Towards Semantic-Based Overlap Measures for Question Answering

Diego Mollá
Centre for Language Technology
Macquarie University
Sydney, NSW 2109
Tel. +61 2 9850 9531
Fax +61 2 9850 9551
diego@ics.mq.edu.au

Abstract

In this paper we present an evaluation of overlap-based measures of similarity for sentences in the same language. The measures include syntactic and semantic information, and to that end they incorporate grammatical relations and flat logical forms. A full parser is required to build the above information. Separate extrinsic evaluations within the context of question answering have been made with two different parsers to test the impact of the parser and the overlap measures.

1 Introduction

Text-based Question Answering (QA) is a hot research topic and the increasing availability of electronic text will ensure that research in this area will continue for long. Much of the current research on QA focuses on the development of methodologies for processing relatively large volumes of text. For example, the competition-based QA track of the Text REtrieval Conference (TREC) (Voorhees, 2001) uses more than 3Gb of source text. Competing systems often exploit the data redundancy existing in the source text. Some of them even use the Web to increase the data redundancy (Brill et al., 2001; Clarke et al., 2001, for example). These systems typically trade accuracy for speed and avoid the use of intensive natural language processing techniques.

Most of the current QA systems are based on an architecture like that of Figure 1 (Hirschman and Gaizauskas, 2001; Voorhees, 2001). In an off-line or indexing stage, an **indexing** module analyses the text documents and creates a set of document images that will be used by the subsequent QA modules. In an on-line stage, a **question analysis** module classifies the question and determines the type of the expected answer. The question analysis module would typically return a list of named-entity types that are compatible with the question (for example a *who* question typically indicates people or organisations). The module may also produce an image of the question. This image may be similar in format to the document images and can range from a simple bag of words (Cooper and Rüger, 2000, for example) to a fairly complex logical form (Harabagiu et al., 2001; Elworthy, 2000, for example). Once the question is analysed, a **document preselection** module identifies the documents that are most likely to contain the answer. This module typically uses information retrieval techniques that rely on bag-of-words approaches and statistical information (Voorhees, 2001). A **filtering** module examines the resulting documents and selects or rewards the named entities that are compatible with the question type. A **scoring** module then performs a more intensive analysis and ranks the preselected named entities (an possibly surrounding text) according to their likelihood to contain the answer. The scoring system relies on the output given by the question analysis module and possibly the images of the preselected documents that were created during the off-line stage. There may be feedback loops between the document preselection, filtering, and scoring modules to increase the likelihood of find-

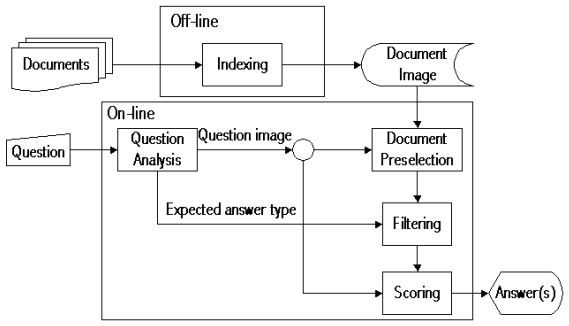

Figure 1: Architecture of a generic question-answering system.

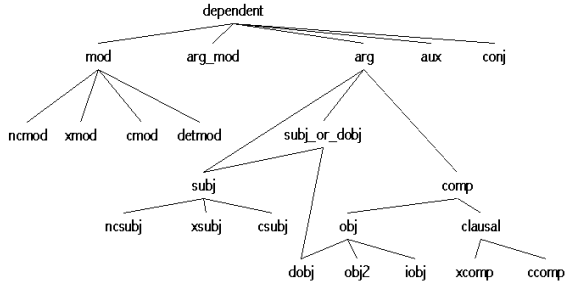

Figure 2: Hierarchy of grammatical relations.

ing difficult answers (Harabagiu et al., 2001, for example).

The scoring methodology used can be as simple as a word overlap or word frequency count, or as complex as an automatic proof system that operates on logical forms of both the questions and the answers. In some QA systems the scoring system relies heavily on the use of sentence patterns (Soubbotin, 2001, for example).

