Corpus linguistics meets language technology: deep syntactic parsing for question answering

Nelleke Oostdijk
Department of Language and Speech
University of Nijmegen, The Netherlands
n.oostdijk@let.kun.nl

Abstract

To the extent that NLP is used by QA systems, it is mostly limited to tokenization, named entity recognition, stemming, POS tagging, and shallow parsing. More sophisticated NLP such as (deep) syntactic parsing is hardly ever used. In the present paper I investigate why this should be the case and try to establish how deep syntactic parsing as developed in the field of corpus linguistics might contribute to improve the overall performance of such systems.

Keywords: deep syntactic parsing, question answering, corpus linguistics

1. Introduction

The amount of information on the Internet is vast and continues to grow steadily. For users to successfully explore the Internet it is essential that they can gain effective access to the information. With the present search engines, it is possible – in response to a user's query – to retrieve a set of documents that are likely to contain relevant information. Although users have learned to make do with the present systems, they are quite some distance removed from what users actually want. Ideally, it should be possible to formulate questions or some other forms of requests for information in the same fashion as one does in human communication, and either retrieve a set of documents or obtain a direct answer, whichever is preferred.

The organizers of the Text REtrieval Conferences (TREC) in 1999 set up a Question Answering (QA) Track to encourage research in this field. Research groups were invited to compete on the following task: given a set of fact-based questions and a collection of newspaper/newswire texts, return a ranked list of [document, answer string] pairs. Answer strings were limited to 50 or 250 bytes and the document had to support the answer (cf. Voorhees 2000). Systems that participated all had more or less the same architecture (cf. de Boni 2001): they created a query from the user's question, used information retrieval to locate (segments of) documents likely to contain an answer, and then pinpointed the most likely answer passage within the candidate documents (cf. Hovy et al. 2001: 339).

Since the first QA track was organized (TREC-8), it has taken on the form of a yearly event which attracts more participants each year. Over the years the task has become more realistic. Whereas the early QA tracks included a single task, in TREC-10 three separate tasks were defined: the main task, the list task and the context task (cf. Voorhees 2002). The main task was essentially the same task as it was set previously, the difference being that it was no longer guaranteed that the text collection contained the answer to a given question. In last year's TREC, the difficulty of the task was further increased as questions now required an exact answer (a noun or a noun phrase) instead of a text snippet containing the answer. It is expected that future TREC QA tracks will increase the kinds and difficulty of the questions that systems are expected to handle. This is in line with the goal of the ARDA/AQUAINT programme which "intends to address a scenario in which multiple inter-related questions are asked in a focused topic area by a skilled, professional information analyst who is attempting to respond to larger, more complex information needs or requirements." (http://www.ic-arda.org/ InfoExploit/aquaint/)

Throughout the various TRECs the basic strategy of most systems has remained the same. However, as Voorhees (2002) observes in her overview of the TREC2001 QA track, "there was much less agreement on the best approaches to realizing that strategy. " She summarizes the approaches taken by the participants in TREC-10 as follows, identifying two areas in which they divert:

> Many groups continued to build systems that attempt a full understanding of the question, but increasingly many groups took a more shallow, data-driven approach. The data-driven approaches rely on simpler pattern matching methods using very large corpora (frequently the web) rather than sophisticated language processing. The idea

exploited in the massive data approach is that in a large enough data source a correct answer will usually be repeated often enough to distinguish it from the noise that happens to occasionally match simple patterns.

A second area in which there is no consensus as to the best approach is classification schemes for answer types. Some systems use a few very broad classes of answer types, while others use many specialized classes. The difference is a standard trade-off between coverage and accuracy. With many specialized answer types, finding the actual answer once an answer type is correctly classified is much easier because of the specificity of the class. However, deciding which class is correct, and ensuring there is a class for all questions, is more difficult with many specialized classes.

In the present paper I investigate the role of natural language processing (NLP) in QA systems and try to establish how deep syntactic parsing as developed in the field of corpus linguistics might contribute to improve the overall performance of such systems.

