Question Answering using Unification-based Grammar*

Vlado Kešelj

Department of Computer Science, University of Waterloo,
Waterloo, ON N2L 3G1, Canada,
vkeselj@cs.uwaterloo.ca, http://www.cs.uwaterloo.ca/~vkeselj

Abstract. The problem of Question Answering (QA) as used in TREC [13] can be formulated as follows:

> Given a collection of natural-language (NL) documents find an answer to given NL query that is a short substring of one of the documents, and it is found in a relevant context.

We present a novel approach to the problem based on unification-based grammars for NL.

Key words: question answering, stochastic unification-based grammars.

1 Introduction

The traditional approach to the problem of QA assumes existence of a knowledge base, which is used to produce answers to NL questions. The problem, defined in this way, is too difficult for the current state of the art in natural language processing (NLP). Schwitter *et al.* 2000 [11] argue that even an easy comprehension test performed on a simple story of three paragraphs, titled "How maple syrup is made," is a too difficult task.

We use an easier version of QA, in which the system returns only a short document substring that contains an answer. This problem formulation is used in the TREC competition, where the QA track is introduced in 1999. Some authors (Mollá *et al.* 1998 [8]) call this version of QA *Answer Extraction,* which is probably a more precise name of the task.

There are four related NL-based retrieval tasks:

- classical information retrieval (IR)—the task of retrieving relevant documents from a collection,
- information extraction (IE)—the task of collecting specific content data,
- QA or answer extraction (AE)—the task of locating an answer, and
- classical QA—the task of providing an intelligible answer to a question using a knowledge base.

* This work is supported by NSERC and OGS.

Let us give more details about the TREC-8 QA task, which is the testbed for our approach. TREC-8 included 198 questions. There were initially 200 questions, but later evaluation has shown that the answers to 2 questions could not be found in the collection. All questions can be successfully answered by relatively short strings—the longest minimal answer has length of 35 bytes. The participants can submit a run of 50-byte or a run of 250-byte strings. Five ranked strings are submitted for each question. A submission is ranked according to the *reciprocal answer rank* (RAR); i.e., 1 mark is given if a correct answer is found in the first string, otherwise 1/2 marks are given for a correct answer in the second string, 1/3 for third string, 1/4 for the fourth string, or 1/5 for a correct answer in the fifth string. Only the highest ranked answer is taken into consideration. Zero mark is given for not having an answer. The final evaluation score for one run is the *mean reciprocal answer rank* (MRAR); i.e., the average of the per-question scores.

Here are three examples from TREC-8. The first question and its satisfiable answer are:

> Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher?"
> — Young

The question with the longest minimal answer is:

> Why did David Koresh ask the FBI for a word processor?
> — To record his revelations

And, a question for which our "classical" IR system could not find an answer is:

> When was London's Docklands Light Railway constructed?
> — 1987

A typical approach to QA consists of two phases: First, some relevant passages are extracted using a standard IR technique. Second, the fine-grained answer extraction is done using a shallow NLP technique. This NLP technique incorporates a shallow semantic representation of the text and the query. We describe a novel approach to QA based on a unification-based grammar.

2 Related Work

QA has received a lot of attention recently. The commercial site AskJeeves [1] offers a service where users can ask NL questions. If an answer is not found, the system offers similar questions for which it knows the answer. We have tried an example session with the question

> When did Beethoven die?

The question is interesting since it appeared in TREC-8, and it is a general question. The system could not find an answer, but it suggested a list of about dozen questions for which it knew some answers. Some of them were:

- Where can I find albums with the music of Beethoven?
- Where can I buy a biography of Beethoven on video or DVD?

  ...
- How can I make my end of life wishes known?
- How will I die?

AskJeeves represents a case-based approach where case problems are question instances and solutions are associated answers. A similar case-based approach is used in FAQ Finder (Burke *et al.* 97 [2]). FAQ Finder uses shallow NLP with a simple lexical semantic method based on WordNet. Given a question, FAQ Finder uses the classical SMART system to retrieve FAQ files relevant to the question, and then using a combination of statistical IR and semantic matching, the relevant question-answer pairs are extracted.