In the present study we have implemented a QA framework that uses simplifications of the modules described above. The emphasis in this study has been placed on the comparison of core methodologies for the scoring stage. To avoid the introduction of unwanted variables, we have avoided the use of methodologies that rely on world knowledge or domain knowledge. Thus, we have refrained from testing the use of external resources or inference systems.

With this QA framework we intend to assess the impact of syntactic and semantic information in a QA task. For that reason, we include information regarding word dependencies, grammatical relations, and logical forms in the procedure to measure the similarity between a question and an answer candidate. The results of the evaluations show both the impact of the scoring measures and the impact of the parsers used to extract the syntactic and semantic information. Section 2 describes grammatical relations and their use in an overlap measure. Section 3 focuses on the overlap of flat logical forms. Section 4 introduces the QA system that was used for the evaluations, and Section 5 explains the methodology used in the automatic evaluations. Finally, Section 6 discusses the results and Section 7 concludes and points to future lines of research.

2 Grammatical Relations

We use the grammatical relations of Carroll et al. (1998), who devised them as a means to provide the canonical representation of the output of parsers for their evaluation. Figure 2 shows the hierarchical classification of the grammatical relations (Briscoe and Carroll, 2000). This hierarchy allows the mapping from the output of an arbitrary parser and therefore allow the evaluation of parsers with different output granularity.

Table 1 lists the grammatical relations used in this paper and the evaluation – For further detail about grammatical relations see (Briscoe and Carroll, 2000). For example, the grammatical relations for the sentence *The man that came ate bananas and apples with a fork without asking* are:

```
DETMOD(_,man,the),
CMOD(that,man,come),
SUBJ(come,man,_),
SUBJ(eat,man,_),
DOBJ(eat,banana,_),
DOBJ(eat,apple,_)
CONJ(and,banana,apple),
NCMOD(fork,eat,with),
DETMOD(_,fork,a),
XCOMP(without,eat,ask)
```

Briscoe and Carroll's grammatical relations are not to be confused with the dependency arcs used in the theory of dependency grammar (Mel'čuk, 1988). To illustrate the difference, consider the sentence *The man that came ate bananas and*

Relation	*Description*
CONJ(type,head+)	Conjunction
MOD(type,head,dependent)	Modifier
CMOD(type,head,dependent)	Clausal modifier
NCMOD(type,head,dependent)	Non-clausal modifier
DETMOD(type,head,dependent)	Determiner
SUBJ(head,dependent,initial_gr)	Subject
OBJ(head,dependent,initial_gr)	Object
DOBJ(head,dependent,initial_gr)	Direct object
XCOMP(head,dependent)	Clausal complement without an overt subject

Table 1: Grammatical relations used in this paper.

apples with a fork. Figure 3 shows the graphical representation of the structure returned by Conexor FDG, a dependency-based parsing system (Tapanainen and Järvinen, 1997). In dependency grammar a unique head is assigned to each word, thus the head of *man* is *ate*. However *man* is the dependent of more than one grammatical relation, namely `SUBJ(eat,man,_)` and `SUBJ(come,man,_)`. Furthermore, in dependency grammar a word can have at most one dependent of each argument type, and so *ate* can have at most one object. But the same is not true for grammatical relations, and we get both `OBJ(eat,banana,_)` and `OBJ(eat,apple,_)`. Thus, grammatical relations provide a sentence representation that is closer to the semantic contents of a sentence than the representation provided by dependency arcs.

Mollá and Hutchinson (2003) used the grammatical relations to compare the accuracy of two broad-coverage dependency-based parsers, Link Grammar (Sleator and Temperley, 1993) and Conexor Functional Dependency Grammar (Tapanainen and Järvinen, 1997) — henceforth referred to as Conexor FDG. The evaluation used a subset of the original relations: `SUBJ`, `OBJ`, `XCOMP`, and `MOD`. This subset was used because of limitations of the output of the parsers and the algorithms for the automatic construction of the grammatical relations. Thus, the reduced grammatical relations for the example *The man that came ate bananas and apples with a fork without asking* is:

```
MOD(that,man,come),
SUBJ(eat,man,_),
SUBJ(come,man,_),
OBJ(eat,banana,_),
OBJ(eat,apple,_),
MOD(fork,eat,with),
XCOMP(without,eat,ask)
```

		Link Grammar	*Conexor FDG*
Precision	SUBJ	50.3%	73.6%
	OBJ	48.5%	84.8%
	XCOMP	62.2%	76.2%
	MOD	57.2%	63.7%
	Average	***54.6%***	***74.6%***
Recall	SUBJ	39.1%	64.5%
	OBJ	50%	53.4%
	XCOMP	32.1%	64.7%
	MOD	53.7%	56.2%
	Average	***43.7%***	***59.7%***

Table 2: Intrinsic evaluations of Link Grammar and Conexor FDG.