2. NLP in QA systems

Although an evaluation of the respective contributions made by NLP components to the relative performance of systems is impossible – in the published reports and papers generally no detailed information is given about how exactly systems work and in-depth analyses of the performance are lacking (cf. De Boni 2002) – an inventory of the types of NLP components used in the various systems participating in the TREC QA track brings to light that to the extent that NLP is used by most systems it is mostly limited to tokenization, named entity recognition, stemming, POS tagging, and shallow parsing. So far, more advanced NLP such as (deep) syntactic parsing is hardly ever used. Various explanations have been given why this should be the case. Apart from the argument put forward by Voorhees viz. “that in a large enough data source a correct answer will usually be repeated often enough to distinguish it from the noise that happens to occasionally match simple patterns”, other arguments have been raised against the use of syntactic parsing in the context of QA. Thus Jurafsky and Martin (2000: 573) observe that analyses obtained through syntactic parsing are “often not particularly well-suited for the task of compositional semantic analysis”. The structures yielded by conventional parsers suffer from the fact that (1) key semantic elements are often widely distributed across parse trees, thus complicating the composition of the meaning representation; (2) many syntactically motivated constituents play essentially no role in semantic processing; and (3) the general nature of many syntactic constituents results in semantic attachments that create nearly vacuous meaning representations (cf. Jurafsky and Martin 2000: 573-574). While in this interpretation the mismatch between syntactic structure and semantic analysis must essentially be attributed to the nature of syntactic analysis, it fails to recognize that this is not an inherent quality of syntax, but rather the consequence of an unfortunate choice of descriptive framework and possibly the inadequacy of the representation of the analysis. Another factor that has played a role in the limited use of syntactic parsing is the fact that the coverage of parsers generally has been insufficient and parsers on the whole have appeared to be not robust enough to deal with text in an unrestricted domain. This suggests that for syntax to play a more than trivial role, it is essential that the parser’s coverage and robustness are extended.

A review of various question answering systems that have participated in the TREC QA track shows that there are two obvious points where it is advantageous to have access to syntactic information, viz. in the analysis of the question and in pinpointing the answer in the ranked texts/segments. “Finding the expected answer type of a natural language question by relying only on the semantics of the question stem (i.e. *What, How*), and some bag-of-words approaches is not always possible, since stems are associated with many different types of answers and shallow syntax (e.g. phrasal parsers) fails to find the most discriminating concept in the question. The syntactic dependencies between the question phrases help solve this ambiguity: the answer type is indicated by the question phrase most connected to other concepts.” (Harabagiu et al. 2001). Syntactic parsing proves even more valuable as requests for information are not just expressed in the terms of *wh*-questions, but may take on a range of syntactic forms (incl. declarative and imperative sentences).

As for pinpointing the answer, a syntactic analysis of texts or segments likely to contain the answer would help overcome some of the limitations inherent in methods relying solely on word level information. So far many systems employ some form of window-based scoring method for determining which answer string is most salient. A 50 or 250 byte window is moved across the texts/segments holding the candidate answers. At each position a score is computed on the basis of the presence of desirable words. The window at the position giving the highest total score is returned as most likely to contain the expected answer (cf. Hovy et al. 2001). One of the drawbacks of this method is that – because of the sole reliance on information at the word level – it is not possible to recognize information of the desired type (such as Person or Location). Given the question *What does the Peugeot company manufacture?* the XEROX system (Hull 2000) on the basis of the sentence *Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands.* returned the following set of possible answers: {*Mr Longuet, decision, bodies, 504 utility vehicle, end, November, fate, Renault, hands*}. Had the system had access to syntactic information such as that the *wh*-element in the question refers to the direct object of *manufacture*, the return of the answer *bodies for its 504 utility vehicle* would have been straightforward. Another problem with the window-based scoring method is that it cannot pinpoint the answer boundaries precisely. This explains the awkward truncation of the answer strings returned. For example, a typical answer returned by one of the systems to the question *What is the name of the rare neurological disorder with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?* is as follows: *who said she has both Tourette's Syndrome and*. Syntactic parsing would be helpful in identifying *Tourette's Syndrome* as the exact answer.

In the light of what was said and argued above, there is no reason to believe that syntactic parsing cannot play a role in question answering. It can, not as a substitute for a lexical probabilistic analysis, but rather in aid of it. The relevant question then is: What information exactly should syntactic parsing provide? and, related to this question, What requirements does this impose on the parser? In the next sections I investigate to what extent a wide-coverage parser such as the current TOSCA parser that was originally developed in the field of corpus linguistics can be of use or what adaptations are necessary.

