In *Proceedings of the Sixth Applied Natural Language Processing Conference* (ANLP-2000), 180–187, 2000.

Examining the Role of Statistical and Linguistic Knowledge Sources in a General-Knowledge Question-Answering System

Claire Cardie[1] and **Vincent Ng**[1] and **David Pierce**[1] and **Chris Buckley**[2]
Department of Computer Science, Cornell University, Ithaca, NY 14853[1]
SaBIR Research[2]
E-mail: cardie,yung,pierce@cs.cornell.edu, chrisb@sabir.com

Abstract

We describe and evaluate an implemented system for general-knowledge question answering. The system combines techniques for standard ad-hoc information retrieval (IR), query-dependent text summarization, and shallow syntactic and semantic sentence analysis. In a series of experiments we examine the role of each statistical and linguistic knowledge source in the question-answering system. In contrast to previous results, we find first that statistical knowledge of word co-occurrences as computed by IR vector space methods can be used to quickly and accurately locate the relevant documents for each question. The use of query-dependent text summarization techniques, however, provides only small increases in performance and severely limits recall levels when inaccurate. Nevertheless, it is the text summarization component that allows subsequent linguistic filters to focus on relevant passages. We find that even very weak linguistic knowledge can offer substantial improvements over purely IR-based techniques for question answering, especially when smoothly integrated with statistical preferences computed by the IR subsystems.

1 Introduction

In this paper, we describe and evaluate an implemented system for general-knowledge question answering. Open-ended question-answering systems that allow users to pose a question of any type, in any language, without domain restrictions, remain beyond the scope of today's text-processing systems. We investigate instead a restricted, but nevertheless useful variation of the problem (TREC-8, 2000):

> Given a large text collection and a set of questions specified in English, find answers to the questions in the collection.

In addition, the restricted task guarantees that:

- the answer exists in the collection,
- all supporting information for the answer lies in a single document, and
- the answer is short — less than 50 bytes in length.

Consider, for example, the question *Which country has the largest part of the Amazon rain forest?*, taken from the TREC8 Question Answering development corpus. The answer (in document LA032590-0089) is *Brazil*.

Previous research has addressed similar question-answering (QA) scenarios using a variety of natural language processing (NLP) and information retrieval (IR) techniques. Lehnert (1978) tackles the difficult task of answering questions in the context of story understanding. Unlike our restricted QA task, questions to Lehnert's system often require answers that are not explicitly mentioned in the story. Her goal then is to answer questions by making inferences about actions and actors in the story using world knowledge in the form of scripts, plans, and goals (Schank and Abelson, 1977). More recently, Burke et al. (1995; 1997) describe a system that answers natural language questions using a database of question-answer pairs built from existing frequently-asked question (FAQ) files. Their FAQFinder system uses IR techniques to match the given question to questions in the database. It then uses the WordNet lexical semantic knowledge base (Miller et al., 1990; Fellbaum, 1998) to improve the quality of the match.

Kupiec (1993) investigates a closed-class QA task that is similar in many respects to the TREC8 QA task that we address here: the system answers general-knowledge questions using an encyclopedia. In addition, Kupiec assumes that all answers are noun phrases. Although our task does not explicitly include a "noun phrase" constraint, the answer length restriction effectively imposes the same bias toward noun phrase answers. Kupiec's MURAX system applies a combination of statistical (IR) and linguistic (NLP) techniques. A series of secondary boolean search queries with proximity constraints is combined with shallow parsing methods to find relevant sections of the encyclopedia, to extract answer hypotheses, and to confirm phrase relations specified in the question. In an evaluation on 70 "Trivial