The results of the evaluation on a corpus annotated with the correct grammatical relations (Carroll et al., 1998) show significantly higher values of recall and precision for Conexor FDG with respect to Link Grammar (Table 2).

The grammatical relations can be used by the scoring module of our QA system. We only need to compute the overlap of grammatical relations between the question and the answer candidate. In theory, we must use the hierarchical organisation of the grammatical relations to decide if two grammatical relations unify. For example, `SUBJ(eat,man,_)` should unify with

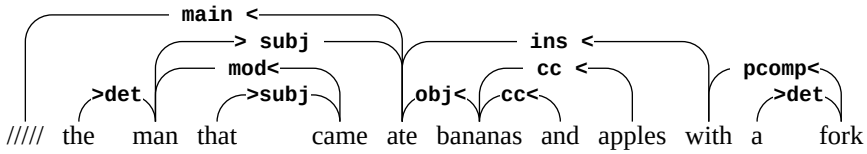

Figure 3: Dependency structure of a sample sentence.

SUB_OR_DOBJ(eat,man). However, since the same parser was used for both the question and the answer, the granularity of grammatical relations will be practically the same. Thus, each grammatical relation can be seen as an unstructured token and the scoring module can simply count the number of common tokens, very much like counting the overlap of words. This was the approach used in our QA prototype.

3 Flat Logical Forms

Flat logical forms have been used in several NLP systems including question-answering systems (Harabagiu et al., 2001; Lin, 2001; Mollá et al., 2000, for example). The flat logical forms that we use in our QA system are borrowed from (Mollá et al., 2000), who uses reification to flatten out nested expressions. For example, the flat logical form of *The cp command will quickly copy files* is:[1]

```
object('cp',o2,[x2]),
object('command',o3,[x3]),
compound_noun(x2,x3),
prop('quickly',p5,[e6]),
evt('copy',e6,[x3,x7]),
object('file',o7,[x7])
```

Flat logical forms express the main predicate-argument dependencies between the entities introduced by the sentence in a form that is suitable for computing semantic similarity. In particular, we only need to find the common predicates between the question and the answer candidate. The only additional complexity is the handling of variables in the terms of the question. It is therefore necessary to instantiate the question variables with constants found in the answer candidate. For example, the logical form of *Which command copies files?* is (the symbols in uppercase indicate variables):

```
object('command',O1,[X1]),
evt('copy',E2,[X1,X2]),
object('file',O2,[X2])
```

If this logical form is to match that of the sentence *The cp command will quickly copy files* above, the scoring module needs to instantiate the variable O1 in the question with the constant o3 in the answer candidate, X1 with x3, and so on. In our implementation we have used Prolog unification.

Since there are several plausible combinations of variable instantiations, the scoring module finds the set of instantiations that provides the highest overlap of logical forms.

Table 3 shows the flat logical forms of questions that differ solely in the argument positions, the flat logical form of an answer candidate, and the resulting overlaps.

4 The Question Answering Framework

In contrast with (Mollá et al., 2000), the semantic interpreters used in our evaluations to compute the logical forms do not use any additional lexical, domain, or world knowledge. Furthermore, there is no disambiguation step and there is no anaphora resolution module. The resulting semantic interpreters may therefore be less accurate, but the resulting QA systems are in a better position to be compared with the QA systems based on grammatical relations described in Section 2. Once it is decided which methodology is better, it is conceivable to add the additional modules that further enhance the expressivity of the sentence image.

For the present evaluation we used the Remedia Publications Reading Comprehension corpus used by DeepRead (Hirschman et al., 1999). The corpus is aimed at testing the degree of reading comprehension by children, and the documents

[1] For illustration purposes, the logical forms used in this paper are slightly different from the ones shown in the literature.

