Information Extraction for Question Answering: Improving Recall Through Syntactic Patterns

Valentin Jijkoun and **Maarten de Rijke**
Informatics Institute
University of Amsterdam
{jijkoun,mdr}@science.uva.nl

Jori Mur
Information Science
University of Groningen
mur@let.rug.nl

Abstract

We investigate the impact of the precision/recall trade-off of information extraction on the performance of an offline corpus-based question answering (QA) system. One of our findings is that, because of the robust final answer selection mechanism of the QA system, recall is more important. We show that the recall of the extraction component can be improved using syntactic parsing instead of more common surface text patterns, substantially increasing the number of factoid questions answered by the QA system.

1 Introduction

Current retrieval systems allow us to locate documents that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a ranked list of documents. Hence, the (often unwilling) user is left with a relatively large amount of text to consume. There is a need for tools that reduce the amount of text one might have to read to obtain the desired information. *Corpus-based question answering* is designed to take a step closer to *information* retrieval rather than *document* retrieval. The question answering (QA) task is to find, in a large collection of data, an answer to a question posed in natural language.

One particular QA strategy that has proved successful on large collections uses surface patterns derived from the question to identify answers. For example, for questions like *When was Gandhi born?*, typical phrases containing the answer are *Gandhi was born in 1869* and *Gandhi (1869–1948)*. These examples suggest that text patterns such as "`name` *was born in* `birth date`" and "`name` (`birth year`–`death year`)" formulated as regular expressions, can be used to select the answer phrase.

Similarly, such lexical or lexico-syntactic patterns can be used to extract specific information on semantic relations from a corpus offline, before actual questions are known, and store it in a repository for quick and easy access. This strategy allows one to handle some frequent question types: *Who is...*, *Where is...*, *What is the capital of...* etc. (Fleischman et al., 2003; Jijkoun et al., 2003).

A great deal of work has addressed the problem of extracting semantic relations from unstructured text. Building on this, much recent work in QA has focused on systems that extract answers from large bodies of text using simple lexico-syntactic patterns. These studies indicate two distinct problems associated with using patterns to extract semantic information from text. First, the patterns yield only a small subset of the information that may be present in a text (the *recall* problem). Second, a fraction of the information that the patterns yield is unreliable (the *precision* problem). The precision of the extracted information can be improved significantly by using machine learning methods to filter out noise (Fleischman et al., 2003). The recall problem is usually addressed by increasing the amount of text data for extraction (taking larger collections (Fleischman et al., 2003)) or by developing more surface patterns (Soubbotin and Soubbotin, 2002).

Some previous studies indicate that in the setting of an end-to-end state-of-the-art QA system, with additional answer finding strategies, sanity checking, and statistical candidate answer re-ranking, recall is more of a problem than precision (Bernardi et al., 2003; Jijkoun et al., 2003): it often seems useful to have *more* data rather than *better* data. The aim of this paper is to address the recall problem by using extraction methods that are linguistically more sophisticated than surface pattern matching. Specifically, we use dependency parsing to extract syntactic relations between entities in a text, which are not necessarily adjacent on the surface level. A small set of hand-built syntactic patterns allows us to detect relevant semantic information. A comparison of the parsing-based approach to a surface-pattern-based method on a set of TREC questions about persons shows a substantial improvement in the amount of the extracted information and number of correctly answered questions.

In our experiments we tried to understand whether linguistically involved methods such as parsing can be beneficial for information extraction, where rather shallow techniques are traditionally employed, and whether the abstraction from surface to syntactic structure of the text does indeed help to

find more information, at the same time avoiding the time-consuming manual development of increasing numbers of surface patterns.

The remainder of the paper is organized as follows. In Section 2 we discuss related work on extracting semantic information. We describe our main research questions and experimental setting in Section 3. Then, in Section 4 we provide details on the extraction methods used (surface and syntactic). Sections 5 and 6 contain a description of our experiments and results, and an error analysis, respectively. We conclude in Section 7.

2 Related Work

There is a large body of work on extracting semantic information using lexical patterns. Hearst (1992) explored the use of lexical patterns for extracting hyponym relations, with patterns such as "such as." Berland and Charniak (1999) extract "part-of" relations. Mann (2002) describes a method for extracting instances from text by means of part-of-speech patterns involving proper nouns.

