Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 33-40.

Performance Issues and Error Analysis in an Open-Domain Question Answering System

Dan Moldovan, Marius Paşca, Sanda Harabagiu and **Mihai Surdeanu**
Language Computer Corporation
Dallas, Texas
{moldovan,marius,sanda,mihai}@languagecomputer.com

Abstract

This paper presents an in-depth analysis of a state-of-the-art Question Answering system. Several scenarios are examined: (1) the performance of each module in a serial baseline system, (2) the impact of feedbacks and the insertion of a logic prover, and (3) the impact of various lexical resources. The main conclusion is that the overall performance depends on the depth of natural language processing resources and the tools used for answer finding.

1 Introduction

Aiming at returning brief answers in response to natural language questions, open-domain Question Answering (QA) systems represent an advanced application of natural language processing. The global metrics used in the QA track evaluations of the Text REtrieval Conference (TREC) (Voorhees, 1999) allow for the overall assessment of the QA system performance. As part of a relentless quest to improve QA systems, it is necessary to measure not only the global performance, but also the performance of each individual module and other architectural features. A detailed performance analysis indicates not only that the system fails to provide an answer but why the system failed. The performance analysis is useful to system designers who want to identify error sources and weak modules.

The QA literature from the last few years reports on global performance of various systems (Abney et al., 2000; Hovy et al., 2001). General evaluation metrics are discussed in (Voorhees and Tice, 2000) and (Breck et al., 2000) but, with few exceptions (Ittycheriah et al., 2001), little is said about in-depth error analysis in QA systems.

Since most QA systems consist of modules that are chained serially (Abney et al., 2000; Prager et al., 2000), the overall performance is controlled by their weakest link. In this case the error analysis is straightforward. Our system architecture uses several feedbacks which complicates significantly the error analysis.

This paper presents an in-depth performance analysis of a state-of-the-art QA system. Several configurations are examined; first, the performance of each module in a baseline chained architecture, then, the impact of feedbacks and the insertion of new advanced modules, and finally, the impact of various lexical resources. Our QA system was ranked high in the last three TREC QA track evaluations (cf. (Voorhees, 1999)). Therefore the results are representative to other QA serial architectures whose internal modules perform equivalent tasks (Abney et al., 2000) or employ similar lexical resources and tools (Hovy et al., 2001; Prager et al., 2000).

2 Taxonomy

The performance of a QA system is tightly coupled with the complexity of questions asked and the difficulty of answer extraction. For example, in TREC many systems were quite successful at providing correct answers to simpler, fact-seeking questions, but failed to answer questions that required reasoning or advanced linguistic analysis (Voorhees, 1999). From the combined set of 1460 evaluation questions, 70% of the participating systems answered successfully questions like Q1013: *"Where is Perth?"*, but none could find a correct answer to complex questions such as Q1165: *"What is the difference between AM radio stations and FM radio stations?"*.

Since performance is affected by the complexity of question processing, we first provide a broad taxonomy of QA systems.

2.1 Criteria

The taxonomy is based on several criteria that play an important role in building QA systems: (1) linguistic and knowledge resources, (2) natural language processing involved, (3) document processing, (4) reasoning methods, (5) whether or not answer is explicitly stated in a document, (6) whether or not answer fusion is necessary.

2.2 Classes of questions

Class 1. QA systems capable of processing factual questions
These systems extract answers as text snippets from one or more documents. Often the answer is found verbatim in a text or as a simple morphological variation. Typically the answers are extracted using empirical methods relying on keyword manipulations.

Class 2. QA systems enabling simple reasoning mechanisms
The characteristic of this class is that answers are found in snippets of text, but unlike in Class 1, inference is necessary to relate the question with the answer. More elaborate answer detection methods such as ontologies or codification of pragmatic knowledge are necessary. Semantic alternations, world knowledge axioms and simple reasoning methods are necessary. An example is Q198: *"How did Socrates died?"* where *die* has to be linked with *drinking poisoned wine.* WordNet and its extensions are sometimes used as sources of world knowledge.

