Answer Extraction in Technical Domains

Fabio Rinaldi[1], Michael Hess[1], Diego Mollá[2], Rolf Schwitter[2], James Dowdall[1], Gerold Schneider[1], and Rachel Fournier[1]

[1] University of Zurich, Department of Computer Science
Winterthurerstrasse 190, CH-8057 Zurich, Switzerland
{rinaldi,hess,dowdall,gschneid,fournier}@ifi.unizh.ch
[2] Department of Computing, Macquarie University
Sydney, Australia
{diego,rolfs}@ics.mq.edu.au

Abstract. In recent years, the information overload caused by the new media has made the shortcomings of traditional Information Retrieval increasingly evident. Practical needs of industry, government organizations and individual users alike push the research community towards systems that can exactly pinpoint those parts of documents that contain the information requested, rather than return a set of relevant documents. Answer Extraction (AE) systems aim to satisfy this need. In this article we discuss the problems faced in AE and present one such system.

1 Introduction

Traditional Information Retrieval (IR) techniques provide a very useful solution to a classical type of *information need*, which can be described with the scenario of "Essay Writing". The user needs to find some information and backup material on a particular topic, and she will sift through a number of documents returned by the IR system. This assumes that the user has sufficient time to elaborate and extract the relevant information from a number of documents.[1] However, a different type of information need is becoming increasingly more common, namely one where the user has to solve a specific problem in a technical domain, which requires finding precise information of a limited size. This could be called a "Problem Solving" scenario. A very fitting example is that of technical manuals. Imagine the situation of an airplane maintenance technician who needs to operate on a defective component which is preventing an airplane from starting. He needs to swiftly locate in the maintenance manual the specific procedure to replace that component. What users really need in this situation are systems capable of analyzing a question (phrased in Natural Language) and searching for a precise answer in document collections.

In this paper we discuss the general problem of Question Answering (QA) and focus on a simpler task, Answer Extraction (AE), as a building block towards

[1] It has been often observed that traditional Information Retrieval should rather be called "Document Retrieval".

the more ambitious goal of QA. We also briefly describe an AE system that is used to solve a real world problem. In section 2 we compare QA, as defined in the TREC competitions [26, 28], with AE. Section 3 presents the ExtrAns system whereas section 4 evaluates it. In section 5 we survey the related work.

2 Question Answering and Answer Extraction

There are different levels of performance that can be expected from a Question Answering system, and a classification is not easy. However, a first broad distinction can be made on the basis of the type of knowledge that the system employs, which ultimately determines which questions the system can answer.

An ideal system would return a grammatically well-formed surface string generated from a non-linguistic knowledge base in response to a natural language query. Unfortunately, many problems in the Knowledge Representation field are still to be solved and a comprehensive repository of world knowledge is not available.[2] What is achievable are systems that acquire their knowledge only from the target data (the documents to be queried). Such a system may allow inferences at the local/linguistic level or across multiple or single texts, depending on the task at hand. Systems employing only knowledge found explicitly in the documents should be called, in our opinion, "Answer Extraction Systems" whereas the term "Question Answering" should be reserved for systems making use of wider inferential capabilities.

The complexity of an Answer Extraction system could be defined in terms of the kind of transformations that it allows over the user query. The most simple approach would be to allow only syntactic variants (such as active/passive), while more sophisticated approaches would gradually include detection of synonyms and of more complex lexical relations among words such as thesaurus relationships like *"subdirectory* ***is a subtype of*** *directory"* as well as textual references (pronouns, definite noun phrases), and finally the use of meaning postulates (such as *"if something is* ***installed*** *in some place, then it* ***is*** *there"*).

The focus of the TREC competitions has been predominantly factual (non-generic, extensional) questions about events, geography and history, such as *"When was Yemen reunified?"* or *"Who is the president of Ghana?"*. It has been observed repeatedly that many such questions would better be directed at encyclopedias rather than at newspaper articles. Questions concerning rule-like or definitional knowledge (generic, intensional questions), such as *"How do you stop a Diesel engine?"* or *"What is a typhoon?"* have received less attention.[3] As technical documents consist almost exclusively of generic statements it is this type of question on which we have focused our attention.

[2] Despite some commendable efforts in this direction [17].

[3] Although a small number of them were included in the QA track of TREC-9 and TREC-10.