Answer candidate	**Flat Logical Form**
John saw Mary	object('john',o1,[x1]), object('mary',o3,[x3]), evt('see',e2,[x1,x3])
Question	**Flat Logical Form**
Did John see Mary?	**object('mary',O,[X]), evt('see',E,[Y,X]), object('john',O2,[Y])**
Did Mary see John?	object('john',O,[X]), **evt('see',E,[Y,X])**, object('mary',O2,[Y])

Table 3: Question answering using flat logical forms. Overlap shown in bold.

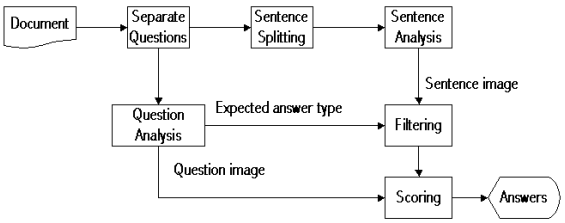

Figure 4: Architecture of the question-answering system.

Regex	*Expected Answer Type*
`^Who`	person, organization
`^What`	any
`^When`	date, time
`^Where`	location
`^Why`	any

Table 4: Question types and expected answer types.

in this corpus are classified into several levels of reading proficiency. There are about 30 documents for each of the levels 2, 3, 4, and 5, with a total of 117 documents. Each document includes a short piece of text and five questions (*who-*, *what-*, *when-*, *where-*, and *why-*) about the text.

The corpus contains annotations of the coreference chains and the named entities. The answers are also marked-up in the text and a gold standard for every answer is available. These annotations make the corpus suitable for the development and test of QA systems.

To evaluate the impact of the parsers and the scoring modules we have developed a simple question answering system framework (Figure 4). Given that every document contains the questions that are to be asked about the document, our QA system does not need to include a preselection stage. Instead, every document is processed independently. The system has a pre-processing stage that segments the document into text and questions. The text is split into sentences and each sentence is analysed independently.

Every question in the document is analysed to produce the question image. The question is also classified into one of the *who-*, *what-*, *when-*, *where-*, and *why-* categories. Depending on the question category, the question analysis module determines and returns the likely named entities of the expected answers.

Table 4 shows the named entities associated with each question type. The question classifier is extremely simple due to the fact that there are always five questions, and they have a very simple pattern. Thus, the regular expressions shown in Table 4 suffice to identify the question types. The procedure to determine the expected answer type is therefore very simple but fairly effective for the corpus and the named-entity types used in the named-entity annotations. Since the focus of this work is on the comparison of the scoring modules we did not feel we needed to produce a more sophisticated question analysis module.

The named-entity annotations provided with the evaluation corpus relieves the system from using a named-entity extraction component. The resulting system performs perfect named-entity extraction, which may artificially enhance the final quality of the answers returned. The results therefore cannot be compared with the results given by QA systems such as those participating in the Question Answering track of TREC (Voorhees, 2001), but they are good for the purposes of comparison that we pursue here.

The answer extraction module penalises all sentences that have no entities compatible with the answer type by giving them an initial score of -100. The only part of the QA system that varies across the comparisons is the scoring component.

The scoring methods described above can therefore be compared free of interference from other modules.

Following the specifications of the QA track of TREC8 to TREC10, the system returns the five answer candidates with highest scores, ranked by scores in descending order.

5 Evaluation Methodology

All four combinations of parser (Link Grammar and Conexor FDG) and scoring module (grammatical relations and flat logical forms) were used in the evaluations. All other modules in the QA system were left untouched.

The evaluation of the quality of the answers was done automatically. An answer candidate is judged correct if more than 80% of its words appear in one of the correct answers provided by the corpus annotations. We have not evaluated the accuracy of this evaluation method, but we have no reasons to believe that the final conclusions of the comparison are significantly affected by this automatic evaluation procedure, since all the experiments were evaluated with the same procedure.

The final measure is TREC QA's Mean Reciprocal Rank (MRR). Thus, for a given question, if the first correct answer returned is in rank R, the question is scored as $1/R$, or 0 if no correct answer is found. The final score of the system is the mean of scores for the individual questions.

