results.

Given the success of this approach, many attempts have then been made to acquire reformulations automatically. [2] use simple word permutations to produce paraphrases of the question. More recently, [4] also uses simple word permutations and verb movements to generate paraphrases for their multilingual QA system.

In the work of [5, 6, 7], answer formulations are produced for query expansion to improve information retrieval. While in [7] reformulation rules to transform a question of the form *What is X?* into *X is* or *X refers to* are built by hand, [6, 5] learns to transform natural language questions into sets of effective search engine queries, optimized specifically for each search engine.

[8] 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 generalisation of the *is-born* relation. [9] does use semantic paraphrases, called *phrasal synonyms*, to enhance their TextMap QA system. However, many of these patterns are manual generalisations of patterns derived automatically by [8].

[10] use transformational grammar to perform syntactic modifications such as Subject-Aux and Subject-Verb movements. [11] learns the best query reformulations (or paraphrases) for their probabilistic QA system. Here again, the paraphrases are syntactic variations of the original question.

[12], 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.

In these systems and most similar approaches, automatic paraphrases are constructed based on lexical or syntactic similarity. When searching a huge document

collection such as the Web, having only syntactic reformulations is acceptable because the collection exhibits a lot of redundancy. However, in a smaller collection, semantic reformulations are necessary.

### 3 Initial Hand-Crafted Patterns

Our work builds on our current reformulation-based QA system [13, 14], where reformulations were hand-crafted and only relied on named entities for semantic constraints. Given a question, the system 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 898 questions of TREC 8 & 9 as training set and used the 1000 questions of TREC 10 & 11 for testing. In total, 77 formulation templates were created, 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, 2 answer formulations are produced per question.

### 4 Learning Semantic Answer Patterns

Our goal is to find many sentences from the Web that contain the correct answer and try to generalize them into syntactico-semantic patterns. First, we use a training corpus of question-answer pairs from which we learn how to generalise each type of questions. Each question-answer pair is analysed to extract its 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. Let us describe this process in detail.

## 4.1 The Training Corpus

We start with a training corpus of 1343 question-answer pairs taken from the TREC-8, TREC-9, and TREC-10 collection data [15, 16, 3]. Each question-answer pair is composed of one question and its corresponding answer. For example:

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

We divided the training corpus according to the question type. We used the classification used in [17] to categorize questions into 7 main classes (what, who, how, where, when, which, why) and 20 subclasses (ex. what-who, who-person, how-many, how-long, ...).

**Sentence Retrieval.** 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 [18]).

In the TREC 8-11 collections, the answers are typically 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<sup>1</sup> 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.

**Semantic Filtering of Sentences.** We then filter the set of sentences retrieved by Google, 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 between question arguments and the answer. We assume that the semantic relation generally appears as the main verb of the question. For

---

<sup>1</sup> This limit was set arbitrarily.

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.

At first, 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 nominalisation or another syntactic construction. For this, we use the Porter stemmer [19] 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.

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

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

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

**Generating the Answer Pattern.** 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 [18]) 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 counter. 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 counter. 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 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.

In total 98 patterns were created automatically, compared to the 77 hand-made patterns in the original system.

**Evaluation.** We tested our newly created patterns using the 493 questions-answers from the TREC-11 collection data [20] and our own QA system [13, 14]. 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 1 shows the result of this comparison based on precision, number of questions with at least one correct answer in the top 5 candidates and mean reciprocal rank (MRR). The evaluation shows an increase in precision of about 16% with the generated patterns (from 0.497 to 0.577). 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 0.321 to 0.404. In addition, since the patterns are generated automatically, no manual work is now necessary.

**Table 1.** Results of the generated patterns compared with the original hand-crafted patterns (TREC-11 data)

System	Nb of questions	Nb of questions with a correct answer in the top 5 candidates	Precision of candidate list	MRR
Original System (Hand-Crafted Patterns)	493	86	0.497	0.321
Generated Patterns	493	101	0.577	0.404
Improvement		17%	16%	26%

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.