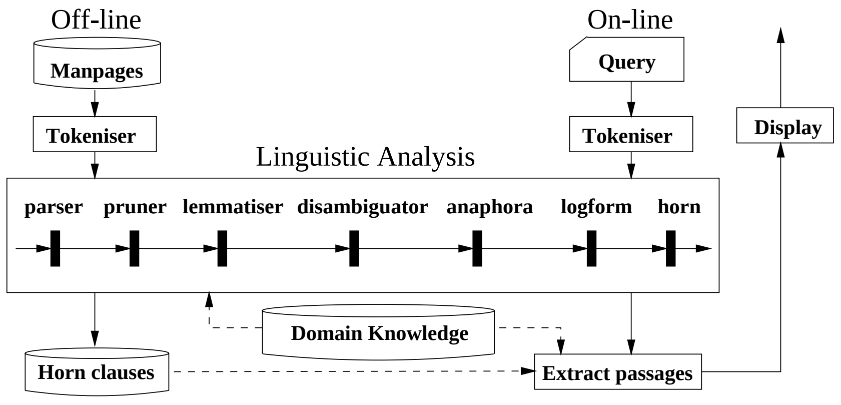


Fig. 1. Architecture of the ExtrAns system

3 A brief Presentation of ExtrAns

Over the past few years our research group has developed an Answer Extraction system (ExtrAns) that works by transforming documents and queries into a semantic representation called Minimal Logical Form (MLF) [21] and derives the answers by logical proof from the documents. A full linguistic (syntactic and semantic) analysis, complete with lexical alternations (synonyms and hyponyms) is performed. While documents are processed in an off-line stage, the query is processed on-line (see Fig. 1).

Two real world applications have so far been implemented with the same underlying technology. The original ExtrAns system is used to extract answers to arbitrary user queries over the Unix documentation files ("man pages"). A set of 500+ unedited man pages has been used for this application. An on-line demo of ExtrAns can be found at the project web page.[4]

More recently we tackled a different domain, the Airplane Maintenance Manuals (AMM) of the Airbus A320, which offered the additional challenges of an SGML-based format and a much larger size (120MB).[5] Despite being developed initially for a specific domain, ExtrAns has demonstrated a high level of domain independence.

As we work on relatively small volumes of data we can afford to process (in an off-line stage) all the documents in our collection rather than just a few selected paragraphs. Clearly in some situations (e.g. processing incoming news) such an approach might not be feasible and paragraph indexing techniques would need to be used. At the moment we have a preselection mechanism which is based on a loose matching of question concepts against the stored semantic representations of the documents. Our current approach is particularly targeted to small and

[4] `http://www.ifi.unizh.ch/cl/extrans/`

[5] Still considerably smaller than the size of the document collections used for TREC.

medium sized collections. For larger collections an initial preselection module would be unavoidable.

In the present section we will briefly describe the ExtrAns system and provide examples from the two applications. Further details can be found in [20–22].

3.1 Lexical and Syntactic Analysis

The document sentences (and user queries) are syntactically processed with the Link Grammar (LG) parser [25] which uses a dependency-based grammar. A corpus-based approach [3] is used to deal with ambiguities that cannot be solved with syntactic information only, in particular attachments of prepositional phrases, gerunds and infinitive constructions.

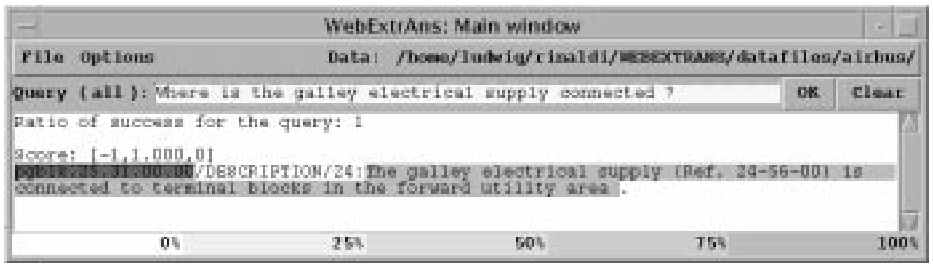
ExtrAns adopts an anaphora resolution algorithm [16] that was originally applied to the syntactic structures generated by McCord's Slot Grammar [18]. So far the resolution is restricted to sentence-internal pronouns but the same algorithm can be applied to sentence-external pronouns too.