3. The TOSCA parser

The current TOSCA parser is the latest in a line of parsers that I developed previously for the linguistic annotation of corpora and which were used in a number of projects including the TOSCA (Tools for Syntactic Corpus Analysis; Oostdijk 1991) and the ICE (International Corpus of English) projects (Greenbaum 1992; Oostdijk 2000). The role of the parser in these projects was two-fold: (1) it served to test the linguistic hypotheses in the underlying formal grammar against the language data in the corpus; and (2) it was instrumental in annotating the corpus. The annotated corpus in its turn was used as input for research in English descriptive linguistics. In view of its role and the intended use of the annotated corpus, the parser was designed to yield for each input string minimally the correct parse in a given context. Obtaining the correct parse at the risk of over generating thus prevailed over minimizing the amount of ambiguity. The parser was one of the key components of the TOSCA analysis system (Aarts et al. 1998), where it would take as input the (manually corrected) result of the part-of-speech tagging. Through manual intervention the one correct parse was selected from possibly multiple parses.

The parser can best be characterized as a linguistically motivated, rule-based deep syntactic parser that has been developed for use in an unrestricted domain. The descriptive framework that is used in the underlying grammar is based on immediate constituency and the rank hierarchy, and in the analyses constituents are labelled for both the functional role and the categorical realization. See, for example, Figure 1 which shows the analysis that is obtained for the utterance *Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands.* [1]

The present implementation does not require (corrected) part-of-speech tagged input, but in principle can handle raw text input. The rate of ambiguity has been sharply reduced mostly through the incorporation of additional linguistic information that was acquired over the years; developments in the field of AGFL (Affix Grammars over Finite Lattices; Nederhof and Koster 1993) have increased overall efficiency. Moreover, the current implementation of the AGFL formalism makes it possible to include a number of heuristics that can be used for handling unknown lexical items and structures.

Application of this parser in the field of QA throws up questions such as 'What syntactic relations are relevant for retrieving information content?', 'To what extent is parsing accuracy required for the task?', 'To what extent should the parser be robust?, 'Can relations and/or attachments be left unspecified/underspecified?' and 'To what extent is a constituency analysis adequate for the present task? I will turn to each of these questions below.