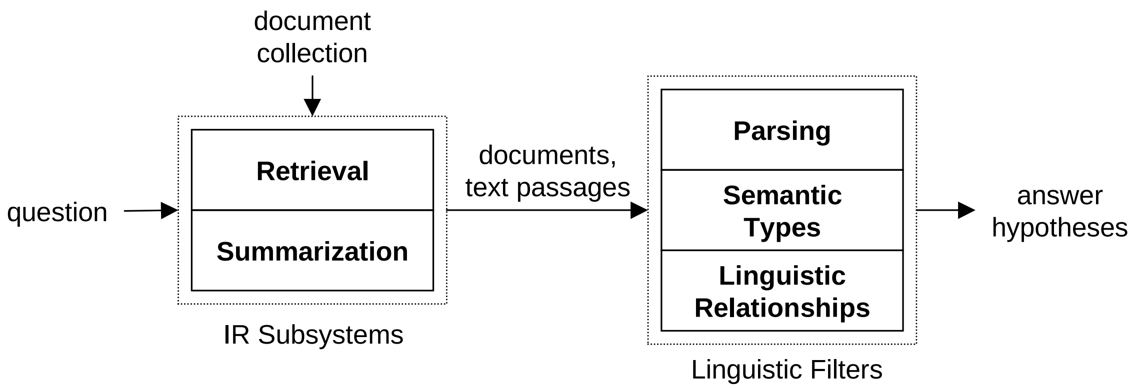

Figure 1: General Architecture of the Question-Answering System

Pursuit" *who* and *what* questions, Kupiec concludes that robust natural language analysis can add to the quality of the information retrieval process. In addition, he claims that, for their closed-class QA task, vector space IR methods (Salton et al., 1975) appear inadequate.

We present here a new approach to the restricted question-answering task described above. Like MURAX, our system draws from both statistical and linguistic sources to find answers to general-knowledge questions. The underlying architecture of the system, however, is very different: it combines vector space IR techniques for document retrieval, a vector space approach to query-dependent text summarization, shallow corpus-based syntactic analysis, and knowledge-based semantic analysis. We evaluate the system on the TREC8 QA development corpus as well as the TREC8 QA test corpus. In particular, all parameters for the final QA system are determined using the development corpus. Our current results are encouraging but not outstanding: the system is able to correctly answer 22 out of 38 of the development questions and 91 out of 200 of the test questions given five guesses for each question. Furthermore, the first guess is correct for 16 out of the 22 development questions and 53 out of 91 of the test questions.

More importantly, we investigate the relative role of each statistical and linguistic knowledge source in the proposed IR/NLP question-answering system. In contrast to previous results, we find that statistical knowledge of word co-occurrences as computed by vector space models of IR can be used to quickly and accurately locate relevant documents in the restricted QA task. When used in isolation, vector space methods for query-dependent text summarization, however, provide relatively small increases in performance. In addition, we find that the text summarization component can severely limit recall levels. Nevertheless, it is the summarization component that allows the linguistic filters to focus on relevant passages. In particular, we find that very weak linguistic knowledge can offer substantial improvements over purely IR-based techniques for question answering, especially when smoothly integrated with the statistical preferences computed by the IR subsystems.

In the next section, we describe the general architecture of the question-answering system. Section 3 describes the baseline system and its information retrieval component. Sections 4–7 describe and evaluate a series of variations to the baseline system that incorporate, in turn, query-dependent text summarization, a syntactic filter, a semantic filter, and an algorithm that allows syntactic knowledge to influence the initial ordering of summary extracts. Section 8 compares our approach to some of those in the recent TREC8 QA evaluation (TREC-8, 2000) and describes directions for future work.

2 System Architecture

The basic architecture of the question-answering system is depicted in Figure 1. It contains two main components: the IR subsystems and the linguistic filters. As a preliminary, offline step, the **IR subsystem** first indexes the text collection from which answers are to be extracted. Given a question, the goal of the IR component is then to return a ranked list of those text chunks (e.g. documents, sentences, or paragraphs) from the indexed collection that are most relevant to the query and from which answer hypotheses can be extracted. Next, the QA system optionally applies one or more **linguistic filters** to the text chunks to extract an ordered list of answer hypotheses. The top hypotheses are concatenated to form five 50-byte guesses as allowed by the TREC8 guidelines. Note that many of these guesses may be difficult to read and judged as incorrect by the

TREC8 assessors: we will also describe the results of generating single phrases as guesses wherever this is possible.