Shallow lexical semantics is also used in the method of *predictive annotation* (Radev *et al.* 2000 [9]) used in TREC-8. In this approach, the questions and the collection documents are annotated with *QA-tokens,* like `DATE$` and `PLACE$`. For example, we can easily see how the classical IR fails with the question: "Where did *smth* happen?" The word 'Where' is crucial for answering the question, yet it is removed as a stop-word. In the predictive annotation approach, the word 'Where' is replaced with the QA-token `PLACE$`, which not only prevents it from being deleted as a stop-word, but it also increases its importance weight. Pattern matching is used to annotate queries. The documents are also annotated with QA-tokens. The two-phase approach is used, consisting of the first IR phase and the second answer-selection phase.

Deeper semantic processing is applied in the system ExtrAns (Mollá *et al.* 98 [8])—a QA system for a set of Unix man pages. ExtrAns is implemented in Prolog. Horn clauses are used for semantic representation of the query and the text. Finding an answer to a query is modeled as "proving" the query using the facts from the text. For example, the sentence

cp copies the contents of *filename1* onto *filename2*.

is translated into the following set of facts:

```
holds(e1)/s1. object(cp,o1,x1)/s1.
object(command,o2,x1)/s1.   evt(copy,e1,[x1,x2])/s1.
object(content,o3,x2)/s1.   object(filename1,o4,x3)/s1.
object(file,o5,x3)/s1.      of(x2,x3)/s1.
object(filename2,o6,x4)/s1. object(file,o7,x4)/s1.
onto(e1,x4)/s1.
```

The question

Which command copies files?

is translated into the query:

```
object(command,_,X)/S,
(evt(copy,E,[X,Y])/S; evt(duplicate,E,[X,Y])/S),
object(file,_Y)/S)
```

The answer is located by the standard Prolog resolution procedure.

This QA approach is brittle, so a multi-level fall-back strategy for graceful degradation is introduced. If the system cannot find an answer, then some contraints are made weaker and the procedure is repeated. If an answer is not found again, more constraints are weakened or removed, and so on. Finally, a keyword-based approach is applied at the last level of the strategy.

In section 4 we present a novel unification-based approach to QA, which is based on the work in unification based-grammars [12]: in particular, the HPSG grammars [10] and attribute-value logic [3].

3 Performance of a classical IR system

As in most other approaches (e.g., [9] and [2]), we also use two phases in QA. Processing the whole document collection would be too expensive for NLP, so in the first phase we rely on the MultiText retrieval engine (Cormack *et al.* 1999 [4]). MultiText engine retrives a ranked list of 25 relevant passages for each question. Each passages has approximately the length of 2000 bytes.

The MultiText engine chooses passages in such way that the middle part of each passage most likely contains the answer to the question. Hence, if we simply extract the middle part of each passage, we have a reasonable solution to QA.

In order to later evaluate our system, we first measure the performace of this simple extraction method on the input passages. The evaluation results are presented in figure 1. The x-axis on the figure presents the answer lengths in bytes, and the y-axis is the score, i.e., MRAR. The answer length varies from 35 bytes, which was the shortest minimal answer on TREC-8, to 2000 bytes. Two answer lengths are marked with two vertical lines on the figure: 50 bytes and 250 bytes. Those were the lengths accepted at TREC-8.

If we merge all 25 passages for each question, and return this large 50000-byte passage as an answer, the score of the run would be 0.91. This is represented by the first horizontal line on the figure. This is an upper bound on the performance of the NLP part, since the passages returned by the search engine do not contain answers to 18 questions ($\frac{1\times 180+0\times 18}{198} \approx 0.91$).

The next lower horizontal line ($y = 0.83$) is obtained by merging first 5 passages for each question (10000 bytes) and treating this as an answer. The significant difference between this line and the previous line shows that a significant number of answers is not located in the first 5 passages, so all 25 passages are important for our task.