[1] For an explanation of the abbreviations used, see the Appendix.

```
-:UTT(utt_cat_prop, prop_cat_s)
  PROP:S(s_type_declarative, complementation_motr, finiteness_past, mood_indic,
                                                                      voice_active)
   SU:NP(number_sing, person_third, nppo_cat_none)
     NPHD:N(n_type_proper, n_class_other, number_sing) "Mr Longuet"
   V:VP(cat_operator_no_cat_op, complementation_motr, finiteness_past, mood_indic,
                                           voice_active, number_sing, person_third)
     MVB:LV(complementation_motr, finiteness_past, mood_indic, number_sing,
                                                              person_third) "said"
   OD:CL(cl_type_zero_subordinate, complementation_motr, finiteness_past, mood_indic,
                                                              voice_active)
     SU:NP(number_sing, person_third, nppo_cat_none)
       DT:DTP
         DTCE:NP(NUMBER, person_third, nppo_cat_none)
           NPHD:N(n_type_proper, n_class_other, number_sing, case_gen) "Peugeot's"
       NPHD:N(n_type_common, n_class_other, number_sing) "decision"
       NPPO:CL(cl_type_zero, complementation_motr, finiteness_infin, mood_indic,
                                                                    voice_active)
         TO:PRTCL(prtcl_type_to) "to"
         V:VP(cat_operator_no_cat_op, complementation_motr, finiteness_infin,
                                        mood_indic, voice_active, NUMBER, PERSON)
           MVB:LV(complementation_motr, finiteness_infin, mood_indic, NUMBER, PERSON)
                                                                          "stop"
         OD:CL(cl_type_zero_subordinate, complementation_motr, finiteness_prespart,
                                                            mood_indic, voice_active)
           V:VP(cat_operator_no_cat_op, complementation_motr, finiteness_prespart,
                                            mood_indic, voice_active, NUMBER, PERSON)
             MVB:LV(complementation_motr, finiteness_prespart, mood_indic, NUMBER,
                                                              PERSON) "manufacturing"
           OD:NP(number_plu, person_third, nppo_cat_none)
             NPHD:N(n_type_common, n_class_other, number_plu) "bodies"
             NPPO:PP(prep_wrd_other)
               P:PREP(prep_wrd_other) "for"
               PC:NP(number_sing, person_third, nppo_cat_none)
                 DT:DTP
                   DTCE:PN(pn_type_possessive, function_pn_dt, NUMBER, person_third)
                                                                                "its"
                 NPHD:N(n_type_common, n_class_other, number_sing) "504 utility
                                                                             vehicle"
         A:PP(prep_wrd_from)
           P:PREP(prep_wrd_from) "from"
           PC:NP(number_sing, person_third, nppo_cat_ajp)
             DT:DTP
               DTCE:ART(NUMBER, definiteness_def) "the"
             NPHD:N(n_type_common, n_class_other, number_sing) "end"
             NPPO:PP(prep_wrd_of)
               P:PREP(prep_wrd_of) "of"
               PC:NP(number_sing, person_third, nppo_cat_none)
                 NPHD:N(n_type_proper, n_class_time, number_sing) "November"
     V:VP(cat_operator_no_cat_op, complementation_motr, finiteness_past, mood_indic,
                                             voice_active, number_sing, person_third)
       MVB:LV(complementation_motr, finiteness_past, mood_indic, number_sing,
                                                                 person_third) "left"
     OD:NP(number_sing, person_third, nppo_cat_pp)
       DT:DTP
         DTCE:ART(NUMBER, definiteness_def) "the"
       NPHD:N(n_type_common, n_class_other, number_sing) "fate"
       NPPO:PP(prep_wrd_of)
         P:PREP(prep_wrd_of) "of"
         PC:NP(number_sing, person_third, nppo_cat_none)
           DT:DTP
             DTCE:ART(NUMBER, definiteness_def) "the"
           NPHD:N(n_type_common, n_class_other, number_sing) "company"
     A:PP(prep_wrd_other)
       P:PREP(prep_wrd_other) "in"
       PC:NP(number_sing, person_third, nppo_cat_none)
         DT:DTP
           DTCE:NP(NUMBER, person_third, nppo_cat_none)
             NPHD:N(n_type_proper, n_class_other, number_sing, case_gen) "Renault's"
           NPHD:N(n_type_common, n_class_other, number_plu) "hands"
  PUNC:PM(punc_type_per) "."
```

Figure 1. Analysis for *Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands* .

4. Deep syntactic parsing for QA

One of the strong points in the syntactic analyses produced by the TOSCA parser is that a descriptive framework is used which is based on immediate constituency and the rank hierarchy. Where a shallow analysis essentially identifies simple phrases in the input and does nothing to structure the analysis result, the TOSCA type of analysis structures the information in such a fashion that it is immediately apparent what information belongs together and should be considered at any one time. In this respect the distinction between clauses and phrases and between phrases and words appears to be essential. Compare Figures 2 and 3. Figure 2 exemplifies part of a shallow analysis for the sentence *Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands.* In this type of analysis typically there is no embedding and no attachment of the prepositional phrases.

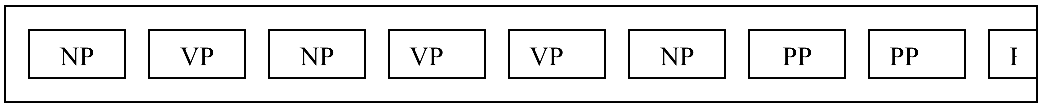

Figure 2. Shallow analysis showing flat/linear structure

(Part of) a deep syntactic analysis for the same sentence is exemplified in Figure 3 (cf. Figure 1). Although a mapping of the syntactic functions onto semantic roles is not entirely trivial, it is feasible to implement an algorithm by means of which this type of syntactic structure can be mapped onto a semantic representation. Syntactic function labels for constituents such as subject, object and adverbial in combination with additional information that is available regarding the type of clause (e.g. active declarative sentence with unmarked word order) and the type of verb (e.g. action verb, event verb) are useful in identifying the agent, recipient, location, etc.