In the sections below, we present and evaluate a series of instantiations of this general architecture, each of which makes different assumptions regarding the type of information that will best support the QA task. The next section begins by describing the baseline QA system.

3 The Vector Space Model for Document Retrieval

It is clear that a successful QA system will need some way to find the documents that are most relevant to the user's question. In a baseline system, we assume that standard IR techniques can be used for this task. In contrast to MURAX, however, we hypothesize that the vector space retrieval model will suffice. In the vector space model, both the question and the documents are represented as vectors with one entry for every unique word that appears in the collection. Each entry is the *term weight*, a real number that indicates the presence or absence of the word in the text. The similarity between a question vector, $Q = q_1, q_2, \ldots, q_n$, and a document vector, $D = d_1, d_2, \ldots, d_n$, is traditionally computed using a cosine similarity measure:

$$Sim(Q, D) = \sum_{i=1}^{n} d_i \cdot q_i$$

Using this measure, the IR system returns a ranked list of those documents most similar to the question.

The Baseline QA System: The Smart Vector Space Model. For the IR component of the baseline QA system, we use Smart (Salton, 1971), a sophisticated text-processing system based on the vector space model and employed as the retrieval engine for a number of the top-performing systems at recent Text REtrieval Conferences (e.g. Buckley et al., 1998a, 1998b). Given a question, Smart returns a ranked list of the documents most relevant to the question. For the baseline QA system and all subsequent variations, we use Smart with standard term-weighting strategies[1] and do not use automatic relevance feedback (Buckley, 1995). In addition, the baseline system applies no linguistic filters. To generate answers for a particular question, the system starts at the beginning of the top-ranked document returned by Smart for the question and constructs five 50-byte chunks consisting of document text with stopwords removed.

[1] We use *Lnu* term weighting for documents and *ltu* term weighting for the question (Singhal et al., 1996).

Evaluation. As noted above, we evaluate each variation of our QA system on 38 TREC8 development questions and 200 TREC8 test questions. The indexed collection is TREC disks 4 and 5 (without Congressional Records). Results for the baseline Smart IR QA system are shown in the first row of Table 1. The system gets 3 out of 38 development questions and 29 out of 200 test questions correct. We judge the system correct if any of the five guesses contains each word of one of the answers. The final column of results shows the mean answer rank across all questions correctly answered.

Smart is actually performing much better than its scores would suggest. For 18 of the 38 development questions, the answer appears in the top-ranked document; for 33 questions, the answer appears in one of the top seven documents. For only two questions does Smart fail to retrieve a good document in the top 25 documents. For the test corpus, over half of the 200 questions are answered in the top-ranked document (110); over 75% of the questions (155) are answered in top five documents. Only 19 questions were not answered in the top 20 documents.

4 Query-Dependent Text Summarization for Question Answering

We next hypothesize that query-dependent text summarization algorithms will improve the performance of the QA system by focusing the system on the most relevant portions of the retrieved documents. The goal for query-dependent summarization algorithms is to provide a short summary of a document *with respect to a specific query.* Although a number of methods for query-dependent text summarization are beginning to be developed and evaluated in a variety of realistic settings (Mani et al., 1999), we again propose the use of vector space methods from IR, which can be easily extended to the summarization task (Salton et al., 1994):

1. Given a question and a document, divide the document into chunks (e.g. sentences, paragraphs, 200-word passages).
2. Generate the vector representation for the question and for each document chunk.
3. Use the cosine similarity measure to determine the similarity of each chunk to the question.
4. Return as the query-dependent summary the most similar chunks up to a predetermined summary length (e.g. 10% or 20% of the original document).

This approach to text summarization was shown to be quite successful in the recent SUMMAC evaluation of text summarization systems (Mani et al., 1999; Buckley et al., 1999). Our general assumption here is that IR approaches can be used to quickly and accurately find both relevant documents and relevant document portions. In related work, Chali *et al.* (1999) also propose text summarization techniques as a primary component for their QA system. They employ a combination of vector-space methods and lexical chaining to derive their sentence-based summaries. We hypothesize that deeper analysis of the summary extracts is better accomplished by methods from NLP that can determine syntactic and semantic relationships between relevant constituents. There is a risk in using query-dependent summaries to focus the search for answer hypotheses, however: if the summarization algorithm is inaccurate, the desired answers will occur outside of the summaries and will not be accessible to subsequent components of the QA system.