Class 3. QA systems capable of answer fusion from different documents
In this class the partial answer information is scattered throughout several documents and answer fusion is necessary. The complexity here ranges from assembling simple lists to far more complex questions like script questions, (e.g. *"How do I assemble a bicycle?"*), or template-like questions (*"What management successions occurred at IBM in the past year?"*).

Class 4. Interactive QA systems
These systems are able to answer questions in the context of previous interactions with the user. As reported in (Harabagiu et al., 2001), processing a list of questions posed in a context involves complex reference resolution. Unlike typical reference resolution algorithms that associate anaphore with a referent, the reference imposed by context questions requires the association of an anaphora from the current question with either one of the previous questions, answers or their anaphora.

Class 5. QA systems capable of analogical reasoning
The characteristic of these systems is their ability to answer speculative questions similar to: *"Is the Fed going to raise interests at their next meeting?"*; *"Is the US out of recession?"*; *"Is the airline industry in trouble?"*.

Since most probably the answer to such questions is not explicitly stated in documents, simply because events may not have happened yet, QA systems from this class decompose the question into queries that extract pieces of evidence, after which answer is formulated using reasoning by analogy. The resources include ad-hoc knowledge bases generated from mining text documents clustered by the question topic. Associated with these knowledge sources are case-based reasoning techniques as well as methods for temporal reasoning, spatial reasoning and evidential reasoning.

Table 1: Distribution of TREC questions

Type	Number (%)
Class 1 (factual)	985 (67.5%)
Class 2 (simple-reasoning)	408 (27.9%)
Class 3 (fusion - list)	25 (1.7%)
Class 4 (interactive - context)	42 (2.9%)
Class 5 (speculative)	0 (0.0%)

Table 1 illustrates the distribution of TREC

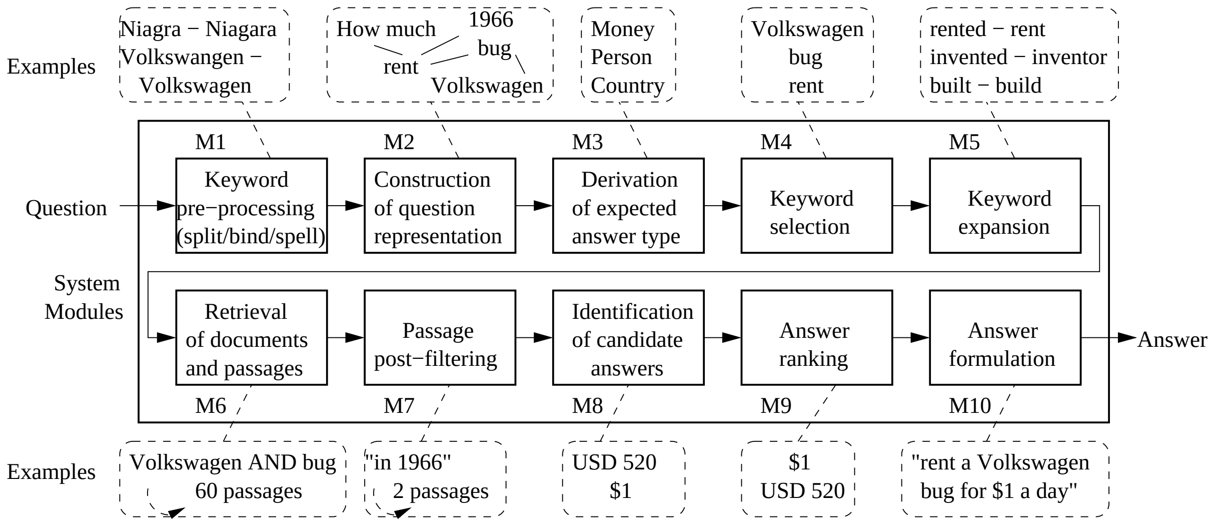

Figure 1: Architecture of baseline serial system (no feedbacks)

questions into the question classes. In addition to 1393 main-task questions collected from TREC-8, TREC-9 and TREC-2001, there are 25 list questions (e.g., *"Name 20 countries that produce coffee."*) and 42 context questions (e.g., *"How long was the Varyag?"*; *"How wide?"*).