The use of lexical patterns to identify answers in corpus-based QA received lots of attention after a team taking part in one of the earlier QA Tracks at TREC showed that the approach was competitive at that stage (Soubbotin and Soubbotin, 2002; Ravichandran and Hovy, 2002). Different aspects of pattern-based methods have been investigated since. E.g., Ravichandran et al. (2003) collect surface patterns automatically in an unsupervised fashion using a collection of trivia question and answer pairs as seeds. These patterns are then used to generate and assess answer candidates for a statistical QA system. Fleischman et al. (2003) focus on the precision of the information extracted using simple part-of-speech patterns. They describe a machine learning method for removing noise in the collected data and showed that the QA system based on this approach outperforms an earlier state-of-the-art system. Similarly, Bernardi et al. (2003) combine the extraction of surface text patterns with WordNet-based filtering of name-apposition pairs to increase precision, but found that it hurt recall more than it helped precision, resulting in fewer questions answered correctly when the extracted information is deployed for QA.

The application of deeper NLP methods has also received much attention in the QA community. The open-domain QA system by LCC (Moldovan et al., 2002) uses predicate-argument relations and lexical chaining to actually *prove* that a text snippet provides an answer to a question. Katz and Lin (2003) use syntactic dependency parsing to extract relations between words, and use these relations rather than individual words to retrieve sentences relevant to a question. They report a substantial improvement for certain types of questions for which the usual term-based retrieval performs quite poorly, but argue that deeper text analysis methods should be applied with care.

3 Experimental Setting

We set up experiments to address two related issues. First, we wanted to understand how the usual precision/recall trade-off shows up in off-line corpus-based QA, and specifically, whether extracting more data of lower quality (i.e., favoring recall) gives a QA system a better performance than extracting smaller amounts of more accurate data (i.e., favoring precision). Second, we tried to verify the hypothesis that syntactic parsing for information extraction does increase the extraction recall by identifying relations between entities not adjacent on the surface layer but connected syntactically.

There are different approaches to the evaluation of information extraction modules. The usual recall and precision metrics (e.g., how many of the interesting bits of information were detected, and how many of the found bits were actually correct) require either a test corpus previously annotated with the required information, or manual evaluation (Fleischman et al., 2003). Although intrinsic evaluation of an IE module is important, we were mainly interested in measuring the performance of this module in context, that is, working as a sub-part of a QA system. We used the number of questions answered correctly as our main performance indicator.

3.1 QA System

For the experiments described below we used an open-domain corpus-based QA system QUARTZ (Jijkoun et al., 2004). The system implements a multi-stream approach, where several different strategies are used in parallel to find possible answers to a question. We ran the system turning on only one stream, *Table Lookup*, which implements an off-line strategy for QA.

The *Table Lookup* stream uses a number of knowledge bases created by pre-processing a document collection. Currently, QUARTZ' knowledge bases include 14 semi-structured tables containing various kinds of information: birth dates of persons, dates of events, geographical locations of different objects, capitals and currencies of countries, etc. All this information is extracted from the corpus off-line, before actual questions are known.

An incoming question is analyzed and assigned to one of 37 predefined question types. Based on

the question type, the *Table Lookup* stream identifies knowledge bases where answers to the question can potentially be found. The stream uses keywords from the question to identify relevant entries in the selected knowledge bases and extracts candidate answers. Finally, the QA system reranks and sanity checks the candidates and selects the final answer.

3.2 Questions and Corpus

To get a clear picture of the impact of using different information extraction methods for the offline construction of knowledge bases, similarly to (Fleischman et al., 2003), we focused only on questions about persons, taken from the TREC-8 through TREC 2003 question sets. The questions we looked at were of two different types: person identification (e.g., *2301. What composer wrote "Die Götterdämmerung"?*) and person definition (e.g., *959. Who was Abraham Lincoln?*). The knowledge base relevant for answering questions of these types is a table with several fields containing a person name, an information bit about the person (e.g., occupation, position, activities), the confidence value assigned by the extraction modules to this information bit (based on its frequency and the reliability of the patterns used for extraction), and the source document identification. The *Table Lookup* finds the entries whose relevant fields best match the keywords from the question.

We performed our experiments with the 336 TREC questions about persons that are known to have at least one answer in the collection. The collection used at TREC 8, 9 and 10 (referred to as *TREC-8* in the rest of the paper) consists of 1,727,783 documents, with 239 of the corresponding questions identified by our system as asking about persons. The collection used at TREC 2002 and 2003 (*AQUAINT*) contains 1,033,461 documents and 97 of the questions for these editions of TREC are person questions.

4 Extraction of Role Information

In this section we describe the two extraction methods we used to create knowledge bases containing information about persons: extraction using surface text patterns and using syntactic patterns.