A small lexicon of nominalizations is used for the most important cases. The main problem here is that the semantic relationship between the base words (mostly, but not exclusively, verbs) and the derived words (mostly, but not exclusively, nouns) is not sufficiently systematic to allow a derivation lexicon to be compiled automatically. Only in relatively rare cases is the relationship as simple as with "to edit <a text>" $\leftrightarrow$ "editor of <a text>"/"<text> editor", as the effort that went into building resources such as NOMLEX [19] also shows.

Recently, we have integrated a new module which is capable of identifying previously detected multi-word domain-specific terminology (stored in a separate external DB) and processing them as single syntactical units. One of the positive effects is that the complexity of parsing the manual is considerably reduced (in some instances by as much as 50%).

User queries are processed on-line and converted into MLFs (possibly expanded by synonyms) and proved by refutation over the document knowledge base. Pointers to the original text attached to the retrieved logical forms allow the system to identify and highlight those words in the retrieved sentence that contribute most to that particular answer [22]. An example of the output of ExtrAns can be seen in Fig. 2. When the user clicks on one of the answers provided, the corresponding document will be displayed with the relevant passages highlighted.

When no direct proof for the user query is found, the system is capable of relaxing the proof criteria in a stepwise manner. First, hyponyms of the query terms will be added, thus making it more general but still logically correct. If that fails, the system will attempt approximate matching, in which the sentence with the highest overlap of predicates with the query is retrieved. The (partially) matching sentences are scored and the best fits are returned. In the case that even this method does not find sufficient answers the system will attempt keyword matching, in which syntactic criteria are abandoned and only information about word classes is used. This last step corresponds approximately to a traditional passage-retrieval methodology with consideration of the POS tags.



```
Data: /home/ludwig/rinaldi/WEBEXTRANS/datafiles/airbus/
Query (all ): Where is the galley electrical supply connected ?
Ratio of success for the query: 1

Score: [-1,1,000,0]
[illegible]/DESCRIPTION/24:The galley electrical supply (Ref. 24-56-00) is
connected to terminal blocks in the forward utility area .
```

Fig. 2. An example of the output of ExtrAns

It is important to note that, in the strict mode, the system finds only logically correct proofs (within the limits of what MLFs can represent; see below), i.e. it is a high precision AE system.

3.2 Semantic Analysis

The meaning of the documents and of the queries produced by ExtrAns is expressed by means of Minimal Logical Forms (MLFs) [24]. The MLFs are designed so that they can be found for *any* sentence (using robust approaches to treat very complex or ungrammatical sentences), and they are optimized for NLP tasks that involve the semantic comparison of sentences, such as AE.

The main feature of the MLFs is the use of reification to achieve flat expressions. As opposed to Hobb's ontologically promiscuous semantics [12], where every predicate is reified, for the time being we apply reification to a very limited number of types of predicates, in particular to objects, eventualities (events or states), and properties.[6] That way we can represent event modifiers, negations, higher order verbs, conditionals, and higher order predicates.

The expressivity of the MLFs is minimal in the sense that the main syntactic dependencies between the words are used to express verb-argument relations, and modifier and adjunct relations. However, complex quantification, tense and aspect, temporal relations, plurality, and modality are not expressed. One of the effects of this kind of underspecification is that several natural language queries, although slightly different in meaning, produce the same logical form.

The MLFs are expressed as conjunctions of predicates with all the variables existentially bound with wide scope. For example, the MLF of the sentence *"cp will quickly copy the files"* is:

(1) holds(e4), object(cp,o1,x1), object(s_command,o2,x1), evt(s_copy,e4,[x1,x6]), object(s_file,o3,x6), prop(quickly,p3,e4).

In other words, there is an entity $x1$ which represents an object of type *cp* and of type *command*, there is an entity $x6$ (a file), there is an entity $e4$,

[6] Another related approach is that taken in Minimal Recursion Semantics [6].

which represents a copying event where the first argument is $x1$ and the second argument is $x6$, there is an entity $p3$ which states that $e4$ is done quickly, and the event $e4$, that is, the copying, holds. The entities $o1$, $o2$, $o3$, $e4$, and $p3$ are the result of reification. The reification of the event, $e4$, has been used to express that the event is done quickly. The other entities are not used in this MLF, but other more complex sentences may need to refer to the reification of objects (non-intersective adjectives) or properties (adjective-modifying adverbs).