3 Serial system architecture

This section introduces the serialized architecture of our QA system in which there are no feedbacks. The complete architecture with all the feedbacks is presented in a later section of the paper. As shown in Figure 1, the architecture consists of 10 modules performing several natural language processing tasks.

The first five modules correspond to question processing, the next two modules perform document and passage processing, and the last three modules perform answer processing.

M1 The individual question words are spell-checked. Words like *Volkswangen* and *Niagra* are expanded into their spelling variants *Volkswagen* and *Niagara.* If necessary, questions such as Q885: *"Rotary engine cars were made by what company?"* are rephrased into a normalized form where the wh-word (*what*) appears at the beginning, e.g. *"What company were rotary engine cars made by?"*.

M2 The input question is parsed and transformed into an internal representation (Harabagiu et al., 2000) capturing question concepts and binary dependencies between the concepts. Stop words (e.g., prepositions or determiners) are identified and removed from the representation. For illustration, the representation for Q013: *"How much could you rent a Volkswagen bug for in 1966?"* captures the binary dependency between the concepts *rent* and *1966.*

M3 The mapping of certain question dependencies on a WordNet-based answer type hierarchy disambiguates the semantic category of the expected answers (Paşca and Harabagiu, 2001). For example, the dependency between *How much* and *rent* for Q013 is exploited to derive the expected answer type *Money.* The answer type is passed to subsequent modules for the identification of possible answers (all monetary values).

M4 Based mainly on part of speech information, a subset of the question concepts are selected as keywords for accessing the underlying document collection. A passage retrieval engine accepts Boolean queries built from the selected keywords, e.g. *Volkswagen* AND *bug.* The retrieval engine returns passages that contain all keywords specified in the Boolean query. Therefore keyword selection is a sensitive task. If the wrong question word (e.g. *much*) is included in the Boolean query (*much* AND *Volkswagen*

AND *bug*), the retrieval is insuccessful since the passages containing the correct answers are missed.

M5 Before the construction of Boolean queries for actual retrieval, the selected keywords are expanded with morphological, lexical or semantic alternations. The alternations correspond to other forms in which the question concepts may occur in the answers. For example, *rented* is expanded into *rent*.

M6 The retrieval engine returns the documents containing all keywords specified in the Boolean queries. The documents are then further restricted to smaller text passages where all keywords are located in the proximity of one another. Each retrieved passage includes additional text (extra lines) before the earliest and after the latest keyword match. For illustration, consider Q005: *"What is the name of the managing director of Apricot Computer?* and the associated Boolean query *Apricot* AND *Computer* AND *director*. The relevant text fragment from the document collection is *"Dr Peter Horne, managing director of Apricot Computers"*. Unless additional text is included in the passages, the actual answer *Peter Horne* would be missed because it occurs before all matched keywords, namely *director*, *Apricot* and *Computer*.

M7 The retrieved passages are further refined for enhanced precision. Passages that do not satisfy the semantic constraints specified in the question are discarded. For example, some of the passages retrieved for Q013 do not satisfy the date constraint *1966*. Out of the 60 passages returned by the retrieval engine for Q013, 2 passages are retained after passage post-filtering.

M8 The search for answers within the retrieved passages is restricted to those candidates corresponding to the expected answer type. If the expected answer type is a named entity such as MONEY, the candidates (*$1*, *USD 520*) are identified with a named entity recognizer. Conversely, if the answer type is a DEFINITION, e.g. Q903: *"What is autism?"*, the candidates are obtained by matching a set of answer patterns on the passages.

