International Conference on Intelligent Text Processing and Computational Linguistics

CICLing 2006: Computational Linguistics and Intelligent Text Processing pp 441-452 | Cite as

Automatic Acquisition of Semantic-Based Question Reformulations for Question Answering

Authors   Authors and affiliations

Jamileh Yousefi, Leila Kosseim

Conference paper

Part of the Lecture Notes in Computer Science book series (LNCS, volume 3878)

Abstract

We use cookies to improve your experience with our site. More information   Accept

Automatic acquisition of semantic-based question reformulations for question answering

Jamileh Yousefi and Leila Kosseim

CLaC laboratory
Department of Computer Science and Software Engineering
1400 de Maisonneuve Blvd. West
Montreal, Quebec, Canada H3G 1M8
j_yousef@cs.concordia.ca, kosseim@cs.concordia.ca

Abstract. In this paper, we present a method for the automatic acquisition of semantic-based reformulations from natural language questions. Our goal is to find useful and generic reformulation patterns, which can be used in our question answering system to find better candidate answers. We used 1343 examples of different types of questions and their corresponding answers from the TREC-8, TREC-9 and TREC-10 collection as training set. The system automatically extracts patterns from sentences retrieved from the Web based on syntactic tags and the semantic relations holding between the main arguments of the question and answer as defined in WordNet. Each extracted pattern is then assigned a weight according to its length, the distance between keywords, the answer sub-phrase score, and the level of semantic similarity between the extracted sentence and the question. The system differs from most other reformulation learning systems in its emphasis on semantic features. To evaluate the generated patterns, we used our own Web QA system and compared its results with manually created patterns and automatically generated ones. The evaluation on about 500 questions from TREC-11 shows some increase in precision and MRR scores. Hence, no loss of quality was experienced, but no manual work is now necessary.

1 Introduction

Question reformulation deals with identifying possible forms of expressing answers given a natural language question. These reformulations can be used in a QA system to retrieve answers in a large document collection. For example given the question *What is another name for the North Star?*, a reformulation-based QA system will search for formulations like *<NP>, another name for the North Star* or *<NP> is another name for the North Star* in the document collection and will instantiate <NP> with the matching noun phrase. The ideal reformulation should not retrieve incorrect answers but should also identify many candidate answers.

Writing reformulations by hand is a tedious tasks that must be repeated for each type of question and for each language in the case of a multilingual QA system. This is why there have been many attempts at acquiring reformulations automatically (ex. [1–5]). However, most work have analyzed string-based or syntactic paraphrases and few have worked on generating semantically equivalent reformulations such as *<NP>, also known as the North Star* or *the North Star is also called <NP>*.

Our goal is to learn semantically equivalent reformulation patterns automatically from natural language questions. We hope to find useful reformulation patterns, which are general enough to be mapped to potential answer contexts but specific enough not to retrieve wrong answers.

2   Related Work

Soubbotin et al. [6] along with [7] were among the first to use surface patterns as the core of their QA system. This approach searches in the document collection for predefined patterns or exact sentences that could be the formulation of the potential answer. In [6], the patterns were hand-crafted, while in [7] simple word permutations were performed to produce paraphrases of the question. More recently, [1] also uses simple word permutations and verb movements to generate paraphrases for their multilingual QA system. In the work of [5, 4, 8], answer formulations are produced for query expansion to improve information retrieval. While in [8] reformulation rules to transform a question of the form *What is X?* into *X is* or *X refers to* are built by hand, [4, 5] learns to transform natural language questions into sets of effective search engine queries, optimized specifically for each search engine. [9] 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 string of words such as *was born on*, *was born in*, ... with no generalisation of the `is-born` relation. [2] 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 [9]. [10] use transformational grammar to perform syntactic modifications such as Subject-Aux and Subject-Verb movements. [11] learn 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 accept-

able because the collection exhibits a lot of redundancy. However, in a smaller collection, semantic reformulations are necessary.

3 Learning reformulation patterns

3.1 Question and Answer Patterns

Our work is based on our current reformulation-based QA system [13, 14], where reformulations were hand-crafted. Given a question, the system needs to identify which answer pattern to look for. It therefore uses:

A question pattern: that defines what the question must look like. For example `Who Vsf PERSON?` is a question pattern that matches *Who is George Bush?*.