The Query-Dependent Text Summarization QA System. In the next version of the QA system, we augment the baseline system to perform query-dependent text summarization for the top k retrieved documents. More specifically, the IR subsystem returns the summary extracts (sentences or paragraphs) for the top k documents after sorting them according to their cosine similarity scores w.r.t. the question. As before, no linguistic filters are applied, and answers are generated by constructing 50-byte chunks from the ordered extracts after removing stopwords. In the experiments below, $k = 7$ for the development questions and $k = 6$ for the test questions.[2]

Evaluation. Results for the Text Summarization QA system using sentence-based summaries are shown in the second row of Table 1. Here we see a relatively small improvement: the system now answers four development and 45 test questions correctly. The mean answer rank, however, improves noticeably from 3.33 to 2.25 for the development corpus and from 3.07 to 2.67 for the test corpus. Paragraph-based summaries yield similar but slightly smaller improvements; as a result, sentence summaries are used exclusively in subsequent sections. Unfortunately, the system's reliance on query-dependent text summarization actually limits its potential: in only 23 of the 38 development questions (61%), for example, does the correct answer appear in the summary for one of the top $k = 7$ documents. The QA system cannot hope to answer correctly any of the remaining 15 questions. For only 135 of the 200 questions in the test corpus (67.5%) does the correct answer appear in the summary for one of the top $k = 6$ documents.[3] It is possible that automatic relevance feedback or coreference resolution would improve performance. We are investigating these options in current work.

The decision of whether or not to incorporate text summarization in the QA system depends, in part, on the ability of subsequent processing components (i.e. the linguistic filters) to locate answer hypotheses. If subsequent components are very good at discarding implausible answers, then summarization methods may limit system performance. Therefore, we investigate next the use of two linguistic filters in conjunction with the query-dependent text summarization methods evaluated here.

5 Incorporating the Noun Phrase Filter

The restricted QA task that we investigate requires answers to be short — no more than 50 bytes in length. This effectively eliminates *how* or *why* questions from consideration. Almost all of the remaining question types are likely to have noun phrases as answers. In the TREC8 development corpus, for example, 36 of 38 questions have noun phrase answers.

As a result, we next investigate the use of a very simple linguistic filter that considers only noun phrases as answer hypotheses. The filter operates on the ordered list of summary extracts for a particular question and produces a list of answer hypotheses, one for each noun phrase (NP) in the extracts in the left-to-right order in which they appeared.

The NP-based QA System. Our implementation of the NP-based QA system uses the Empire noun phrase finder, which is described in detail in Cardie and Pierce (1998). Empire identifies *base NPs* — non-recursive noun phrases — using a very simple algorithm that matches part-of-speech tag sequences based on a learned noun phrase grammar. The approach is able to achieve 94% precision and recall for base NPs derived from the Penn Treebank Wall Street Journal (Marcus et al., 1993). In the experiments below, the NP filter follows the application of the document retrieval and text summarization components. Pronoun answer hypotheses are discarded, and the NPs are assembled into 50-byte chunks.

Evaluation. Results for the NP-based QA system are shown in the third row of Table 1. The noun phrase filter markedly improves system performance for the development corpus, nearly dou-

[2] The value for k was chosen so that at least 80% of the questions in the set had answers appearing in the retrieved documents ranked 1–k. We have not experimented extensively with many values of k and expect that better performance can be obtained by tuning k for each text collection.

[3] Paragraph-based summaries provide better coverage on the test corpus than sentence-based summaries: for 151 questions, the correct answer appears in the summary for one of the top k documents. This suggests that paragraph summaries might be better suited for use with more sophisticated linguistic filters that are capable of discerning the answer in the larger summary.