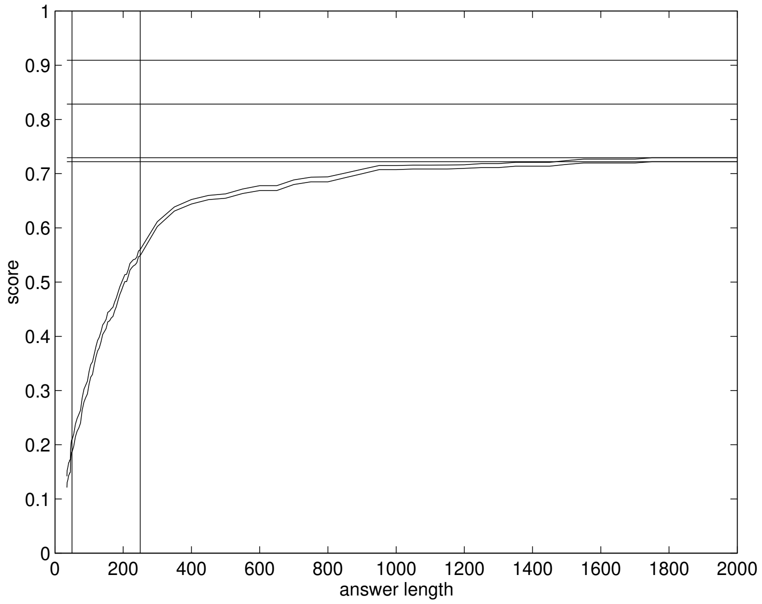

Fig. 1. Simple extraction method

The next two lower horizontal lines (0.73 and 0.72) are the scores obtained by submitting all and the first five 2000-byte passages as answer strings, respectively. The difference between the numbers is not big since even when the first correct answer is below the 5th submitted passage, its reciprocal rank adds less then $0.2 = 1/5$ marks. A significant difference between the 50 000-byte line (0.91) and these lines implies that the performance can be significantly improved just by a better ranking of the passages.

Below these horizontal lines, there are two monotonically increasing curves, which represent the scores of the strings submitted by the simple extraction method. The lower curve is for the first 5 strings and the higher curve is for all 25 strings. The score for the first five strings is 0.187 for 50-byte run, and 0.549 for the 250-byte run. If we compare this to the results reported on TREC-8, we can see that the 50-byte result is between 13th and 14th among 17 50-byte runs with scores from 0.071 to 0.660; and the 250-byte result is between 1st and 2nd among 24 of the 250-byte runs with scores from 0.060 to 0.646.

The conclusion of this analysis is: The NLP part could improve the results obtained from the search engine by:

1. re-ranking the passages (0.18 gain), and
2. locating more precisely the candidate string ($0.543 = 0.73 - 0.187$ gain for 50-byte run, and $0.181 = 0.73 - 0.549$ gain for the 250-byte run).

4 QA using unification-based grammars

A description of unification-based grammars for NL can be found in Shieber 86 [12]. We rely on a more specific kind of unification-based grammar called *Head-driven Phrase Structure Grammar* (HPSG) (Sag and Wasow 99 [10]). The HPSG grammar formalism is described in Carpenter 92 [3]. We use the formalism as defined in Kešelj 99 [6], which also describes the Java parser Stefy [5] for HPSGs used in our experiments.

Let us give a short and informal introduction to unification-based grammars. The unification-based NL formalisms use a data structure called *attribute-value matrix* (AVM) or *typed feature structure.* An AVM can be regarded as the standard record data structure in programming languages. For example an AVM is:

$$\begin{bmatrix} \textit{employee} & \\ \textsf{ID:} & 001 \\ \textsf{NAME:} & \text{J. Q. Public} \\ \textsf{TEL:} & 123\text{-}4567 \\ \textsf{DEPT:} & \begin{bmatrix} \textit{dept} & \\ \textsf{ID:} & \text{SL} \\ \textsf{NAME:} & \text{Sales} \end{bmatrix} \end{bmatrix}.$$