ExtrAns' domain knowledge determines that *cp* is a command name, and the words defined in the thesaurus will be replaced with their synset code (here represented as s_command, s_copy, and s_file). We have developed a small domain-specific thesaurus based on the same format as WordNet [7].

The MLFs are derived from the syntactic information produced by Link Grammar (LG) [25]. The methodology to produce the MLFs is relatively simple, one only needs to follow the main dependencies produced by the LG. However, as has been said elsewhere [21], the internal complexities of the dependency structures produced by the LG must be taken into account when producing the MLFs. The LG has a robust component that makes it possible to return structures even if the sentences are too complex or ungrammatical. The resulting structures can still be processed by ExtrAns and the corresponding MLFs are produced, possibly extended with special predicates that mark the unprocessed words as "keywords".

ExtrAns finds the answers to the questions by forming the MLFs of the questions and then running Prolog's default resolution mechanism to find those MLFs that can prove the question. Thus, the logical form of the question *"which command can duplicate files?"* is:

(2) object(s_command,O1,X1), evt(s_copy,E1,[X1,X2]), object(s_file,O2,X2)

The variables introduced in a question MLF are converted into Prolog variables. The resulting MLF can be run as a Prolog query that will succeed provided that the MLF of the sentence *"cp will quickly copy the files"* has been asserted. A sentence identifier and a pointer (indicating the tokens from which the predicate has been derived) are attached to each predicate of a MLF in the knowledge base. This information matches against additional variables attached to the predicates in the question (not shown in the example above) and is eventually used to highlight the answer in the context of the document (see Fig. 2). The use of Prolog resolution will find the answers that can logically prove the question, but given that the MLFs are simplified logical forms converted into flat structures, ExtrAns will find sentences that, logically speaking, are not exact answers but are still relevant to the user's question, such as: *"cp copies files"*, *"cp does not copy a file onto itself"*, *"if the user types y, then cp copies files"*.

In our view MLFs open up a potential path to a stepwise development of a question answering system by allowing monotonically incremental refinements of the representation without the need to destruct previous partial information [24]. While MLFs specify the core meaning of sentences they leave underspecified those aspects of semantics that are less relevant or too hard to analyse, for the time being.

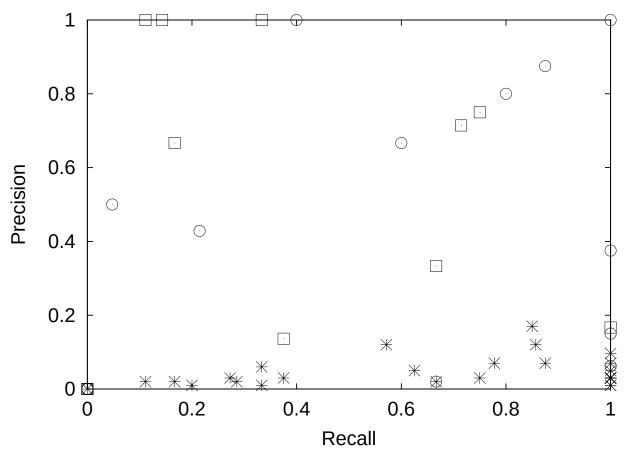


Fig. 3. Recall against precision for 30 queries and the top 100 hits per query. Prise's results are displayed with a star ($*$), and ExtrAns' results with circles ($\odot$) for the default search and with squares ($\square$) for the approximate matching.

4 Evaluation

We conducted two different kinds of evaluation, one designed to compare the original ExtrAns system against a standard IR system, and one designed to give us a feeling for the portability of ExtrAns to a new domain. For the initial evaluation we used a set of 30 queries over 500 manual pages. The system chosen for the comparison was Prise, a system developed by NIST [10]. Since Prise returns full documents, we used ExtrAns' tokenizer to find the sentence boundaries and to create independent documents, one per sentence in the manual pages. Then Prise was run with our set of queries, which lead to an average of 908 hits per query. The set of all correct answers was compiled mainly by hand. As Prise provides a ranked output, in order to compute precision and recall one has to select a cut-off value (n). The combined plot of pairs computed for each n did not show significant differences with the plot for $n = 100$: the values for ExtrAns were nearly the same, and for Prise, the number of recall and precision pairs increased but the area with the highest density of points remains the same. We will therefore concentrate on the plot for $n = 100$.