	Development Corpus			Test Corpus		
Question-Answering System Variation	Correct	(%)	MAR	Correct	(%)	MAR
Smart Vector Space Model	3/38	0.079	3.33	29/200	0.145	3.07
Query-Dependent Text Summarization	4/38	0.105	2.25	45/200	0.225	2.67
Text Summarization + NPs	7/38	0.184	2.29	50/200	0.250	2.66
Text Summarization + NPs + Semantic Type	21/38	0.553	1.38	86/200	0.430	1.90
Text Summarization with Syntactic Ordering + NPs + Semantic Type	22/38	0.579	1.32	91/200	0.455	1.82

Table 1: Evaluation of the Role of Statistical and Limited Linguistic Knowledge for the TREC8 Question Answering Task. Results for 38 development and 200 test questions are shown. The mean answer rank (MAR) is computed w.r.t. all questions correctly answered.

bling the number of questions answered correctly. We found these results somewhat surprising since this linguistic filter is rather weak: we expected it to work well only in combination with the semantic filter described below. The noun phrase filter has much less of an effect on the test corpus, improving performance on questions answered from 45 to 50.

In a separate experiment, we applied the NP filter to the baseline system that includes no text summarization component. Here the NP filter does not improve performance — the system gets only two questions correct. This indicates that the NP filter depends critically on the text summarization component. As a result, we will continue to use query-dependent text summarization in the experiments below.

The NP filter provides the first opportunity to look at single-phrase answers. The preceding QA systems produced answers that were rather unnaturally chunked into 50-byte strings. When such chunking is disabled, only one development and 20 test questions are answered. The difference in performance between the NP filter with chunking and the NP filter alone clearly indicates that the NP filter is extracting good guesses, but that subsequent linguistic processing is needed to promote the best guesses to the top of the ranked guess list.

6 Incorporating Semantic Type Information

The NP filter does not explicitly consider the question in its search for noun phrase answers. It is clear, however, that a QA system must pay greater attention to the syntactic and semantic constraints specified in the question. For example, a question like *Who was president of the US in 1995?* indicates that the answer is likely to be a person. In addition, there should be supporting evidence from the answer document that the person was *president*, and, more specifically, held this office *in the US* and *in 1995*.

We introduce here a second linguistic filter that considers the primary semantic constraint from the question. The filter begins by determining the *question type*, i.e. the semantic type requested in the question. It then takes the ordered set of summary extracts supplied by the IR subsytem, uses the syntactic filter from Section 5 to extract NPs, and generates an answer hypothesis for every noun phrase that is semantically compatible with the question type. Our implementation of this semantic class filter is described below. The filter currently makes no attempt to confirm other linguistic relations mentioned in the question.

The Semantic Type Checking QA System. For most questions, the question word itself determines the semantic type of the answer. This is true for *who*, *where*, and *when* questions, for example, which request a *person*, *place*, and *time* expression as an answer. For many *which* and *what* questions, however, determining the question type requires additional syntactic analysis. For these, we currently extract the head noun in the question as the question type. For example, in *Which country has the largest part of the Amazon rain forest?* we identify *country* as the question type. Our heuristics for determining question type were based on the development corpus and were designed to be general, but have not yet been directly evaluated on a separate question corpus.

Given the question type and an answer hypothesis, the Semantic Type Checking QA System then uses WordNet to check that an appropriate ancestor-descendent relationship holds. Given *Brazil* as an answer hypothesis for the above question, for example, Wordnet's type hierarchy confirms that *Brazil* is a subtype of *country*, allowing the system to conclude that the semantic type of the answer hypothesis matches the question type.

For words (mostly proper nouns) that do not appear in WordNet, heuristics are used to determine semantic type. There are heuristics to recognize 13 basic question types: Person, Location, Date, Month, Year, Time, Age, Weight, Area, Volume, Length, Amount, and Number. For Person questions, for example, the system relies primarily on a rule that checks for capitalization and abbreviations in order to identify phrases that correspond to people. There are approximately 20 such rules that together cover all 13 question types listed above. The rules effectively operate as a very simple named entity identifier.