One specific and important feature of AVMs is *re-entrancy,* or *structure sharing.* For example, in AVM

$$\begin{bmatrix} \textit{nlist} & \\ \textsf{HEAD:} & \boxed{1} \\ \textsf{TAIL:} & \begin{bmatrix} \textit{nlist} & \\ \textsf{HEAD:} & \boxed{1} \\ \textsf{TAIL:} & [\textit{list}] \end{bmatrix} \end{bmatrix}$$

the symbol $\boxed{1}$ denotes the elements that are the same, i.e., shared. Furthermore, we allow AVMs to be cyclic in a sense illustrated by the following AVM:

$$\boxed{1} \begin{bmatrix} \textit{employee} & \\ \textsf{ID:} & 001 \\ \textsf{NAME:} & \text{J. Q. Public} \\ \textsf{DEPT:} & \begin{bmatrix} \textit{dept} & \\ \textsf{ID:} & \text{SL} \\ \textsf{NAME:} & \text{Sales} \\ \textsf{MNGR:} & \boxed{1} \end{bmatrix} \end{bmatrix}.$$

This AVM represents an employee, who is a member of the "Sales" department, who is additionally the manager of the department.

The following AVM illustrates how the AVMs are used to describe NL syntax. It shows how we can enforce number agreement in the sentence 'He writes.' using AVMs and structure sharing:

$$\left[\begin{array}{ll} \textit{sentence} & \\ \mathsf{NP:} & \left[\begin{array}{ll} \textit{pron} & \\ \mathsf{ORTH:} & \text{He} \\ \mathsf{AGR:} & \boxed{1}\,\text{sg} \end{array}\right] \\ \mathsf{VP:} & \left[\begin{array}{ll} \textit{verb} & \\ \mathsf{ORTH:} & \text{writes.} \\ \mathsf{AGR:} & \boxed{1} \end{array}\right] \end{array}\right].$$

AVMs are also a useful building block for representing semantics. It is an important advantage of AVMs and unification-based grammars. Representing semantics in this way has the following advantages:

1. it improves parsing disambiguation since we can use semantics to prune some alternatives, and
2. we have a uniform representation of syntax and semantics, which makes the grammar easier to maintain.

Let us use the query "When did Beethoven die?," which appeared in TREC-8, to illustrate a unification-based grammar and a way in which semantics can be represented in this grammar. The following grammar of 3 rules accepts this query:

$$\left[\begin{array}{ll} Q & \\ \mathsf{SEM:} & \left[\begin{array}{ll} \mathsf{SELECT:} & \boxed{3} \\ \mathsf{WHERE:} & \boxed{2}\left[\begin{array}{ll} \mathsf{DATE:} & \boxed{3} \end{array}\right] \end{array}\right] \end{array}\right] \rightarrow \text{When did } \boxed{1}\left[\, NP \,\right] \left[\begin{array}{ll} V & \\ \mathsf{S:} & \boxed{1} \\ \mathsf{SEM:} & \boxed{2} \end{array}\right]$$

$$\left[\begin{array}{ll} V & \\ \mathsf{S:} & \boxed{1}\left[\, \textit{person} \,\right] \\ \mathsf{SEM:} & \left[\begin{array}{ll} \mathsf{EVENT:} & \text{die} \\ \mathsf{SUBJ:} & \boxed{1} \end{array}\right] \end{array}\right] \rightarrow \text{die}$$

$$\left[\begin{array}{ll} \textit{person} & \\ \mathsf{NAME:} & \text{Beethoven} \end{array}\right] \rightarrow \text{Beethoven}$$

The result of the query parsing is the following AVM:

$$\begin{bmatrix} Q & \\ \mathsf{SEM:} & \begin{bmatrix} \mathsf{SELECT:} & \boxed{1} \\ \mathsf{WHERE:} & \begin{bmatrix} \mathsf{EVENT:} & \text{die} \\ \mathsf{SUBJ:} & \begin{bmatrix} \textit{person} & \\ \mathsf{NAME:} & \text{Beethoven} \end{bmatrix} \\ \mathsf{DATE:} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The attributes names SELECT and WHERE are inspired by the SQL query language.