M9 Each candidate answer receives a relevance score according to lexical and proximity features such as distance between keywords, or the occurrence of the candidate answer within an apposition. The candidates are sorted in decreasing order of their scores.

M10 The system selects the candidate answers with the highest relevance scores. The final answers are either fragments of text extracted from the passages around the best candidate answers, or they are internally generated.

4 Error analysis for the baseline serial system

4.1 Performance experiments

The system was tested on 1460 questions collected from TREC-8, 9 and TREC-2001. Answers were extracted from a 3 Gbyte text collection containing about 1 million documents from sources such as Los Angeles Times and Wall Street Journal. Each answer has 50 bytes.

The accuracy was measured by the Mean Reciprocal Rate (MRR) metric used by NIST in the TREC QA evaluations (Voorhees, 1999). The reciprocal ranking basically assigns a number equal to 1/R where R is the rank of the correct answer. Only the first 5 answers are considered, thus R is less or equal to 5. When the system does not return a correct answer in top 5, the precision score for that question is zero. The overall system precision is the mean of the individual scores. System answers were measured against correct answers provided by NIST.

4.2 Module errors

The inspection of internal traces, at various checkpoints inserted after each module from Figure 1, reveals the system errors for each evaluation question. The goal in this experiment is to identify the earliest module in the chain (from left to right) that prevents the system to find the right answer, i.e. causes the error.

As shown in Table 2, question pre-processing is responsible for 7.1% of the errors distributed among module M1 (1.9%) and M2 (5.3%). Most errors in module M2 are due to incorrect parsing (4.5%). Two of the ten modules (M3 and M5) account for more than half of the errors. The failure of either module makes it hard (or impos-

Table 2: Distribution of errors per system module

Module	Module definition	Errors (%)
(M1)	Keyword pre-processing (split/bind/spell check)	1.9
(M2)	Construction of internal question representation	5.2
(M3)	Derivation of expected answer type	36.4
(M4)	Keyword selection (incorrectly added or excluded)	8.9
(M5)	Keyword expansion desirable but missing	25.7
(M6)	Actual retrieval (limit on passage number or size)	1.6
(M7)	Passage post-filtering (incorrectly discarded)	1.6
(M8)	Identification of candidate answers	8.0
(M9)	Answer ranking	6.3
(M10)	Answer formulation	4.4

sible) for subsequent modules to perform their task. Whenever the derivation of the expected answer type (module M3) fails, the set of candidate answers identified in the retrieved passages is either empty in 28.2% of the cases (when the answer type is unknown) or contains the wrong entities for 8.2% (when the answer type is incorrect). If the keywords used for passage retrieval are not expanded with the semantically related forms occurring in the answers (module M5), the relevant passages are missed.

The selection of keywords from the internal question representation (module M4) coupled with the keyword expansion (module M5) generate 34.6% of the errors. Both these modules affect the output of passage retrieval, since the set of retrieved passages depends on the Boolean queries built and submitted to the retrieval engine by the QA system.

Modules M6 and M7 are responsible for the retrieval of passages where answers may actually occur. Their combined errors is 3.2%. In module M6 there are parameters to control the number of retrieved documents and passages, as well as the size of each passage.

Answer processing is done in modules M8 through M10. When the expected answer type is correctly detected, the identification of the candidate answers (module M8) produces 8.0% errors. 3.1% errors are due to named entity recognition (incomplete dictionaries) and 4.9% are due to spurious answer pattern matching. Modules M9 and M10 fail to rank the correct answer within the top 5 returned in 10.7% of the cases. Module M9 fails if the correct answer candidate is not ranked within the top 5, whereas M10 fails if the returned answer string is incomplete, namely it does not fit within 50 bytes.