An initial inspection of the first results revealed that the overall scores were very low. As a result, several answer candidates would receive the same score. Table 5 shows an example of the answers returned for a question. We can observe that all five answer candidates were given the same score, but only the last candidate was correct. There is no way for the system to know which of the sentences with same score is best. To determine the impact of returning several sentences with the same score, two indices were computed in addition to the MRR. The two indices correspond to the MRR that would result if the correct answer was chosen first or last among those of the same score. These indices represent the "best" and "worst" case, respectively, and ideally they would be almost equal.

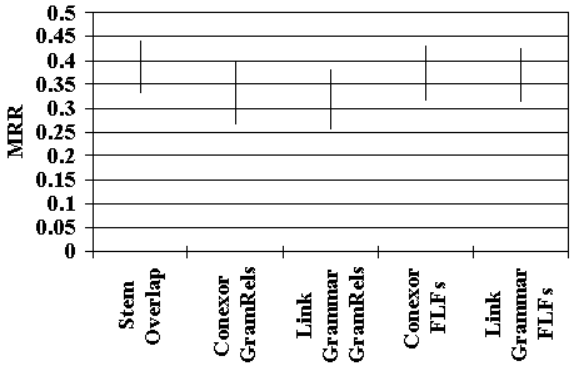

Figure 5: Evaluation of scoring measures.

6 Results and Discussion

The vertical bars in Figure 5 show the variation between the best and worst MRR for the text of levels 3, 4, and 5 together. Five cases are displayed, corresponding with a baseline scoring based on stem overlap and all combinations of parser (either Conexor FDG or Link Grammar) and scoring measure (either overlap of grammatical relations or overlap of flat logical forms).

Overall, we can see that the flat logical forms give better results than the grammatical relations. This is not surprising, since the flat logical forms were designed for tasks that require the semantic comparison of sentences. In contrast, the grammatical relations were designed for the comparison of parsers.

Also, Conexor FDG produces better results than Link Grammar. These results confirm Mollá and Hutchinson (2003)'s findings that Conexor FDG is slightly better than Link Grammar in a similar QA system that is based on a different corpus and evaluation methodology. Furthermore, the increase of performance with Conexor FDG is only marginal. This is in contrast with a direct comparison between Link Grammar and Conexor FDG which, as shown in Table 2 above, rated Conexor FDG much better than Link Grammar. The current experiment shows that, regardless of the method used (grammatical relations or flat logical forms), the MRR of the QA system that uses Conexor FDG is only slightly higher than the MRR of the QA system that uses Link Grammar. Thus, our results present further evidence that intrinsic evaluations

Rank	*Sentence*	*Score*	*Overlap*	*Correct*
1	1989 Remedia Publications, Comprehension s-4	1	`compound_noun(v_x2,v_x3)`	no
2	(North Redwood, Minn	1	`compound_noun(v_x2,v_x3)`	no
3	Not long ago, he ran an ad in some newspapers In small towns	1	`object(ad,v_o7,[v_x7])`	no
4	The ad showed a drawing of lovely furniture	1	`object(ad,v_o2,[v_x2])`	no
5	The ad said the furniture was for sale	1	`object(ad,v_o2,[v_x2])`	yes

Table 5: Answers returned for the question *What does the Sears ad offer?*

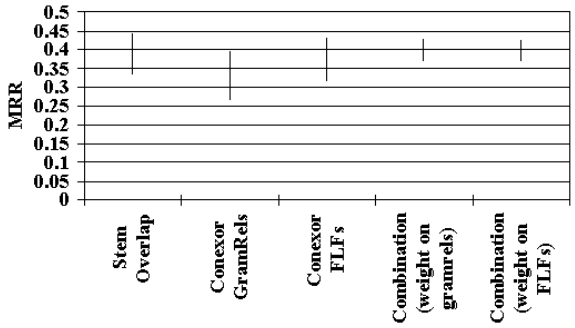

Figure 6: Combination of overlap measures.

are of very limited value, as stated already by Galliers and Sparck Jones (1993). An extrinsic evaluation that shows the impact of the modules to evaluate within the context of an application (QA in this case) may give results that differ substantially from those of an intrinsic evaluation.