Evaluation. Results for the Semantic Type Checking variation of the QA system are shown in the fourth row of Table 1. Here we see a dramatic increase in performance: the system answers three times as many development questions (21) correctly over the previous variation. This is especially encouraging given that the IR and text summarization components limit the maximum number correct to 23. In addition, the mean answer rank improves from 2.29 to 1.38. A closer look at Table 1, however, indicates problems with the semantic type checking linguistic filter. While performance on the development corpus increases by 37 percentage points (from 18.4% correct to 55.3% correct), relative gains for the test corpus are much smaller. There is only an improvement of 18 percentage points, from 25.0% correct (50/200) to 43.0% correct (86/200). This is a clear indication that the heuristics used in the semantic type checking component, which were designed based on the development corpus, do not generalize well to different question sets. Replacing the current heuristics with a Named Entity identification component or learning the heuristics using standard inductive learning techniques should help with the scalability of this linguistic filter.

Nevertheless, it is somewhat surprising that very weak syntactic information (the NP filter) and weak semantic class information (question type checking) can produce such improvements. In particular, it appears that it is reasonable to rely implicitly on the IR subsystems to enforce the other linguistic relationships specified in the query (e.g. that *Clinton* is *president*, that this office was held *in the US* and *in 1995*).

Finally, when 50-byte chunking is disabled for the semantic type checking QA variation, there is a decrease in the number of questions correctly answered, to 19 and 57 for the development and test corpus, respectively.

7 Syntactic Preferences for Ordering Summary Extracts

Syntactic and semantic linguistic knowledge has been used thus far as post-processing filters that locate and confirm answer hypotheses from the statistically specified summary extracts. We hypothesized that further improvements might be made by allowing this linguistic knowledge to influence the initial ordering of text chunks for the linguistic filters. In a final system, we begin to investigate this claim. Our general approach is to define a new scoring measure that operates on the summary extracts and can be used to reorder the extracts based on linguistic knowledge.

The QA System with Linguistic Reordering of Summary Extracts. As described above, our final version of the QA system ranks summary extracts according to both their vector space similarity to the question as well as linguistic evidence that the answer lies within the extract. In particular, each summary extract E for question q is ranked according to a new score, s_q:

$$s_q(E) = w(E) \cdot LR_q(E)$$

The intuition behind the new score is to prefer summary extracts that exhibit the same linguistic relationships as the question (as indicated by LR_q) and to give more weight (as indicated by w) to linguistic relationship matches in extracts from higher-ranked documents. More specifically, $LR_q(E)$ is the number of linguistic relationships from the question that appear in E. In the experiments below, $LR_q(E)$ is just the number of base NPs from the question that appear in the summary extract. In future work, we plan to include other pairwise linguistic relationships (e.g. subject-verb relationships, verb-object relationships, pp-attachment relationships). The weight $w(E)$ is a number between 0 and 1 that is based on the retrieval rank r of the document that contains E:

$$w(E) = max(m, 1 - p \cdot r)$$

In our experiments, $m = 0.5$ and $p = 0.1$. Both values were selected manually based on the development corpus; an extensive search for the best such values was not done.

The summary extracts are sorted according to the new scoring measure and the ranked list of sentences is provided to the linguistic filters as before.

Evaluation. Results for this final variation of the QA system are shown in the bottom row of Table 1. Here we see a fairly minor increase in performance over the use of linguistic filters alone: the system answers only one more question correctly than the previous variation for the development corpus and answers five additional questions for the test corpus. The mean answer rank improves only negligibly. Sixteen of the 22 correct answers (73%) appear as the top-ranked guess for the development corpus; only 53 out of 91 correct answers (58%) appear as the top-ranked guess for the test corpus. Unfortunately, when 50-byte chunking is disabled, system performance drops precipitously, by 5% (to 20 out of 38) for the development corpus and by 13% (to 65 out of 200) for the test corpus. As noted above, this indicates that the filters are finding the answers, but more sophisticated linguistic sorting is needed to promote the best answers to the top. Through its LR_q term, the new scoring measure does provide a mechanism for allowing other linguistic relationships to influence the initial ordering of summary extracts. The current results, however, indicate that with only very weak syntactic information (i.e. base noun phrases), the new scoring measure is only marginally successful in reordering the summary extracts based on syntactic information.