Clearly, the performance of an information extraction module depends on the set of language phenomena or patterns covered, but this relation is not straightforward: having more patterns allows one to find more information, and thus increases recall, but it might introduce additional noise that hurts precision. Since in our experiments we aimed at comparing extraction modules based on surface text vs. syntactic patterns, we tried to keep these two modules parallel in terms of the phenomena covered.

First, the collections were tagged with a Named Entity tagger based on TnT (TnT, 2003) and trained on CoNLL data (CoNLL, 2003). The Named Entity tagger was used mainly to identify person names as separate entities. Although the tagging itself was not perfect, we found it useful for restricting our surface text patterns.

Below we describe the two extraction methods.

4.1 Extraction with Surface Text Patterns

To extract information about roles, we used the set of surface patterns originally developed for the QA system we used at TREC 2003 (Jijkoun et al., 2004). The patterns are listed in Table 1.

In these patterns, `person` is a phrase that is tagged as person by the Named Entity tagger, `role` is a word from a list of roles extracted from the WordNet (all hyponyms of the word 'person,' 15703 entries),[1] `role-verb` is from a manually constructed list of "important" verbs (*discovered, invented*, etc.; 48 entries), *leader* is a phrase identifying leadership from a manually created list of leaders (*president, minister*, etc.; 22 entries). Finally, `superlat` is the superlative form of an adjective and `location` is a phrase tagged as location by the Named Entity tagger.

4.2 Extraction with Syntactic Patterns

To use the syntactic structure of sentences for role information extraction, the collections were parsed with Minipar (Lin, 1998), a broad coverage dependency parser for English. Minipar is reported to achieve 88% precision and 80% recall with respect to dependency relations when evaluated on the SUSANNE corpus. We found that it performed well on the newpaper and newswire texts of our collections and was fairly robust to fragmented and not well-formed sentences frequent in this domain. Before extraction, Minipar's output was cleaned and made more compact. For example, we removed some empty nodes in the dependency parse to resolve non-local dependencies. While not loosing any important information, this made parses easier to analyse when developing patterns for extraction.

Table 2 lists the patterns that were used to extract information about persons; we show syntactic dependencies as arrows from dependents to heads, with Minipar's dependency labels above the arrows.

As with the earlier surface patterns, `role` is one of the nouns in the list of roles (hyponyms of *person*

[1] The list of roles is used to increase precision by filtering out snippets that may not be about roles; in some of the experiments below, we turn this filtering mechanism off.

Pattern	Example
`... role, person`	*The British actress, Emma Thompson*
`... (superlat|`*first*`|`*last*`)..., person`	*The first man to set foot on the moon, Armstrong*
`person,... role...`	*Audrey Hepburn, goodwill ambassador for UNICEF.*
`person,... (superlat|`*first*`|`*last*`)...`	*Brown, Democrats' first black chairman.*
`person,... role-verb...`	*Christopher Columbus, who discovered America,*
`... role person`	*District Attoney Gil Garcetti*
`role... person`	*The captain of the Titanic Edward John Smith*
`person,... leader... location`	*Tony Blair, the prime minister of England*
`location... leader, person`	*The British foreign secretary , Jack Straw*

Table 1: Surface patterns.

Pattern	Example
Apposition `person` $\xrightarrow{\text{appo}}$ `role`	*a major developer, Joseph Beard*
Apposition `person` $\xleftarrow{\text{appo}}$ `role`	*Jerry Lewis, a Republican congressman*
Clause `person` $\xrightarrow{\text{subj}}$ `role-verb`	*Bell invented the telephone*
Person `person` $\xrightarrow{\text{person}}$ `role`	*Vice President Al Gore*
Nominal modifier `person` $\xleftarrow{\text{nn}}$ `role`	*businessman Bill Shockley*
Subject `person` $\xrightarrow{\text{subj}}$ `role`	*Alvarado was chancellor from 1983 to 1984*
Conjunction `person` $\xleftarrow{\text{conj}}$ `role` (this is a frequent parsing error)	*Fu Wanzhong, director of the Provincial Department of Foreign Trade*

Table 2: Syntactic patterns.

in WordNet), `role-verb` is one of the "important verbs." The only restriction for `person` was that it should contain a proper noun.