## 5 Semantic Candidate Re-ranking

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.

**Sentence Retrieval.** We first run the QA system on the Web and retrieve its top 200 answer candidates<sup>2</sup>. This first run can be done with the newly acquired semantic patterns or the original hand-crafted ones. In fact, section 5 presents the results for both methods.

<sup>2</sup> This number was set arbitrarily.

For example, with our question *Who provides telephone service in Orange County, California*, the system retrieves the following candidates:

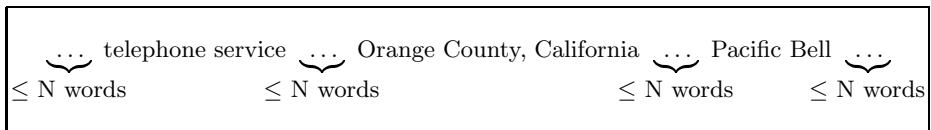
Southwestern Bell  
Pacific Bell

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 (ex. *provides*) and the argument(s) of the relation (ex. *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)`).

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<sup>3</sup> or less. 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 Figure 1.



**Fig. 1.** Example of a context window

**Examining 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 4.1). 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.

<sup>3</sup> In our experiment, N was set to 5.

**Re-Ranking Candidate Answers.** 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 5 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.

**Evaluation.** We evaluated the semantic re-ranking alone again with the TREC-11 data. Table 2 shows the results. Here again, both the precision is higher (from 0.497 to 0.656) and the MRR is higher (from 0.321 to 0.537). A higher MRR 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 and 5% of good answers were lost during the process.

**Table 2.** Results of the semantic re-ranking (TREC-11 data)

System	Nb of questions	Precision	MRR
Original System	493	0.497	0.321
Semantic Re-Ranking	493	0.656	0.537
Improvement		32%	67%

## 6 Evaluation of the Combined Approach

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 3 and 4.

As Table 3 shows, the combined approach (A+B) yields a precision that is higher than the original system (0.638 versus 0.497) yet does not rely on manual expertise to hand-craft patterns. The MRR also increased from 0.321 to 0.554. It is therefore more advantageous when doing QA on a different domain or a new language. A further analysis of the results for each type of question (see Table 4) reveals that all question types benefit from this approach.

**Table 3.** Results of each type of approach (TREC-11 data)

	Nb of questions	Nb of questions with a correct answer in the top 5 candidates	Precision	MRR
Original System	493	86	0.497	0.321
Generated Patterns (A)	493	101	0.577	0.404
Semantic Re-Ranking (B)	493	86	0.656	0.537
Combined (A+B)	493	99	0.638	0.554

**Table 4.** The results of the combined approach based on question categories (TREC-11 data)

Question Type	Frequency	Original system		Combined (A+B)	
		MRR	Precision	MRR	Precision
who	52 (10.4%)	0.301	0.571	0.447	0.710
what	266 (53.2%)	0.229	0.500	0.546	0.612
where	39 (7.8%)	0.500	0.533	0.786	0.786
when	71 (14.2%)	0.688	0.687	0.615	0.720
how + adj/adv	53 (10.6%)	0.194	0.277	0.477	0.545
which	12 (2.4%)	0	0	0	0
why	0 (0.0%)	0	0	0	0

It is worth mentioning that the combined approach (A+B) does not seem to do better at precision than re-ranking (B) alone (0.638 versus 0.656). While this difference is not statistically significant, it appears because the new patterns acquire a different set of candidate answers. In addition, since we used the live Web to perform the experiments, each time we run the system, a different set of answers can be retrieved. The semantic re-ranking (B) was evaluated with the list of candidates extracted from the original system; while the combined approach (A+B) was evaluated as a brand new run. This is why (A) found 101 candidates, while (A+B) found 99.

## 7 Conclusion and Future Work

We presented a method for acquiring reformulation patterns automatically based on both 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 precision by 16% and MRR by 26% compared to hand-crafted rules. The semantic re-ranking improves the results more significantly (about 30% for precision and 67% for MRR); while using the two approaches together is comparable to re-ranking alone, but removes the need for human intervention.