As noted above, the final system (with the liberal 50-byte answer chunker) correctly answers 22 out of 38 questions for the development corpus. Of the 16 errors, the text retrieval component is responsible for five (31.2%), the text summarization component for ten (62.5%), and the linguistic filters for one (6.3%). In this analysis we consider the linguistic filters responsible for an error if they were unable to promote an available answer hypothesis to one of the top five guesses. A slightly different situation arises for the test corpus: of the 109 errors, the text retrieval component is responsible for 39 (35.8%), the text summarization component for 26 (23.9%), and the linguistic filters for 44 (40.4%). As discussed in Section 6, the heuristics that comprise the semantic type checking filter do not scale to the test corpus and are the primary reason for the larger percentage of errors attributed to the linguistic filters for that corpus.

8 Related Work and Conclusions

We have described and evaluated a series of question-answering systems, each of which incorporates a different combination of statistical and linguistic knowledge sources. We find that even very weak linguistic knowledge can offer substantial improvements over purely IR-based techniques especially when smoothly integrated with the text passage preferences computed by the IR subsystems. Although our primary goal was to investigate the use of statistical and linguistic knowledge sources, it is possible to compare our approach and our results to those for systems in the recent TREC8 QA evaluation. Scores on the TREC8 test corpus for systems participating in the QA evaluation ranged between 3 and 146 correct. Discarding the top three scores and the worst three scores, the remaining eight systems achieved between 52 and 91 correct. Using the liberal answer chunker, our final QA system equals the best of these systems (91 correct); without it, our score of 65 correct places our QA system near the middle of this group of eight.

Like the work described here, virtually all of the top-ranked TREC8 systems use a combination of IR and shallow NLP for their QA systems. IBM's AnSel system (Prager et al., 2000), for example, employs finite-state patterns as its primary shallow NLP component. These are used to recognize a fairly broad set of about 20 named entities. The IR component indexes only text passages associated with these entities. The AT&T QA system (Singhal et al., 2000), the Qanda system (Breck et al., 2000), and the SyncMatcher system (Oard et al., 2000) all employ vector-space methods from IR, named entity identifiers, and a fairly simple question type determiner. In addition, SyncMatcher uses a broad-coverage dependency parser to enforce phrase relationship constraints. Instead of the vector space model, the LASSO system (Moldovan et al., 2000) uses boolean search operators for paragraph retrieval. Recognition of answer hypotheses in their system relies on identifying named entities. Finally, the Cymphony QA system (Srihari and Li, 2000) relies heavily on named entity identification; it also employs standard IR techniques and a shallow parser.

In terms of statistical and linguistic knowledge sources employed, the primary difference between these systems and ours is our lack of an adequate named entity tagger. Incorporation of such a tagger will be a focus of future work. In addition, we believe that the retrieval and summarization components can be improved by incorporating automatic relevance feedback (Buckley, 1995) and coreference resolution. Morton (1999), for example, shows that coreference resolution improves passage retrieval for their question-answering system. We also plan to reconsider paragraph-based summaries given their coverage on the test corpus. The most critical area for improvement, however, is the linguistic filters. The semantic type filter will be greatly improved by the addition of a named entity tagger, but we believe that additional gains can be attained by augmenting named entity identification with information from WordNet. Finally, we currently make no attempt to confirm any phrase relations from the query. Without this, system performance will remain severely limited.

9 Acknowledgments

This work was supported in part by NSF Grants IRI–9624639, GER–9454149, and EIA–9703470.

References

E. Breck, J. Burger, L. Ferro, D. House, M. Light, and I. Mani. 2000. A Sys Called Qanda. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

C. Buckley, M. Mitra, J. Walz, and C. Cardie. 1998a. SMART high precision: TREC 7. In E. Voorhees, editor, *Proceedings of the Seventh Text REtrieval Conference TREC 7*, pages 285–298. NIST Special Publication 500–242.