The relevant passage containing the answer contains the following excerpts:

> *If Beethoven felt... after finishing his monumental Ninth in early 1824... (he would die of jaundice and ascites on March 26, 1827), ...*

A part of the semantic representation of the above text is the following AVM:

$$\begin{bmatrix} \mathsf{EVENT:} & \text{die} \\ \mathsf{SUBJ:} & \text{Beethoven} \\ \mathsf{DATE:} & \text{March 26, 1827} \end{bmatrix}$$

which is easily matched to the query.

We will now describe how we use a unification-based grammar in QA. The parser Stefy [6] uses a chart parsing algorithm. Three charts are constructed in the process of parsing: a passive chart, an active chart, and a context chart. Figure 2 illustrates the use of those charts. We do not discuss basics of chart parsing here, but we will just briefly describe three charts used in our approach.[1] The passive chart is used to store finished edges. The active chart stores unfinished edges, which are analogous to the dotted rules in the Earley's parsing algorithm. The context chart stores objects in context in form of unfinished edges. The edges in the context chart are used for anaphoric resolution.

The result of parsing the question is an AVM that represents the question—its syntax and semantics. For each passage relevant to that question, we apply chart parsing on the whole passage. As a result, we obtain many edges within the passage representing semantics (and syntax) of various parts of passages (sub-sentence parts, sentences, and even some inter-sentence links). Finally, we locate the answer by finding the AVM in the passage chart that is best-matched to the question AVM. This is illustrated in figure 3.

Ideally, the matching procedure should be simple in the context of a unification-based grammar: We unify the question semantic matrix with the semantic matrix of the chart edge, and if the unification succeeds with an instantiation of the

[1] Chart parsing is a well-known subject. For details of the approach taken in Stefy please see [6].

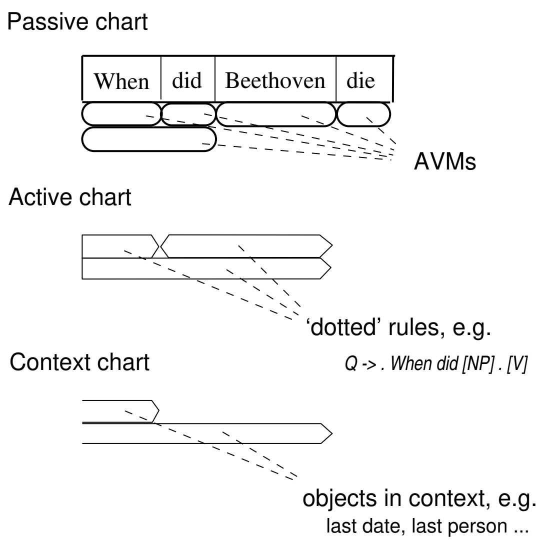

Fig. 2. Charts

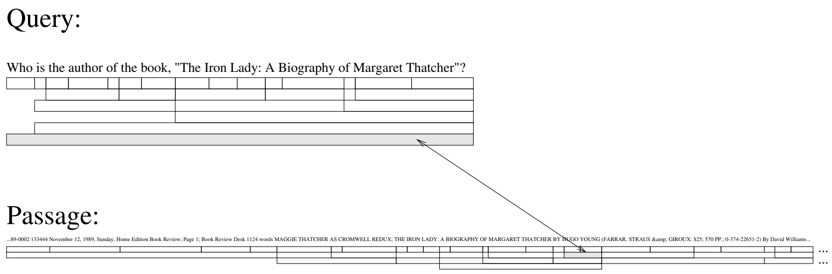

Fig. 3. Matching AVMs

'SELECT' variable, we have found our answer. However, this assumes perfect parsing of the query and the passage, which is not a realistic assumption. To solve the problem, we apply a graded unification procedure (similar to [7]). This increases the number of successful unifications. In order to reduce the errors, each unification operation returns a score, which is a measure of confidence of the operation.