4.3 Resource errors

The second set of experiments consists of disabling the main natural language resources used in the QA system, namely the access to WordNet and the named entity recognizer, to assess their impact on the overall answer accuracy. Note that the parser is an integral part of our question processing model and therefore it is impractical to disable it.

Denote with b the baseline system performance when all resources are enabled. The precision score (MRR) drops to $0.59b$ if WordNet is disabled. The derivation of the answer type (module M3) and keyword expansion (module M5) from Figure 1 are the two modules that are most influenced by WordNet. For example, the WordNet noun hierarchies specify that the concept *pilot* is a specialization of *aviator*, which in turn is a kind of *person*. The answer type for Q037: *"What was the name of the US helicopter pilot shot down over North Korea?"* is *Person*. The system cannot derive the answer type correctly unless it has access to WordNet hierarchies because the ambiguous question stem *What* alone does not provide any clue as to what the expected answer type is. A closer anal-

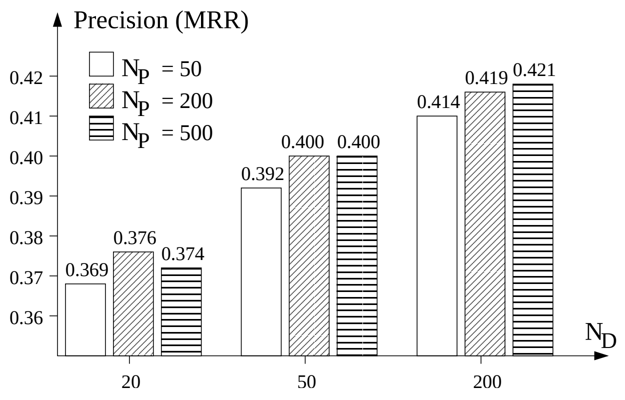

Figure 2: Impact of maximum number of documents and passages processed

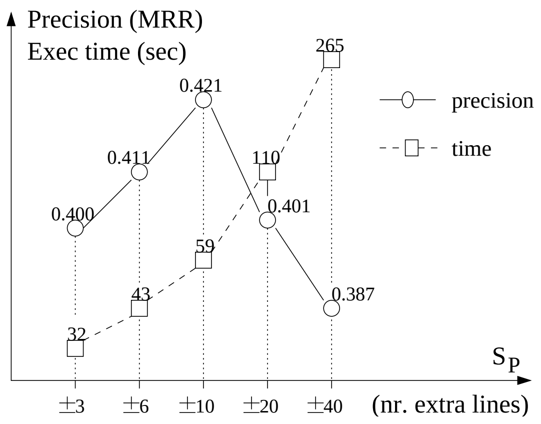

Figure 3: Impact of passage size on precision and execution time

ysis shows that the performance drop is more significant for the *What* questions. When WordNet is disabled, the MRR for the *What* questions drops to *0.37b* as compared to *0.59b* for the entire set. This result indicates that the availability of lexico-semantic information becomes more important for difficult questions.

By disabling the named entity recognizer, the answer processing lacks the semantic information necessary to identify candidate answers. Loose approximations for the candidate answers are computed based strictly on keywords matching. In this case the precision drops to *0.32b*.

5 Impact of system parameters

The quantitative performance of the QA system is largely dependent on the amount of text retrieved from the document collection; the more text is retrieved the better the chance of finding the answer. However, practical QA systems cannot afford to apply time consuming NLP techniques (especially parsing) to very large amounts of text. Retrieval parameters are used to provide trade-offs between the amount of text passed into the answer processing module and the accuracy of the extracted answers. The QA system has the following *retrieval parameters*:

- N_D - the maximum number of documents retrieved from a sub-collection (default value 200 for each of 12 sub-collections); [1]
- N_P - the maximum number of passages processed to identify candidate answers (default value 500).
- S_P - the size allowance for each retrieved passage (default value 10 lines before the earliest and after the latest keyword match);

When N_D and N_P are set to smaller values, the execution time is lower but relevant documents and passages may be missed. Figure 2 illustrates the impact of the parameters N_D and N_P on the precision computed over the entire set of 1460 test questions. The higher the number of documents retrieved, the higher the precision score. It is apparent that N_P has a relatively smaller impact on the precision than N_D. This is due to the fact that the retrieved passages are re-ordered based on a set of lexical features, such that the identification of the candidate answers is performed on the top N_P re-ordered passages.