Fig. 3 shows that precision is in general higher for ExtrAns than for Prise, and that Prise has better recall values. In the upper right corner, we can see a higher density of ExtrAns' values which is likely to shift to the left if we use a less restricted set of queries. The fact that ExtrAns never stopped at the hyponym and keyword search is also related to the actual query set. If the queries were more complex, we would have some recall and precision pairs corresponding to the keyword search, and this would probably cause a lower overall precision.

When we started to work on a new domain we designed a simple evaluation framework, to test the domain independence of the system. Our porting to the domain of Airbus Maintenance manuals did not involve modification of any linguistic component, but simply of the I/O interfaces of the system and the development of a new tokenizer capable to deal with SGML/XML markup. We selected semi-randomly 100 sentences from the data documents and prepared 100 questions to which those sentences could be an answer. When the port was complete we tested the questions and we obtained the expected answer on 84% of them, in another 9% of cases we obtained a correct answer, different from the one we expected, in 7% of cases we did not obtain a correct answer. We are now in the process of analyzing these results. We are also planning a similar type of evaluation with questions directly formulated by the potential users of the system (which might not have a straightforward answer in the manual).

5 Related Work

IR techniques can be used to implement QA/AE systems, by applying them at the passage or sentence level. Portions of text with the maximum overlap of question terms contain, with a certain probability, an answer. The relevance of the passages is almost invariably determined on the basis of the weights assigned to individual terms, and these weights are computed from term frequencies in the documents (or passages) and in the entire document collection (the *tf/idf* measure). Since this measure is blind to syntactic (and hence semantic) relationships it does not distinguish between hits that are logically correct and others that are purely coincidental. "Bag of words" approaches will never be able to distinguish different strings that contain the same words in different syntactic configurations, such as "absence of evidence" and "evidence of absence".

Results from the two first TREC Question Answering Tracks [26, 28] showed clearly that traditional IR techniques are not sufficient for satisfactory Answer Extraction. When the answer is restricted to a very small window of text (50 bytes), systems that relied only on those techniques fared significantly worse than systems that employed some kind of language processing.

More successful approaches employ special treatment for some terms [8] (e.g. named entity recognition [14]) or a taxonomy of questions [13]. There appears to be some convergence towards a common architecture which is based on four core components [1, 23]. Passage Retrieval [4] is used to identify paragraphs (or text windows) that show similarity to the question (according to some system specific metric), a Question Classification module is used to detect possible answer types [11], an Entity Extraction module [15] analyzes the passages and extracts all the entities that are potential answers and finally a Scoring module [2] ranks these entities against the question type, thus leading to the selection of the answer(s).

The systems that obtained the best results in the QA track of TREC have gradually moved into NLP techniques, such as semantics and logical forms. Falcon [9] (the best performing system in TREC-9) performs a complete analysis of a set of preselected paragraphs for each query and of the query itself and creates, after several intermediate steps, a logical representation inspired by the notation proposed by Hobbs [12]. Another similarity between ExtrAns and Falcon is that both build a semantic form starting from a dependency-based representation of the questions, although the syntactic analysis in Falcon is based on a statistical parser [5] while we use a dependency parser. As for the type of inferencing used, while ExtrAns uses standard deduction (proving questions over documents), Falcon uses an abductive backchaining mechanism to exclude erroneous answers.

6 Conclusion

In this article we have proposed as a first step towards Question Answering a more restricted kind of task for which we suggest the use of the term "Answer Extraction". We have described the differences between QA and AE and have presented an example of an AE system.

If Answer Extraction is to perform satisfactorily in technical domains over limited amounts of textual data with very little redundancy it must make maximal use of the information contained in the documents. This means that the meaning of both queries and documents must be taken into account, by syntactic and semantic analysis. Our fully functioning AE system, ExtrAns, shows that such applications are within the reach of present-day technology.