An answer pattern: Once a question pattern is activated (is found to match the input question) a set of answer patterns will be looked for in the document collection. An answer pattern specifies the form of sentences that may contain a possible candidate answer. For example, for the question *Who is George Bush?*, the system tries to find sentences that match any one of these answer patterns:

```
<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.

In the current implementation, both sets of patterns are hand-crafted using the following types of tags:

- Named-entity tags (e.g. `PERSON`) – found using the GateNE named entity tagger [15].
- Part-of-speech tags (e.g. `Vsf`) – found using [16]
- Tags on strings (e.g. `QT`, `ANY-SEQUENCE-WORDS`)
- Specific keywords (e.g. `Who`, `by`)

In this paper, we will discuss how answer patterns can be discovered automatically.

3.2 The Training Corpus

Our learning algorithm starts with a training corpus of 1343 question-answer pairs taken from the TREC-8, TREC-9, and TREC-10 collection data [17–19]. Each question-answer pair is composed of one question and its corresponding answer. The following are some examples:

```
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
```

Corpus	Size
Who Corpus	208
Where Corpus	119
When Corpus	88
What Corpus	747
Why Corpus	8
Which Corpus	32
How Corpus	111
Other Corpus	30
Total	**1343**

Table 1. Training corpus files according to the type of question. The size is the number of question-answer pairs.

We divided the training corpus according to the question type. Questions are classified depending on the kind of information sought. In fact, one major factor to guessing the answer type is to know the question type. We used the classification used in [20] 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, . . .). Table 1 shows our training files along with their sizes (number of question-answer pairs).

3.3 Overview of the algorithm

Each question-answer pair is analyzed to extract the arguments and the semantic relation holding between the question arguments and the answer. A query is then formulated using the arguments extracted from the question-answer pair. The formulated query is sent to a Web search engine which returns the N most relevant documents. The sentences that contain all the query terms are then filtered to keep only these that contain the same semantic relation. These are then passed to a sentence splitter, a part-of-speech tagger, and a noun phrase chunker, to select be generalized into an answer pattern using syntactic and semantic tags. Finally, a confidence weight is assigned to each generated pattern according to its semantic distance from the question and the frequency of the pattern. Let us now describe each of these steps in detail.

4   Generating Answer Patterns

The goal here is to find many sentences from the document collection (here, the Web) that contain the answer and see if we can generalize them into syntactico-semantic patterns.

4.1   Extracting Arguments

For each question-answer pair, we define an argument set as the set of terms which we believe a relevant document should contain. To give an 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 contains 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. To obtain the argument sets, the question is chunked (with the BaseNP chunker [16]) to identify its base noun phrases. All the base noun phrases detected in the question are grouped in a set called Q-ARGUMENTS.

Q-ARGUMENTS = {``telephone service'', ``Orange County, California''}

In the TREC 8-11 collections, the candidate answer is typically a noun phrase that can contain one or more words. 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[1] words and to contain no stop word. Finally, we assign a score to each sub-phrase according to proportion of the words in the sub-phrase compared to the total number of words in the candidate answer. For example, the sub-phrases and the score assigned for the previous question-answer pair are:

Pacific Bell 1 $\qquad$ Pacific $\frac{1}{2}$ $\qquad$ Bell $\frac{1}{2}$

The sub-phrase score will be used later to rank the patterns (and ultimately, the extracted answers) from the retrieved sentences (see section 4.6). Finally, we group the original candidate answer and all its sub-phrases in the set ANS-ARGUMENTS. For example,

ANS-ARGUMENTS = {(Pacific Bell,1), (Pacific,$\frac{1}{2}$), (Bell,$\frac{1}{2}$) }

4.2 Document Retrieval

At this stage, we construct a query in the format accepted by the Google search engine. The query is formed using all the arguments extracted from the question (Q-ARGUMENTS), and the original candidate answer or one of its sub-phrases (ANS-ARGUMENTS) are conjugated with arithmetic operators. For example,

``telephone service'' + ``Orange County, California'' + ``Pacific Bell''

We post the structured query to the Google search engine and then we scan the first 500 retrieved documents to identify the sentences that are likely to contain the answer. From the above documents, only those sentences that contain all of the question arguments and at least one answer argument are retained.

4.3 Extracting Semantic Relations