Figure 3 shows the possible trade-off between overall precision and execution time as a function of the passage size S_P. Interestingly, the highest precision score occurs for the default setting, ±10. When S_P is smaller, the answers are missed because they do not fit in the retrieved passages. When S_P is larger, the actually relevant text fragments are submerged in a large amount of text. Consequently the answer ranking module (M9 from Figure 1) sorts through a very large number of candidate answers, and it

[1]The passage retrieval engine manages the TREC document collection as a set of 12 separate sub-collections

does not always rank the correct answers within the top 5 returned.

6 Impact of Feedbacks

The results presented in previous sections correspond to the serialized baseline architecture from Figure 1. That architecture is in fact a simplified version of our system which uses several feedbacks to boost the overall performance.

As shown in Figure 4, the architecture with feedbacks extends the serialized architecture in several ways. Keyword expansion (module M5) is enhanced to include lexico-semantic alternations from WordNet. A new module for logic proving and justification of the answers is inserted before answer ranking. In addition, three loops become an integral part of the system: the passage retrieval loop (loop 1); the lexico-semantic loop (loop 2); and the logic proving loop (loop 3).

As part of loop 1, the Q/A system adjusts Boolean queries before passing them to the retrieval engine. If the output from the retrieval engine is too small, a keyword is dropped and retrieval resumed. If the output is too large, a keyword is added and a new iteration started, until the output size is neither too large, nor too small. When lexico-semantic connections from the question to the retrieved passages are not possible, loop 2 is triggered. Question keywords are replaced with WordNet-based alternations and retrieval is resumed. Loop 3 relies on a logic prover that verifies the unifications between the question and logic forms. When the unifications fail, the keywords are expanded with semantically related alternations and retrieval resumes.

Table 3: Impact of feedbacks on precision

Feedback added	Precision (MRR)	Incremental enhancement
none	$0.421=b$	0%
Passage retrieval (loop 1)	$0.468=b_1$	b+11%
Lexico-semantic (loop 2)	$0.542=b_2$	b_1+15%
Proving (loop 3)	$0.572=b_3$	b_2+5%

Table 3 illustrates the impact of the retrieval loops on the answer accuracy. The knowledge brought into the question answering process by lexico-semantic alternations has the highest individual contribution, followed by the mechanism of adding/dropping keywords.

The insertion of the logic proving module adds a new complexity layer to answer processing, enabling more trade-offs between processing complexity and answer accuracy. Table 4 shows the overall precision for four different settings. The first setting, *direct extraction*, corresponds to the simplest QA system that does not use any NLP techniques or resources. The answers are extracted from the start of each passage, and returned in the order in which the passages were retrieved. The precision is only 0.028. When the NLP techniques are enabled, with the exception of the derivation of the expected answer type, the precision improves from 0.028 to 0.150. The answer accuracy is still limited because the candidate answers cannot be properly identified without knowing their semantic category (persons, cities and so forth). If the derivation of the expected answer type is also enabled, the precision score changes to 0.468. Finally, when all feedbacks are enabled the highest overall precision of 0.572 is achieved. Comparatively, the answer processing modules of other QA systems usually span over levels 2 and 3 from Table 4.