References

1. Steven Abney, Michael Collins, and Amit Singhal. Answer extraction. In Sergei Nirenburg, editor, *Proc. 6th Applied Natural Language Processing Conference*, pages 296–301, Seattle, WA, 2000. Morgan Kaufmann.
2. Eric Breck, John Burger, Lisa Ferro, Warren Greiff, Marc Light, Inderjeet Mani, and Jason Rennie. Another sys called qanda. In Voorhees and Harman [28].
3. Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. COLING '94*, volume 2, pages 998–1004, Kyoto, Japan, 1994.
4. C.L.A. Clarke, G.V. Cormack, D.I.E. Kisman, and T.R. Lynam. Question answering by passage selection (MultiText experiments for TREC-9). In Voorhees and Harman [28].
5. Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34st Annual Meeting of the Association for Computational Linguistics, ACL-96*, pages 184–191, 1996.
6. Ann Copestake, Dan Flickinger, and Ivan A. Sag. Minimal recursion semantics: an introduction. Technical report, CSLI, Stanford University, Stanford, CA, 1997.
7. Christiane Fellbaum. Wordnet: Introduction. In Christiane Fellbaum, editor, *WordNet: an electronic lexical database*, Language, Speech, and Communication, pages 1–19. MIT Press, Cambrige, MA, 1998.
8. Olivier Ferrett, Brigitte Grau, Martine Hurault-Plantet, and Gabriel Illouz. Qualc - the question-answering system of limsi-cnrs. In Voorhees and Harman [28].
9. Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morarescu. FALCON: Boosting knowledge for answer engines. In Voorhees and Harman [28].

10. Donna K. Harman and Gerald T. Candela. A very fast prototype retrieval using statistical ranking. *SIGIR Forum*, 23(3/4):100–110, 1989.
11. Ulf Hermjakob. Parsing and question classification for question answering. In *Proc. of the ACL'01 workshop "Open-Domain Question Answering"*, pages 17–22, 2001.
12. Jerry R. Hobbs. Ontological promiscuity. In *Proc. ACL'85*, pages 61–69. University of Chicago, Association for Computational Linguistics, 1985.
13. Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in webclopedia. In Voorhees and Harman [28].
14. Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. University of Sheffield TREC-8 Q&A System. In Voorhees and Harman [27].
15. Harksoo Kim, Kyungsun Kim, Gary Geunbae Lee, and Jungyun Seo. Maya: A fast question-answering system based on a predicate answer indexer. In *Proc. of the ACL'01 workshop "Open-Domain Question Answering"*, pages 9–16, 2001.
16. Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.
17. D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 11, 1995.
18. Michael McCord, Arendse Bernth, Shalom Lappin, and Wlodek Zadrozny. Natural language processing within a slot grammar framework. *International Journal on Artificial Intelligence Tools*, 1(2):229–277, 1992.
19. Adam Meyers, Catherine Macleod, Roman Yangarber, Ralph Grishman, Leslie Barrett, and Ruth Reeves. Using NOMLEX to produce nominalization patterns for information extraction. In *Proceedings: the Computational Treatment of Nominals, Montreal, Canada, (Coling-ACL98 workshop)*, August 1998.
20. Diego Mollá. ExtrAns: An answer extraction system for unix manpages – on-line manual. Technical report, Computational Linguistics, University of Zurich, 1998. `http://www.ifi.unizh.ch/CL/`.
21. Diego Mollá, Gerold Schneider, Rolf Schwitter, and Michael Hess. Answer Extraction using a Dependency Grammar in ExtrAns. *Traitement Automatique de Langues (T.A.L.), Special Issue on Dependency Grammar*, 41(1):127–156, 2000.
22. Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. Extrans, an answer extraction system. *T.A.L. special issue on Information Retrieval oriented Natural Language Processing*, 2000.
23. Marius Pasca and Sanda Harabagiu. Answer mining from on-line documents. In *Proc. of the ACL'01 workshop "Open-Domain Question Answering"*, pages 38–45, 2001.
24. Gerold Schneider, Diego Mollá Aliod, and Michael Hess. Inkrementelle minimale logische formen für die antwortextraktion. 1999. Proceedings of 4th Linguistic Colloquium, University of Mainz, FASK, September 7-10, 1999.
25. Daniel D. Sleator and Davy Temperley. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292, 1993.
26. Ellen M. Voorhees. The TREC-8 Question Answering Track Report. In Voorhees and Harman [27].
27. Ellen M. Voorhees and Donna Harman, editors. *The Eighth Text REtrieval Conference (TREC-8)*. NIST, 2000.
28. Ellen M. Voorhees and Donna Harman, editors. *Proceedings of the Ninth Text REtrieval Conference (TREC-9), Gaithersburg, Maryland, November 13-16, 2000. Preliminary on-line version: http://trec.nist.gov/pubs/trec9/t9_proceedings.html*, 2001.