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 similar evidence. In this respect, our approach is much more tolerant and allows us to find more evidence to support answers. On the other hand, as we look for evidence anywhere in a window of words, rather than a strict string match, 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. However, if we perform the QA task on a much smaller corpus, such as in closed-domain QA, looking for semantic equivalences may be more fruitful.

The current implementation only looks at semantic relations holding between two or three arguments. However, 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 generalise answer patterns. It is not clear if or how changing this value will affect the results.

This work tried to improve the quality of our QA system, but without looking at performance issues. In a real-time QA system, quality is important but if a question takes too long to be analysed, the system is practically unusable. Further work is thus necessary to measure such things as response time and scalability to a real application.

**Acknowledgments.** 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. Soubbotin, M., Soubbotin, S.: Patterns of potential answer expressions as clues to the right answers. [3] 175–182
2. Brill, E., Lin, J., Banko, M., Dumais, S., Ng, A.: Data-Intensive Question Answering. In: Proceedings of The Tenth Text Retrieval Conference (TREC-X), Gaithersburg, Maryland (2001) 393–400
3. NIST: Proceedings of TREC-10, Gaithersburg, Maryland, NIST (2001)
4. Aceves-Prez, R., Villaseor-Pineda, L., y Gmez, M.M.: Towards a Multilingual QA System based on the Web Data Redundancy. In: Proceedings of the 3rd Atlantic Web Intelligence Conference, AWIC 2005. Lecture Notes in Artificial Intelligence, No. 3528, Lodz, Poland, Springer (2005)
5. Agichtein, E., Lawrence, S., Gravano, L.: Learning search engine specific query transformations for question answering. In: Proceedings of WWW10, Hong Kong (2001) 169–178
6. Agichtein, E., Gravano, L.: Snowball: Extracting Relations from Large Plain-Text Collections. In: Proceedings of the 5th ACM International Conference on Digital Libraries. (2000)
7. Lawrence, S., Giles, C.L.: Context and Page Analysis for Improved Web Search. *IEEE Internet Computing* **2** (1998) 38–46
8. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proceedings of the 40th ACL conference, Philadelphia (2002)
9. Hermjakob, U., Echihabi, A., Marcu, D.: Natural language based reformulation resource and wide exploitation for question answering. [20]
10. Kwok, C.C.T., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: *World Wide Web*. (2001) 150–161

11. Radev, D.R., Qi, H., Zheng, Z., Blair-Goldensohn, S., Zhang, Z., Fan, W., Prager, J.M.: Mining the web for answers to natural language questions. In: CIKM. (2001) 143–150
12. Duclaye, F., Yvon, F., Collin, O.: 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) 390–396
13. Kosseim, L., Plamondon, L., Guillemette, L.: Answer formulation for question-answering. In: Proceedings of The Sixteenth Canadian Conference on Artificial Intelligence (AI'2003), Halifax, Canada, AI-2003 (2003)
14. Plamondon, L., Lapalme, G., Kosseim, L.: The QUANTUM Question-Answering System at TREC-11. [20]
15. NIST: Proceedings of TREC-8, Gaithersburg, Maryland, NIST (1999)
16. NIST: Proceedings of TREC-9, Gaithersburg, Maryland, NIST (2000)
17. Plamondon, L., Lapalme, G., Kosseim, L.: The QUANTUM Question Answering System. In: Proceedings of The Tenth Text Retrieval Conference (TREC-10), Gaithersburg, Maryland (2001) 157–165
18. Ramshaw, L., Marcus, M.: Text chunking using transformation-based learning. In: Proceedings of the Third ACL Workshop on Very Large Corpora, MIT (1995) 82–94
19. Porter, M.: An algorithm for suffix stripping. *Program* **14** (1980) 130–137
20. NIST: Proceedings of TREC-11, Gaithersburg, Maryland, NIST (2002)