When an occurence of a pattern was found in a parsed sentence, the relation (`person`; `info-bit`) was extracted, where `info-bit` is a sequence of all words below `role` or `role-verb` in the dependency graph (i.e., all dependents along with their dependents etc.), excluding the `person`. For example, for the sentence *Jane Goodall, an expert on chimps, says that evidence for sophisticated mental performances by apes has become ever more convincing*, that matches the pattern `person` $\xleftarrow{\text{appo}}$ `role`, the extracted information was (*Jane Goodall*; *an expert on chimps*).

5 Experiments and Results

We ran both surface pattern and syntactic pattern extraction modules on the two collections, with a switch for role filtering. The performance of the *Table Lookup* stream of our QA system was then evaluated on the 336 role questions using the answer patterns provided by the TREC organizers. An early error analysis showed that many of the incorrect answers were due to the table lookup process (see Section 3) rather than the information extraction method itself: correct answers were in the tables, but the lookup mechanism failed to find them or picked up other, irrelevant bits of information. Since we were interested in evaluating the two extraction methods rather than the lookup mechanism, we performed another experiment: we reduced the sizes of the collections to simplify the automatic lookup. For each TREC question with an answer in the collection, NIST provides a list of documents that are known to contain an answer to this question. We put together the document lists for all questions, which left us with much smaller sub-collections (16.4 MB for the questions for the TREC-8 collection and 3.2 MB for the AQUAINT collection). Then, we ran the two extraction modules on these small collections and evaluated the performance of the QA system on the resulting tables. All the results reported below were obtained with these sub-collections. Comparison of the extraction modules on the full TREC collections gave very similar relative results.

Table 3 gives the results of the different runs for the syntactic pattern extraction and the surface pattern extraction on the TREC-8 collection: the number of correct answers (in the top one and the top three answer candidates) for the 239 person questions. The columns labeled *Roles* + show the results for the extraction modules using the list of possible roles from WordNet (Section 4), and the columns labeled *Roles* − show the results when the extraction modules consider *any* word as possibly denoting a role. The results of the runs on the AQUAINT collection with 97 questions are shown in Table 4.

The syntactic pattern module without role filtering scored best of all, with more than a third of the

	Syntactic patterns		Surface patterns	
Rank	Roles −	Roles +	Roles −	Roles +
1	80 (34%)	73 (31%)	59 (25%)	54 (23%)
1–3	90 (38%)	79 (33%)	68 (29%)	59 (25%)

Table 3: Correct answers for the TREC-8 collection (239 questions).

	Syntactic patterns		Surface patterns	
Rank	Roles −	Roles +	Roles −	Roles +
1	16 (17%)	14 (14%)	9 (9%)	6 (6%)
1–3	20 (21%)	14 (14%)	11 (11%)	6 (6%)

Table 4: Correct answers for the AQUAINT collection (97 questions).

questions answered correctly for the TREC-8 collection. Another interesting observation is that in all experiments the modules based on syntactic patterns outperformed the surface-text-based extraction.

Furthermore, there is a striking difference between the results in Table 3 (questions from TREC 8, 9 and 10) and the results in Table 4 (questions from TREC 2002 and 2003). The questions from the more recent editions of TREC are known to be much harder: indeed, the *Table Lookup* stream answers only 21% of the questions from TREC 2002 and 2003, vs. 38% for earlier TRECs.

In all experiments, both for syntactic and surface patterns, using the list of roles as a filtering mechanism decreases the number of correct answers. Using lexical information from WordNet improves the precision of the extraction modules less than it hurts the recall. Moreover, in the context of our knowledge base lookup mechanism, low precision of the extracted information does not seem to be an obstacle: the irrelevant information that gets into the tables is either never asked for or filtered out during the final sanity check and answer selection stage. This confirms the conclusions of (Bernardi et al., 2003): in this specific task having *more* data seems to be more useful than having *better* data.

To illustrate the interplay between the precision and recall of the extraction module and the performance of the QA system, Table 5 gives the comparison of the different extraction mechanisms (syntactic and surface patterns, using or not using the list of roles for filtering). The row labelled *# facts* shows the size of the created knowledge base, i.e., the number of entries of the form (*person*, *info*), extracted by each method. The row labelled *Precision* shows the precision of the extracted information (i.e., how many entries are correct, according to a human annotator) estimated by random sampling and manual evaluation of 1% of the data for each table, similar to (Fleischman et al., 2003). The row labelled *Corr. answers* gives the number of questions correctly answered using the extracted information.

	Syntactic patterns		Surface patterns	
	Roles −	Roles +	Roles −	Roles +
# facts	29890	9830	28803	6028
Precision	54%	61%	23%	68%
Corr. answers	34%	31%	25%	23%

Table 5: Comparison of the tables built with different extraction methods on the TREC-8 collection.