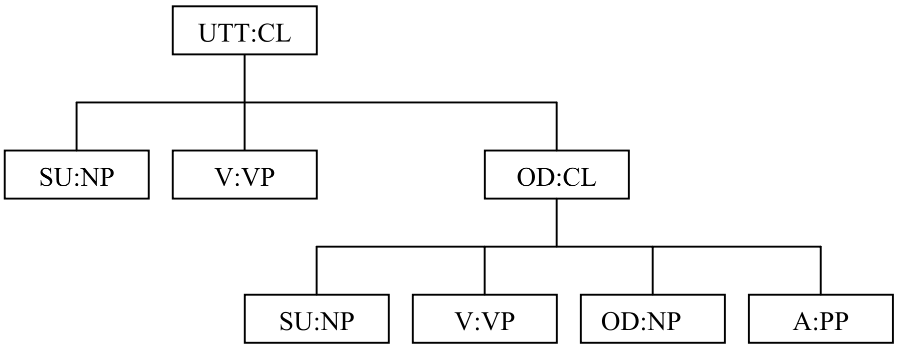

Figure 3. Hierarchical structure showing a (clausal) analysis based on immediate constituency and the rank hierarchy

While clausal functions play a crucial role in the mapping of the syntactic structure onto a semantic representation, phrasal functions such as head and modifier are useful since they help identify the key element in a phrase (cf. Figure 4).

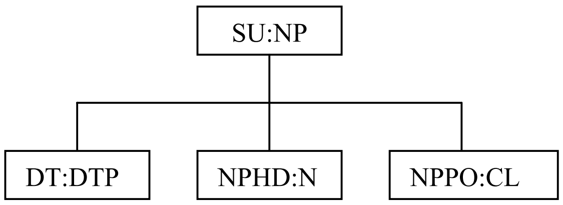

Figure 4. Phrasal structure

5. Adaptation of the TOSCA parser for use in QA

Having argued above that there is a role for syntactic analysis in QA and deep syntactic parsing can contribute to a better performance of QA systems, this section more specifically addresses the question What adaptations are necessary in order to make the TOSCA parser optimally suited for use in QA? The discussion focuses on the following points: coverage, accuracy, ambiguity, and robustness.

Coverage

Since the parser will be used to parse the questions and the ranked paragraphs/segments containing candidate answer strings, wide coverage is required. The TOSCA grammar already covers most syntactic constructs, including structures that are highly marked and/or relatively infrequent, so that in this respect few extentions are necessary. There are two areas, however, where adaptation of the parser in the light of the QA task is definitely required. Thus, additional rules are necessary for the adequate analysis of non-sentential constructions (incl. headlines and enumerations) that are characteristic of newspaper/newswire text. At present coverage is also poor in the area of named entities. The parser lexicon contains only a limited set of proper names and holds no information as to which items denote for example currencies and weights. In addition, the grammar does very little in the way of incorporating rules that describe the patterns displayed in names, dates, addresses, amounts and such like items.

Accuracy and ambiguity

Irrespective of whether the parser is used for the annotation of corpora with an eye to the advancement of linguistic description or the parser is used in an application such as a QA system, the accuracy should be as high as possible. While I agree with Hovy et al. (2001:341) that "parsing accuracy is particularly important for question parsing, because for only one question there may be several answers in a large document collection", I also think that parsing accuracy is quite important too for parsing the segments containing the candidate answer strings, especially when exact answers are required instead of text snippets containing the answer. In so far as the TOSCA parser at present yields ambiguous analyses for certain input strings, it is desirable to incorporate additional information in the parser lexicon from lexical resources such as WordNet (Fellbaum 1998) and FrameNet (http://www.icsi.berkeley.edu/~framenet). This includes semantic-syntactic information about collocations and subcategorization which will help to steer the parsing process and further reduce the degree of ambiguity. For those inputs where the parse result still remains ambiguous experiments are necessary to investigate the effect of using whatever parse the system generates first (instead of the one correct parse) and also the effect of collapsing the ambiguous parse trees into one while leaving the constituent boundaries and/or labels under specified.

Robustness

More than in the context of the linguistic annotation of corpora, a parser that is used with a QA system must be robust. When used for the annotation of corpora the performance of the parser is usually monitored by the linguist and failure to analyse a given input is taken as an indication that the input must be closely studied and possibly the lexicon and/or grammar adapted. For QA systems parse failures are unacceptable as they tend to block the (automatic) retrieval process.

6. Conclusion

In the present paper I have addressed the question as to whether and, if so, how deep syntactic parsing as developed for the linguistic annotation of corpora might contribute to improve the overall performance of question answering systems. I have argued that a wide-coverage deep linguistic parser such as the TOSCA parser *can* contribute, especially when it comes to analyzing the question and pinpointing the answer. Adaptations that are necessary so as to make the parser better suited for use in QA must be aimed at improving the recognition of named entities and non-sentential constructions, and increasing robustness. Further research is necessary to determine how best to strike a balance between granularity and ambiguity in the analyses.