Table 4: Performance of answer processing

Answer processing complexity level	Modules used	Precision (MRR)
(1) Direct extraction	M1-M6, M10	0.028
(2) Lexical matching	M1-M7, M9-M10	0.150
(3) Semantic matching	M1-M10	0.468
(4) Feedbacks enabled	all	0.572

The final precision scores for TREC-8, TREC-9 and TREC-2001 are respectively 0.555, 0.580, and 0.570. Therefore the precision did not vary much in spite of the higher degree of difficulty. This is due to the increased use of natural lan-

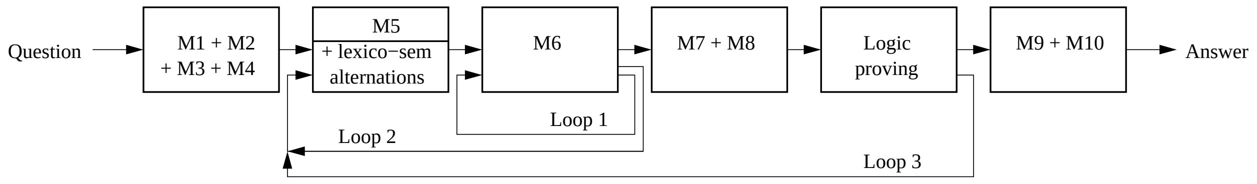

Figure 4: Architecture with feedbacks

guage processing in our system.

7 Conclusions

The main conclusion is that the overall performance of QA systems is directly related to the depth of natural language processing resources and the tools used for answer finding. As shown in Table 4, the performance of information retrieval techniques is significantly enhanced when lexico-semantic information is fully exploited throughout the answer finding process.

Table 2 illustrates that the performance bottlenecks of our QA system are due to two modules, namely the derivation of the expected answer type and the keyword expansion. The bottlenecks are not specific to our QA system but reflect the limitations of current QA technologies. Question answering systems perform better when the relevant passages and the candidate answers are clearly defined in the questions. The main problem is the lack of powerful schemes and algorithms for modeling complex questions in order to derive as much information as possible, and for performing a well-guided search through thousands of text documents.

The lexico-semantic information imported in the QA system through the retrieval feedbacks brings consistent improvements over serial processing. Per-component errors are spread uniformly over the first four classes of question complexities, indicating how our system improved over the years.

References

S. Abney, M. Collins, and A. Singhal. 2000. Answer extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP-2000)*, pages 296–301, Seattle, Washington.

E. Breck, J. Burger, L. Ferro, L. Hirschman, D. House, M. Light, and I. Mani. 2000. How to evaluate your question answering system every day ... and still get real work done. In *Proceedings of the 2nd Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

S. Harabagiu, M. Paşca, and S. Maiorano. 2000. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Saarbrucken, Germany.

S. Harabagiu, D. Moldovan, M. Paşca, M. Surdeanu, R. Mihalcea, R. Gîrju, V. Rus, F. Lăcătuşu P. Morărescu, and R. Bunescu. 2001. Answering complex, list and context questions with lcc's question-answering server. In *Proceedings of the 10th Text REtrieval Conference (TREC-2001)*, Gaithersburg, Maryland. NIST.

E. Hovy, L. Gerber, U. Hermjakob, C.Y. Lin, and D. Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the Human Language Technology Conference (HLT-2001)*, San Diego, California.

A. Ittycheriah, M. Franz, W. Zhu, and A. Ratnaparkhi. 2001. Question answering using maximum-entropy components. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, Pittsburgh, Pennsylvania.

M. Paşca and S. Harabagiu. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01), Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, Pittsburgh, Pennsylvania, June.

J. Prager, E. Brown, A. Coden, and D. Radev. 2000. Question answering by predictive annotation. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval (SIGIR-2000)*, pages 184–191, Athens, Greece.

E.M. Voorhees and D.M. Tice. 2000. Building a question-answering test collection. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval (SIGIR-2000)*, pages 200–207, Athens, Greece.

E.M. Voorhees. 1999. The TREC-8 Question Answering track report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, pages 77–82, Gaithersburg, Maryland. NIST.