The results in Table 5 indicate that role filtering affects the syntactic and surfaces modules quite differently. Filtering seems almost essential for the surface-pattern-based extraction, as it increases the precision from 23% to 68%. This confirms the results of Fleischman et al. (2003): shallow methods may benefit significantly from the post-processing. On the other hand, the precision improvement for the syntactic module is modest: from 54% to 61%.

The data from the syntactic module contains much less noise, although the sizes of the extracted tables before role filtering are almost the same. After filtering, the number of valid entries from the syntactic module (i.e., the table size multiplied by the estimated precision) is about 6000. This is substantially better than the recall of the surface module (about 4100 valid entries).

6 Error Analysis

In theory, all relatively simple facts extracted by the surface pattern module should also be extracted by the syntactic pattern module. Moreover, the syntactic patterns should extract more facts, especially ones whose structure deviates from the patterns predefined in the surface pattern module, e.g., where elements adjacent in the syntactic parse tree are far apart on the surface level. To better understand the differences between the two extraction approaches and to verify the conjecture that syntactic parsing does indeed increase the recall of the extracted information, we performed a further (manual) error analysis, identifying questions that were answered with one extraction method but not with the other.

Tables 6 and 7 gives the breakdown of the performance of the two modules, again in terms of the questions answered correctly. We show the results for the 239 questions on the TREC-8 collection; for the 97 questions on the AQUAINT corpus the relative scores are similar. As Tables 6 and 7 indicate, not all questions answered by the surface pattern module were also answered by the syntactic pattern module, contrary to our expectations. We took a closer look at the questions for which the two modules performed differently.

		Syntactic patterns	
		correct	incorrect
Surface patterns	correct	47	12
	incorrect	32	148

Table 6: Performance analysis for the TREC-8 collection with role filtering.

		Syntactic patterns	
		correct	incorrect
Surface patterns	correct	51	17
	incorrect	39	132

Table 7: Performance analysis for the TREC-8 collection without role filtering.

6.1 Syntactic Patterns vs. Surface Patterns

There were three types of errors responsible for producing an incorrect answer by the syntactic pattern module for questions correctly answered with surface patterns. The most frequent errors were parsing errors. For 6 out of 12 questions (see Table 6) the answer was not extracted by the syntactic pattern method, because the sentences containing the answers were not parsed correctly. The next most frequent error was caused by the table lookup process. For 4 questions out of the 12, the required information was extracted but simply not selected from the table as the answer due to a failure of the lookup algorithm. The remaining errors (2 out of 12) were of a different type: for these 2 cases the surface pattern extraction did perform better than the syntactic method. In both cases this was because of wildcards allowed in the surface patterns. E.g., for the sentence ... *aviator Charles Lindbergh married Anne Spencer Morrow...* the syntactic pattern method extracted only the relation

(*Charles Lindbergh*; *aviator*),

whereas the surface pattern method also extracted

(*Anne Spencer Morrow*; *aviator Charles Lindbergh married*),

because of the pattern '`role... person`' with `role` instantiated with *aviator* and `person` with *Anne Spencer Morrow*. In fact, the extracted information is not even correct, because Anne is not an aviator but Lindbergh's wife. However, due to the fuzzy nature of the lookup mechanism, this new entry in the knowledge base allows the QA system to answer correctly the question *646. Who was Charles Lindbergh's wife?*, which is not answered with the syntactic pattern extraction module.

To summarize, of the 12 questions where the surface patterns outperformed the syntactic patterns

- 6 questions were not answered by the syntactic method due to parsing errors,
- 4 were not answered because of the table lookup failure and
- for 2 the surface-based method was more appropriate.

6.2 Surface Patterns vs. Syntactic Patterns

We also took a closer look at the 32 questions for which the syntactic extraction performed better than the surface patterns (see Table 6). For the surface pattern extraction module there were also three types of errors. First, some patterns were missing, e.g., `person role-verb...`. The only difference from one of the actually used patterns (`person,... role-verb...`) is that there is no comma between `person` and `role-verb`. This type of incompleteness of the set of the surface patterns was the cause for 16 errors out of 32.

The second class of errors was caused by the Named Entity tagger. E.g., *Abraham Lincoln* was always tagged as `location`, so the name never matched any of the surface patterns. Out of 32 questions, 10 were answered incorrectly for this reason.