We will now go over four examples from TREC-8, which were difficult for the classical IR method, and which can be handled with our approach.

Example 1. The question

When was London's Docklands Light Railway constructed?

is represented as the following AVM:

$$\left[\begin{array}{ll} \mathsf{SELECT:} & \boxed{1} \\ \mathsf{WHERE:} & \left[\begin{array}{ll} \mathsf{EVENT:} & \text{construct} \\ \mathsf{OBJ:} & \left[\begin{array}{l} \textit{name} \\ \mathsf{ORTH:}\ \text{London's\ldots Railway} \end{array}\right] \\ \mathsf{DATE:} & \boxed{1} \end{array}\right] \end{array}\right]$$

A string containing the answer is:

...the opening of the first 7 1/2 miles of the railway in 1987.

and it is represented in this way:

$$\left[\begin{array}{ll} \mathsf{HEAD:} & \text{opening} \\ \mathsf{OF:} & \left[\begin{array}{ll} \mathsf{HEAD:} & \text{miles} \\ \mathsf{OF:} & \text{the railway} \end{array}\right] \\ \mathsf{DATE:} & 1987 \end{array}\right]$$

These two AVMs are easily matched. We see how simple syntactic analysis improves performance.

Example 2. The question

How much could you rent a Volkswagen bug for in 1966?

is represented by

$$\left[\begin{array}{ll} \mathsf{SELECT:} & \boxed{1} \\ \mathsf{WHERE:} & \left[\begin{array}{ll} \mathsf{EVENT:} & \text{rent} \\ \mathsf{OBJ:} & \left[\mathsf{HEAD:}\ \text{bug}\right] \\ \mathsf{FOR:} & \boxed{1} \\ \mathsf{DATE:} & 1966 \end{array}\right] \end{array}\right]$$

A relevant string is:

...your could rent a Volkswagen bug for \$1 a day.

which produces the following AVM:

$$\begin{bmatrix} \textsf{HEAD:} & \text{rent} \\ \textsf{OBJ:} & \begin{bmatrix} \textsf{HEAD:} & \text{bug} \end{bmatrix} \\ \textsf{FOR:} & \text{\$1 a day} \end{bmatrix}$$

Again, simple syntactic processing with a unification matching procedure solves the problem.

Example 3. The question

In what year did Ireland elect its first woman president?

is represented by:

$$\begin{bmatrix} \textsf{SELECT:} & \boxed{1} \\ \textsf{WHERE:} & \begin{bmatrix} \textsf{EVENT:} & \text{elect} \\ \textsf{SUBJ:} & \begin{bmatrix} \textsf{HEAD:} & \text{Ireland} \end{bmatrix} \\ \textsf{OBJ:} & \begin{bmatrix} \textsf{HEAD:} & \text{president} \\ \textsf{MOD:} & \text{first woman} \end{bmatrix} \\ \textsf{DATE:} & \begin{bmatrix} \textsf{YEAR:} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The following excerpts from a passage are relevant to answering the question:

December 3, 1990, ...Ireland The nation's first female president, Mary Robinson, was inaugurated today...

Using the context chart, we resolve the reference 'today', unify it with the date given at the beginning of the article ('December 3, 1990'), and obtain the following representation:

$$\begin{bmatrix} \textsf{EVENT:} & \text{inaugurate} \\ \textsf{OBJ:} & \begin{bmatrix} \textsf{HEAD:} & \text{president} \\ \textsf{MOD:} & \text{first female} \end{bmatrix} \\ \textsf{DATE:} & \begin{bmatrix} \textsf{DAY:} & 3 \\ \textsf{MONTH:} & \text{December} \\ \textsf{YEAR:} & 1990 \end{bmatrix} \end{bmatrix}$$

This example illustrates how anaphoric resolution is important in solving the problem.