C. Buckley, M. Mitra, J. Walz, and C. Cardie. 1998b. Using clustering and superconcepts within

SMART : TREC 6. In E. Voorhees, editor, *Proceedings of the Sixth Text REtrieval Conference TREC 6*, pages 107–124. NIST Special Publication 500–240.

C. Buckley, C. Cardie, S. Mardis, M. Mitra, D. Pierce, K. Wagstaff, and J. Walz. 1999. The Smart/Empire TIPSTER IR System. In *Proceedings, TIPSTER Text Program (Phase III)*. Morgan Kaufmann. To appear.

Chris Buckley. 1995. *Massive Query Expansion for Relevance Feedback*. Cornell University, Ph.D. Thesis, Ithaca, New York.

R. Burke, K. Hammond, and J. Kozlovsky. 1995. Knowledge-Based Information Retrieval from Semi-Structured Text. In *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, pages 19–24. AAAI Press.

R. Burke, K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg. 1997. question answering from Frequently-Asked Question Files. Technical Report TR–97–05, University of Chicago.

C. Cardie and D. Pierce. 1998. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and COLING-98*, pages 218–224, University of Montreal, Montreal, Canada. Association for Computational Linguistics.

Y. Chali, S. Matwin, and S. Szpakowicz. 1999. Query-Biased Text Summarization as a Question-Answering Technique. In *Proceedings of the AAAI Fall Symposium on Question Answering Systems*, pages 52–56. AAAI Press. AAAI TR FS–99–02.

C. Fellbaum. 1998. *WordNet: An Electronical Lexical Database*. MIT Press, Cambridge, MA.

J. Kupiec. 1993. MURAX: A Robust Linguistic approach For Question Answering Using An On-Line Encyclopedia. In *Proceedings of ACM SIGIR*, pages 181–190.

W. Lehnert. 1978. *The Process of Question Answering*. Lawrence Erlbaum Associates, Hillsdale, NJ.

I. Mani, T. Firmin, D. House, G. Klein, B. Sundheim, and L. Hirschman. 1999. The TIPSTER SUMMAC Text Summarization Evaluation. In *Ninth Annual Meeting of the EACL*, University of Bergen, Bergen, Norway.

M. Marcus, M. Marcinkiewicz, and B. Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–245.

D. Moldovan, S. Harabagiu, M. Paşca, R. Mihalcea, R. Goodrum, R. Gîrju, and V. Rus. 2000. LASSO: A Tool for Surfing the Answer Net. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

T. S. Morton. 1999. Using Coreference to Improve Passage Retrieval for Question Answering. In *Proceedings of the AAAI Fall Symposium on Question Answering Systems*, pages 72–74. AAAI Press. AAAI TR FS–99–02.

D. W. Oard, J. Wang, D. Lin, and I. Soboroff. 2000. TREC-8 Experiments at Maryland: CLIR, QA and Routing. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

J. Prager, D. Radev, E. Brown, A. Coden, and V. Samn. 2000. The Use of Predictive Annotation for Question Answering in TREC8. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

G. Salton, A. Wong, and C.S. Yang. 1975. A vector space model for information retrieval. *Communications of the ACM*, 18(11):613–620.

G. Salton, J. Allan, C. Buckley, and M. Mitra. 1994. Automatic analysis, theme generation and summarization of machine-readable texts. *Science*, 264:1421–1426, June.

Gerard Salton, editor. 1971. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ.

R. C. Schank and R. P. Abelson. 1977. *Scripts, plans, goals, and understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Amit Singhal, Chris Buckley, and Mandar Mitra. 1996. Pivoted document length normalization. In H. Frei, D. Harman, P. Schauble, and R. Wilkinson, editors, *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery.

A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, and F. Pereira. 2000. AT&T at TREC-8. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

R. Srihari and W. Li. 2000. Question Answering Supported by Information Extraction. In E. Voorhees, editor, *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST Special Publication. In press.

TREC-8. 2000. *Proceedings of the Eighth Text REtrieval Conference TREC 8*. NIST. In press.