What is surprising is the fact that, as Figure 5 shows, a measure based on simple stem overlap gives better results. Subsequent experiments indicate that a combination of the above scoring systems plus overlaps of word dependencies and word forms produce better results and they have a narrower difference between best and worst cases. For example, Figure 6 shows the results of two possible combinations:

Weight on grammatical relations:

$$Score = 27 \times GRO + 9 \times FO + 3 \times DO + O$$

Weight on flat logical forms:

$$Score = 27 \times FO + 9 \times GRO + 3 \times DO + O$$

In the above formulas, O stands for word form overlap, DO is the overlap of dependencies, FO is the overlap of minimal logical forms, and GRO is the overlap of grammatical relations.

The dependencies used were extracted from the Conexor FDG parser, which is dependency-based. All the experiments about the combination of scoring systems were used with Conexor FDG. Similar results are expected with Link Grammar or other parsers. We are experimenting with the impact of other possible scoring combinations.

7 Conclusions and Further Research

This research shows that the combined information on word dependencies, grammatical relations and flat logical forms improves the accuracy of the system with respect to the individual measures, though the additional resources required to extract syntactic and semantic information may not justify the use of these measures against simple word overlap.

Further research is necessary to determine the reason for this. For example, it may be that the very nature of the Reading Comprehension document set makes it unlikely to represent real-world text. The text is intentionally simple, the documents are short and there is little text redundancy. Furthermore, the fact that the texts are of varied topics and that every document contains the questions that apply to the document makes it highly unlikely that the questions associated to a specific document have eligible answer candidates outside the document. For all these reasons we are performing similar experiments with the corpus used in the QA track of TREC 10 and TREC 11. How-

ever, preliminary results support the results presented in this paper.

It may well be that one needs to compute more complex overlap measures to leverage the additional information. Additional further research includes the evaluation of weighted overlap measures that consider the relative importance of specific grammatical relations or logical form terms, and the determination of the optimal weights to be given.

Finally, perhaps the simple questions used in the Reading Comprehension corpus and in TREC do not require the use of much linguistic information. An evaluation framework with more complex questions is necessary to test this possibility.

We also plan to evaluate the impact of other NLP modules such as anaphora resolvers and disambiguators, and the use of lexical resources (e.g. WordNet or localisations of WordNet) and domain and world knowledge. The current QA system can be easily expanded to allow for all of these evaluations.

References

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Ted Briscoe and John Carroll. 2000. Grammatical relation annotation. On-line document. `http://www.cogs.susx.ac.uk/lab/nlp/carroll/grdescription/index.html`.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC98*.

C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2001. Web reinforced question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Richard J. Cooper and Stefan M. Rüger. 2000. A simple question answering system. In Voorhees and Harman (Voorhees and Harman, 2000).

David Elworthy. 2000. Question answering using a large NLP system. In Voorhees and Harman (Voorhees and Harman, 2000).

Julia R. Galliers and Karen Sparck Jones. 1993. Evaluating natural language processing systems. Technical Report TR-291, Computer Laboratory, University of Cambridge.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's question-answering server. In Voorhees and Harman (Voorhees and Harman, 2001).

Lynette Hirschman and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Read: A reading comprehension system. In *Proc. ACL'99*. University of Maryland.

Jimmy J. Lin. 2001. Indexing and retrieving natural language using ternary expressions. Master's thesis, MIT.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Diego Mollá and Ben Hutchinson. 2003. Intrinsic versus extrinsic evaluations of parsing systems. In *Proc. European Association for Computational Linguistics (EACL), workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, Budapest, April. Association for Computational Linguistics, ACL.

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *Traitment Automatique des Langues*, 41(2):495–522.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

M. M. Soubbotin. 2001. Patterns of potential answer expression as clues to the right answers. In Voorhees and Harman (Voorhees and Harman, 2001).

Pasi Tapanainen and Timo Järvinen. 1997. A nonprojective dependency parser. In *Procs. ANLP-97*. ACL.

Ellen M. Voorhees and Donna K. Harman, editors. 2000. *The Ninth Text REtrieval Conference (TREC-9)*, number 500-249 in NIST Special Publication. NIST.

Ellen M. Voorhees and Donna K. Harman, editors. 2001. *The Tenth Text REtrieval Conference (TREC-10)*, number 500-250 in NIST Special Publication. NIST.

Ellen M. Voorhees. 2001. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.