Example 4. The question

When did Beethoven die?

is represented as

$$\begin{bmatrix} \mathsf{SELECT:} & \boxed{1} \\ \mathsf{WHERE:} & \begin{bmatrix} \mathsf{EVENT:} & \text{die} \\ \mathsf{SUBJ:} & \text{beethoven} \\ \mathsf{DATE:} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

The relevant parts of the passage are:

> If Beethoven felt... after finishing his monumental Ninth in early 1824... (he would die of jaundice and ascites on March 26, 1827), ...

and the relevant AVM produced in the chart is:

$$\begin{bmatrix} \mathsf{EVENT:} & \text{die} \\ \mathsf{SUBJ:} & \text{Beethoven} \\ \mathsf{DATE:} & \begin{bmatrix} \mathsf{DAY:} & 26 \\ \mathsf{MONTH:} & \text{March} \\ \mathsf{YEAR:} & 1827 \end{bmatrix} \end{bmatrix}$$

In this example, the resolution of the pronoun 'he' is important in solving the problem.

5 Conclusion and Future Work

In this paper, we present a novel approach to QA, based on a unification-based grammar. Although some unification-based approaches to QA are successfully implemented, we do not know of any previous use of unification-based NL grammars. We argue that the semantic representation obtained from such grammars is robust and easy to maintain.

The preliminary evaluation results show that the NL approach can significantly improve performance of a classical IR system. We have described the algorithm, presented four case examples.

The evaluation is not done by now at a scale that would provide confident results, so it is a part of future work.

Acknowledgments

I wish to thank Dr. Nick Cercone for valuable discussions and comments regarding this work. I thank Dr. Charlie Clarke for providing me with the MultiText passages from TREC-8.

References

1. Askjeeves. WWW, 2000. `http://askjeeves.com/`.
2. R. Burke, Hammond K., V. Kulyukin, S. Lytinen, Tomuro N., and S. Schoenberg. Question answering from frequently asked question files. *AI Magazine*, 18(2):57–66, 1997.
3. Bob Carpenter. *The Logic of Typed Feature Structures with Applications to Unification-based Grammars, Logic Programming and Constraint Resolution*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, New York, 1992.
4. G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast automatic passage ranking (MultiText experiments for TREC-8). In *Proceedings of the Eight Text REtrieval Conference (TREC-8)*, 1999. `http://trec.nist.gov/pubs/trec8/t8_proceedings.html`.
5. Vlado Kešelj. Stefy. WWW, 2000. `http://www.cs.uwaterloo.ca/~vkeselj/stefy.html`.
6. Vlado Kešelj. Stefy: Java parser for HPSGs, version 0.1. Technical Report CS-99-26, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2000. `ftp://cs-archive.uwaterloo.ca/cs-archive/CS-99-26/`.
7. Albert Kim. Graded unification: A framework for interactive processing. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 313–315, June 1994.
8. Diego Mollá Alio, Jawad Berri, and Michael Hess. A real world implementation of answer extraction. In *Proc. of 9th International Conference and Workshop on Database and Expert Systems. Workshop "Natural Language and Information Systems" (NLIS'98)*, Vienna, 1998.
9. Dragomir R. Radev, John Prager, and Valerie Samn. Ranking suspected answers to natural language questions using predictive annotation. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP)*, Seattle, WA, May 2000.
10. Ivan A. Sag and Thomas Wasow. *Syntactic Theory: A Formal Introduction.* CSLI Publications, Stanford, 1999.
11. Rolf Schwitter, Diego Mollá, Rachel Fournier, and Michael Hess. Answer extraction: Towards better evaluations of NLP systems. In *Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems, ANLP-NAACL*, pages 20–27, Seattle, Washington, USA, May 2000.
12. Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Number 4 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford University, Stanford, CA, 1986.
13. Ellen M. Voorhees and Dawn M. Tice. The TREC-8 question answering track evaluation. In *Proceedings of the Eight Text REtrieval Conference (TREC-8)*, 1999. `http://trec.nist.gov/pubs/trec8/t8_proceedings.html`.