The key aspect of this research is to find reformulations that are semantically equivalent. To do this, we need to find sentences that contain equivalent semantic

---
[1] This limit was set arbitrary.

relations holding between question arguments and the answer. In fact, semantic relations are used in measuring the relevance of sentences with the question-answer pair. 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*

The representation of the above concepts is done in the following constructs:

Relation schema: ARGUMENT-1: *telephone service*

RELATION: *provide*

ARGUMENT-2: *Orange County, California*

If the main verb of the question is an auxiliary verb, then we ignore the semantic relation. For example, in:

Q: *Who is the president of Stanford University?*

A: *Donald Kennedy*

The semantic relation is ignored and the semantic validation of the relevant sentence (see section 4.4) is based on the frequency of the answer context alone.

Once the semantic relation is determined, we generate a semantic representation vector composed of the relation word, its synonyms, one-level hyponyms and all hypernyms obtained from WordNet. This vector will serve later to verify the semantic validity of the sentences retrieved. To weight each answer pattern according to our confidence level, we also assign a weight to each term in the vector based on the semantic distance of the term to the original verb in WordNet. We want to estimate the likelihood that the sentence and the question actually refer to the same fact or event. We assign the similarity measure using the following weights:

- 1: for the original verb in the question.
- $\frac{1}{2}$: for strict synonyms of the question verb, i.e. a verb in the same synset.
- $\frac{1}{8}$: for hyponyms and hypernyms of the question verb.

Since the relation word can be polysemous, we consider all its possible senses. The representation of the above concepts is done in the following construction:

Relation	provide
Synonyms(provide)	{supply, render, offer, furnish, ... }
Hyponyms(provide)	{charge, date, feed, calk, fund, stint, ... }
Hypernyms(provide)	{give, transfer stipulate, qualify, ... }
Semantic representation vector	{(provide,1), (supply,$\frac{1}{2}$), (render,$\frac{1}{2}$), (offer,$\frac{1}{2}$), (furnish,$\frac{1}{2}$), ..., (charge,$\frac{1}{8}$), (date,$\frac{1}{8}$), (feed,$\frac{1}{8}$), ..., (give,$\frac{1}{8}$), (transfer,$\frac{1}{8}$), ... }

4.4 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. We only choose sentences that have the same relation as the relation extracted from the question-answer pair. To

do so, we examine all verbs in the selected sentences for a possible semantic relation. We check if the main verb of the sentence is a synonym, hypernym, or hyponym of the original verb in the question. The verb is valid if it occurs in the semantic representation vector. For example, with our running example, both these sentences will be retained:

Sentence 1 *California's Baby Bell, SBC Pacific Bell, still <u>provides</u> nearly all of the local phone service in Orange County, California, California.*

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

Because both sentences contain a verb (*"provide"* and *"offer"*) that is included in the semantic representation vector of the question verb (*"provide"*).

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 other syntactic and morpho-syntactic variations. In such a case, we use the Porter stemmer [21] to find the stem of the adjectives and nouns and then we check if it has the same stem as the original verb or another verb from its semantic representation vector. For example, for the phrase *"provider of"* we check if the stem of the original verb *"provide"* or one of its synonyms, hyponym or hypernym is the same as *"provide"* (the stem of *"provider"*). For example, the following sentence is also selected:

Sentence 3 *Pacific Bell, major <u>provider</u> of telephone service in Orange County, California...*

4.5 Generating the Answer Pattern

Once we have identified a set of sentences containing the answer, the arguments and the same semantic relation, we try to generalize them into a pattern using both syntactic and semantic features. Each sentence is tagged and syntactically chunked (with [16]) to identify POS tags and base noun phrases. To construct a general form for answer patterns, we replace the noun phrase corresponding to ANS-ARGUMENT by the tag <ANSWER> and the noun phrases corresponding to Q-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 <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]/NP in/IN [Orange/NNP County,/NNP California./NNP]/NP
```

will generate the following pattern:

```
<ANSWER> <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.

Finally, we try to replace the <ANSWER> tag with the corresponding named-entity tag. To do so, we tag the answer in the question-answer pair of the training set with the GateNE named entity tagger [15]. If the answer does not correspond to a named entity, then we back off to using the *expected* named entity corresponding to the question type. Again, if this cannot be determined (e.g. more than one type can satisfy the question), then we back off to a syntactic tag. In our example, since the question calls for an organization, the following are produced:

```
<ORGANIZATION> <VERB> <QARG1> in <QARG2> | senseOf(provide)
<ORGANIZATION> <QARG1> <QARG2> | senseOf(provide)
<ORGANIZATION> <QARG1> <VERB> <QARG2>| senseOf(provide)
```

4.6   Assigning Confidence Weights

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 between the main verbs of the pattern and the question 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 ith pattern of the pattern set P extracted for a question-answer pair; we compute each factor as the following:

$count(P_i)$ 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 (number of words) between the answer and the closest term from Q-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 $\frac{number\ of\ words\ that\ are\ present\ in\ both\ p_i\ and\ candidate\ answer}{total\ number\ of\ words\ in\ the\ candidate\ answer}$.

$semantic\_sim(V_Q, S_{P_i})$ 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. Here, we use the weights given to terms in the semantic representation vector of (V_Q). As described in section 4.3, 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 for a pattern is based on the combined score of the previous four factors computed as:

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

Note that this function is not necessarily the optimal way of combining these contributing factors, nor are the factors complete by any means. However, as long as long as it produces an acceptable ranking, we can apply it to the patterns. The error produced by just a simple acceptable ranking function is negligible compared to the error present in other modules, such as the named entity recognizer.

Figure 1 shows an example of a question pattern along with its ranked answer patterns for the question *Who was the first man to fly across the Pacific Ocean?*.

Weight	Answer Pattern
1.00	`<QARG1> on <QARG2> <PERSON>	senseOf(fly)`
0.59	`<QARG1> to <VERB> on <QARG2> <PERSON>	senseOf(fly)`
0.43	`<PERSON> QARG1 on QARG2	senseOf(fly)`
0.24	`<QARG2> to <VERB> on <QARG2> <PERSON>	senseOf(fly)`

Fig. 1. Example of ranked answer patterns for the question *Who was the first man to fly across the Pacific Ocean?*.

5   Evaluation

We tested our newly created patterns using the 493 questions-answers from the TREC-11 collection data [22]. We submitted these questions to our Web-QA system [13, 14]. The system was evaluated with the original hand-crafted reformulation patterns and with learned ones. Then the answers from both runs were compared. Tables 2, 3 and 4. Tables 2 and 3 show the result of this comparison based on precision and the number of questions with at least one candidate answer. Table 4 shows the mean reciprocal rank (MRR) for each type of question. The evaluation shows comparable results in precision and MRR scores with a slight increase with the generated patterns. Hence, no loss of quality was experienced, but no manual work is now necessary. Although the results show an increase in precision and MRR; we actually expected a greater improvement. However, we believe that the potential improvement is actually greater than what is shown. For now, the results are limited to the syntax of the patterns currently implemented in the QA system: the tags in the patterns that are recognized are very limited, preventing a greater granularity of patterns. For example,

currently there is no differentiation between past, past participle, or third person verbs. This makes most of the new reformulated patterns we extracted not recognizable by the QA component.

Question type	Nb of questions	Nb of questions with a correct answer in the top 5 candidates	Precision of candidate list
who	52	20	0.571
what	266	42	0.500
where	39	8	0.533
when	71	11	0.687
how + adj/adv	53	5	0.277
which	12	0	0
Total	493	92	0.538

Table 2. Results for each question category with the original hand-crafted patterns

Question type	Nb of questions	Nb of questions with a correct answer in the top 5 candidates	Precision of candidate list
who	52	24	0.648
what	266	42	0.552
where	39	11	0.578
when	71	18	0.720
how + adj/adv	53	6	0.462
which	12	0	0
Total	493	113	0.646

Table 3. Results for each question category with the generated patterns

6 Conclusion and Future Work

We presented a method for acquiring reformulations patterns automatically based on both syntactic and semantic features. The experimental evaluation of our reformulations shows that using new generated patterns does not increase the precision and MRR significantly compared to hand-crafted rules, but do eliminate the need for human intervention.