Acknowledgement

Thanks are due to Lou Boves for his comments on an earlier version of this paper.

References

Aarts J, van Halteren H, Oostdijk N 1998 The linguistic annotation of corpora: The TOSCA Analysis System. *International Journal of Corpus Linguistics*, 3(2): 189-210.

ARDA/AQUAINT http://www.ic-arda.org/InfoExploit/aquaint/

Atwell E 1996 Comparative evaluation of grammatical annotation models. In Sutcliffe R, Koch H, McElligott A (eds), *Industrial parsing of software manuals*. Amsterdam, Rodopi, pp 13-23.

de Boni M 2001 Question answering system architecture. http://www-users.cs.york.ac.uk/~mdeboni/research/general/general_qa_system.html

de Boni M 2002 QA task Overview. http://www-users.cs.york.ac.uk/~mdeboni/research/general/trec2002.html

Greenbaum S 1992 A new corpus of English: ICE. In Svartvik J (ed.), *Directions in corpus linguistics. Proceedings of Nobel Symposium 82 Stockholm, 4-8 August 1991.* Berlin, New York, Mouton de Gruyter, pp 171-179.

Fellbaum C (ed.) 1998 *WordNet: An electronic lexical database*. Cambridge, Mass., MIT Press.

FrameNet. http://www.icsi.berkeley.edu/~framenet.

Harabagiu S, Moldovan D, Paşca M, Mihalcea R, Surdeanu M, Bunescu R, Gîrju R, Rus V, Morărescu P 2001 Falcon: boosting knowlegde for answer engines. http://trec.nist.gov/pubs/trec9/t9_proceedings.html

Hovy E, Gerber L, Hermjakob U, Lin C, Ravichandran D 2001 Towards semantics-based answer pinpointing. In *Proceedings of HLT 2001. First International Conference on Human Language Technology Research*. San Francisco, Morgan Kaufmann, pp 339-345.

Hull D 2000 XEROX TREC-8 Question Answering Track Report. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

Jurafsky D, Martin J (eds.) 2000 *Speech and language processing. An introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NJ, Prentice Hall, Pearson.

Lin D 1996 Dependency-based parser evaluation. A study with a software manual corpus. In Sutcliffe R, Koch H, McElligott A (eds.), *Industrial parsing of software manuals*. Amsterdam, Rodopi, pp 25-45.

Nederhof M, Koster C 1993 A customized grammar workbench. In Aarts J, de Haan P, Oostdijk N (eds.), *English language corpora: design, analysis and exploitation*. Amsterdam, Rodopi, pp 145-161.

Oostdijk N 1991 *Corpus linguistics and the automatic analysis of English*. Amsterdam: Rodopi.

Oostdijk N 1996 Using the TOSCA analysis system to analyse a software manual corpus. In Sutcliffe R, Koch H, McElligott A (eds.), *Industrial parsing of software manuals*. Amsterdam, Rodopi, pp 179-206.

Oostdijk N 2000 Corpus-based English linguistics at a crossroads. *English Studies* 81(2): 127-141.

TREC – Text REtrieval Conferences. http://trec.nist.gov/

Voorhees E 2000 The TREC-8 Question Answering Track Report. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

Voorhees E 2002 The TREC-10 Question Answering Track Report. http://trec.nist.gov/pubs/trec10/t10_proceedings.html

Appendix

In the example analyses abbreviations were used to denote function and categories. They are explained below.

Function labels:		*Category labels:*	
A	adverbial	ART	article
DT	determiner	CL	clause
DTCE	central determiner	DTP	determiner phrase
MVB	main verb	LV	lexical verb
NPHD	head of noun phrase	N	noun
NPPO	noun phrase postmodifier	NP	noun phrase
OD	direct object	PM	punctuation mark
P	preposition	PN	pronoun
PC	prepositional complement	PP	prepositional phrase
PROP	proposition	PREP	preposition

PUNC	punctuation	PRTCL	particle
SU	subject	S	sentence
TO	*to* infinitive marker	UTT	utterance
V	verb	VP	verb phrase