Finally, for 6 questions out of 32, the syntactic extraction performed better because the information could not be captured on the surface level. For example, the surface pattern module did not extract the fact that Oswald killed Kennedy from the sentence ... *when Lee Harvey Oswald allegedly shot and killed President John F. Kennedy...*, because none of the patterns matched. Indeed, *Lee Harvey Oswald* and the potentially interesting verb *killed* are quite far apart in the text, but there is an immediate relation (subject) on the syntactic level.

It is worth pointing out that there were no lookup errors for the surface pattern method, even though it used the exact same lookup mechanism as the approach based on syntactic patterns (that did experience various lookup errors, as we have seen). It seems that the increased recall of the syntactic pattern approach caused problems by making the lookup process harder.

To summarize, out of 32 questions answered using syntactic extraction method but not by the surface pattern approach

- 16 questions would have required extending the set of surface patterns,
- 10 questions were not answered because of NE tagging error, and
- 6 questions required syntactic analysis for extraction of the relevant information.

6.3 Adding Patterns?

We briefly return to a problem noted for extraction based on surface patterns: the absence of certain surface patterns. The surface pattern `person role-verb...` was not added because, we felt, it would introduce too much noise in the knowledge base. With dependency parsing this is not an issue as we can require that `person` is the subject of `role-verb`. So in this case the syntactic pattern module has a clear advantage. More generally, while we believe that extraction methods based on hand-crafted patterns are necessarily incomplete (in that they will fail to extract certain relevant facts), these observations suggest that coping with the incompleteness is a more serious problem for the surface patterns than for the syntactic ones.

7 Conclusions

We described a set of experiments aimed at comparing different information extraction methods in the context of off-line corpus-based Question Answering. Our main finding is that a linguistically deeper method, based on dependency parsing and a small number of simple syntactic patterns, allows an off-line QA system to correctly answer substantially more questions than a traditional method based on surface text patterns. Although the syntactic method showed lower precision of the extracted facts (61% vs. 68%), in spite of parsing errors the recall was higher than that of the surface-based method, judging by the number of correctly answered questions (31% vs. 23%). Thus, the syntactic analysis can in fact be considered as another, *intensive* way of improving the recall of information extraction, in addition to successfully used *extensive* ways, such as developing larger numbers of surface patterns or increasing the size of the collection.

Moreover, we confirmed the claim that for a complex off-line QA system, with statistical as well as knowledge-intensive sanity checking answer selection modules, recall of the information extraction module is more important than precision, and a simple WordNet-based method for improving precision does not help QA. In our future work we plan to investigate the effect of more sophisticated and, probably, more accurate filtering methods (Fleischman et al., 2003) on the QA results.

8 Acknowledgements

Valentin Jijkoun and Maarten de Rijke were supported by a grant from the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. De Rijke was also supported by NWO under project numbers 365-20-005, 612.069.006, 612.000.106, 612.000.207, and 612.066.302.

References

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*.

R. Bernardi, V. Jijkoun, G. Mishne, and M. de Rijke. 2003. Selectively using linguistic resources throughout the question answering pipeline. In *Proceedings of the 2nd CoLogNET-ElsNET Symposium*.

M. Fleischman, E. Hovy, and A. Echihabi. 2003. Offline strategies for online question answering: answering questions before they are asked. In *Proceedings of the 41st Annual Meeting of the ACL*.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*.

V. Jijkoun, G. Mishne, and M. de Rijke. 2003. Preprocessing Documents to Answer Dutch Questions. In *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)*.

V. Jijkoun, G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. 2004. The University of Amsterdam at the TREC 2003 Question Answering Track. In *Proceedings of the TREC-2003 Conference*.

B. Katz and J. Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*.

D. Lin. 1998. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*.

G. Mann. 2002. Fine-grained proper noun ontologies for question answering. In *SemaNet'02: Building and Using Semantic Networks*.

D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, A. Novischi F. Lacatusu, A. Badulescu, and O. Bolohan. 2002. LCC tools for question answering. In *Proceedings of the TREC-2002*.

TnT Statistical Part of Speech Tagging. 2003. URL: `http://www.coli.uni-sb.de/~thorsten/tnt/`.

CoNLL: Conference on Natural Language Learning. 2003. URL: `http://cnts.uia.ac.be/signll/shared.html`.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the ACL*.

D. Ravichandran, A. Ittycheriah, and S. Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of the HLT-NAACL Conference*.

M.M. Soubbotin and S.M. Soubbotin. 2002. Use of patterns for detection of likely answer strings: A systematic approach. In *Proceedings of the TREC-2002 Conference*.