As opposed to several other approaches that reinforce their candidate answers by looking on the Web; 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. On the other hand, as we look for evidence that fit a certain pattern with many possible words fitting the pattern, rather that a

Question Type	Frequency	Hand-crafted patterns		Automatic patterns	
		MRR	Precision	MRR	Precision
who	52 (10.4%)	0.301	0.571	0.396	0.648
what	266 (53.2%)	0.229	0.500	0.317	0.552
where	39 (7.8%)	0.500	0.533	0.348	0.578
when	71 (14.2%)	0.688	0.687	0.643	0.720
how + adj/adv	53 (10.6%)	0.194	0.277	0.310	0.462
which	12 (2.4%)	0	0	0	0

Table 4. The results based on question categories.

strict string match, we are more sensitive to mistakes and wrong interpretations. Indeed, 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, and we will not catch it. When looking in a very large corpus such as the Web, this may lead to more noise than a strict lexical string match approach. 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 arguments. However, it can easily be extend 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. An careful evaluation would be necessary to ensure that the approach does not introduce too many constraints and consequently filters out too many candidates.

Our approach to checking syntactic and morpho-suyntactic variations (as described in section 4.4) is very crude: we only check for the presence or absence of a similar a stem with no respect to syntax. A more precise approach should be used; see the work of [23–25] for example.

Finally, to improve the quality of the patterns, we suggest a systematic evaluation and adjustment of the parameters that take part in weighting the patterns; for example, the size of the windows and the sub-phrase scoring.

Acknowledgments

References

1. Aceves-Pérez, R., nor Pineda, L.V., Montes-y-Gòmez, 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)
2. Hermjakob, U., Echihabi, A., Marcu, D.: Natural language based reformulation resource and wide exploitation for question answering. [22]

3. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proceedings of ACL-2002, Philadelphia (2002)
4. Agichtein, E., Gravano, L.: Snowball: Extracting Relations from Large Plain-Text Collections. In: Proceedings of the 5th ACM International Conference on Digital Libraries. (2000)
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. Soubbotin, M., Soubbotin, S.: Patterns of potential answer expressions as clues to the right answers. In: Proceedings of The Tenth Text Retrieval Conference (TREC-X), Gaithersburg, Maryland (2001) 175–182
7. Brill, E., Lin, J., Banko, M., Dumais, S., Ng, A.: Data-Intensive Question Answering. [19] 393–400
8. Lawrence, S., Giles, C.L.: Context and Page Analysis for Improved Web Search. IEEE Internet Computing **2** (1998) 38–46
9. Rivachandram, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proceeding of ACL Conference, Philadephia (2002) 41–47
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: 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. [22]
15. Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities **36** (2002) 223–254
16. 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
17. NIST: Proceedings of TREC-8, Gaithersburg, Maryland, NIST (1999)
18. NIST: Proceedings of TREC-9, Gaithersburg, Maryland, NIST (2000)
19. NIST: Proceedings of TREC-10, Gaithersburg, Maryland, NIST (2001)
20. Plamondon, L., Lapalme, G., Kosseim, L.: The QUANTUM Question Answering System. [22]
21. Porter, M.: An algorithm for suffix stripping. Program **14** (1980) 130–137
22. NIST: Proceedings of TREC-11, Gaithersburg, Maryland, NIST (2002)
23. Jacquemin, C.: Spotting and discovering terms through natural language processing. MIT Press, Cambridge, Mass. (2001)
24. Ferro, J.V., Barcala, F.M., Alonso, M.A.: Using syntactic dependency-pairs conflation to improve retrieval performance in spanish. In: CICLing. (2002) 381–390
25. Ribadas, F.J., Vilares, M., Vilares, J.: Semantic similarity between sentences through approximate tree matching. In Jorge S. Marques, N.P.d.l.B., Pina, P., eds.: Pattern Recognition and Image Analysis. Volume 3523 of Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg-New York (2005) 638–646