Improving Text Retrieval Precision and Answer Accuracy in Question Answering Systems

Matthew W. Bilotti and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
{ mbilotti, ehn }@cs.cmu.edu

Abstract

Question Answering (QA) systems are often built modularly, with a text retrieval component feeding forward into an answer extraction component. Conventional wisdom suggests that, the higher the quality of the retrieval results used as input to the answer extraction module, the better the extracted answers, and hence system accuracy, will be. This turns out to be a poor assumption, because text retrieval and answer extraction are tightly coupled. Improvements in retrieval quality can be lost at the answer extraction module, which can not necessarily recognize the additional answer candidates provided by improved retrieval. Going forward, to improve accuracy on the QA task, systems will need greater coordination between text retrieval and answer extraction modules.

1 Introduction

The task of Question Answering (QA) involves taking a question phrased in natural human language and locating specific answers to that question expressed within a text collection. Regardless of system architecture, or whether the system is operating over a closed text collection or the web, most QA systems use text retrieval as a first step to narrow the search space for the answer to the question to a subset of the text collection (Hirschman and Gaizauskas, 2001). The remainder of the QA process amounts to a gradual narrowing of the search space, using successively more finely-grained filters to extract, validate and present one or more answers to the question.

Perhaps the most popular system architecture in the QA research community is the modular architecture, in most variations of which, text retrieval is represented as a separate component, isolated by a software abstraction from question analysis and answer extraction mechanisms. The widely-accepted pipelined modular architecture imposes a strict linear ordering on the system's control flow, with the analysis of the input question used as input to the text retrieval module, and the retrieved results feeding into the downstream answer extraction components.

Proponents of the modular architecture naturally view the QA task as decomposable, and to a certain extent, it is. The modules, however, can never be fully decoupled, because question analysis and answer extraction components, at least, depend on a common representation for answers and perhaps also a common set of text processing tools. This dependency is necessary to enable the answer extraction mechanism to determine whether answers exist in retrieved text, by analyzing it and comparing it against the question analysis module's answer specification. In practice, the text retrieval component does not use the common representation for scoring text; either the question analysis module or an explicit query formulation component maps it into a representation queryable by the text retrieval component.

The pipelined modular QA system architecture also carries with it an assumption about the compositionality of the components. It is easy to observe that errors cascade as the QA process moves through downstream modules, and this leads to the intuition that maximizing performance of individual modules minimizes the error at each stage of the pipeline, which, in turn, should maximize overall end-to-end system accuracy.

It is a good idea to pause to question what this intuition is telling us. Is end-to-end QA system performance really a linear function of individual

[ARG0 [PERSON *John*]] [TARGET *loves*] [ARG1 [PERSON *Mary*]]

Figure 1: Example OpenEphyra semantic representation for the sentence, *John loves Mary*. Note that *John* is identified as the ARG0, the agent, or doer, of the *love* action. *Mary* is identified as the ARG1, the patient, or to whom the *love* action is being done. Both *John* and *Mary* are also identified as PERSON named entity types.

components? Is component performance really additive? This paper argues that the answer is no, not in general, and offers the counterexample of a high-precision text retrieval system that can check constraints against the common representation at retrieval time, which is integrated into a publicly-available pipelined modular QA system that is otherwise unchanged.

Ignoring the dependency between the answer extraction mechanism and the text retrieval component creates a problem. The answer extraction module is not able to handle the more sophisticated types of matches provided by the improved text retrieval module, and so it ignores them, leaving end-to-end system performance largely unchanged. The lesson learned is that a module improved in isolation does not necessarily provide an improvement in end-to-end system accuracy, and the paper concludes with recommendations for further research in bringing text retrieval and answer extraction closer together.

2 Improving Text Retrieval in Isolation

This section documents an attempt to improve the performance of a QA system by substituting its existing text retrieval component with for high-precision retrieval system capable of checking linguistic and semantic constraints at retrieval time.

2.1 The OpenEphyra QA System

OpenEphyra is the freely-available, open-source version of the Ephyra[1] QA system (Schlaefer et al., 2006; Schlaefer et al., 2007). OpenEphyra is a pipelined modular QA system having four stages: question analysis, query generation, search and answer extraction and selection. OpenEphyra also includes support for answer projection, or the use of the web to find answers to the question, which are then used to find supporting text in the corpus. Answer projection support was disabled for the purposes of this paper.

[1]See: `http://www.ephyra.info`

The common representation in OpenEphyra is a verb predicate-argument structure, augmented with named entity types, in which verb arguments are labeled with semantic roles in the style of PropBank (Kingsbury et al., 2002). This feature requires the separate download[2] of a semantic parser called ASSERT (Pradhan et al., 2004), which was trained on PropBank. See Figure 1 for an example representation for the sentence, *John loves Mary*.

OpenEphyra comes packaged with standard baseline methods for answer extraction and selection. For example, it extracts answers from retrieved text based on named entity instances matching the expected answer type as determined by the question analysis module. It can also look for predicate-argument structures that match the question structure, and can extract the argument corresponding to the argument in the question representing the interrogative phrase. OpenEphyra's default answer selection algorithm filters out answers containing question keyterms, merges subsets, and combines scores of duplicate answers.

2.2 Test Collection

The corpus used in this experiment is the AQUAINT corpus (Graff, 2002), the standard corpus for the TREC[3] QA evaluations held in 2002 through 2005. The corpus was prepared using MXTerminator (Reynar and Ratnaparkhi, 1997) for sentence segmentation, BBN Identifinder (Bikel et al., 1999) for named entity recognition, as well as the aforementioned ASSERT for identification of verb predicate-argument structures and PropBank-style semantic role labeling of the arguments.

The test collection consists of 109 questions from the QA track at TREC 2002 with extensive document-level relevance judgments (Bilotti et al., 2004; Lin and Katz, 2006) over the AQUAINT corpus. A set of sentence-level judgments was pre-

[2]See: `http://www.cemantix.org`

[3]Text REtrieval Conferences organized by the U.S. National Institute of Standards and Technology

Existing query	`#combine[sentence]( #any:person first person reach south pole )`
Top-ranked result	*Dufek became the first person to land an airplane at the South Pole.*
Second-ranked result	*He reached the North Pole in 1991.*
High-precision query	`#combine[sentence]( #max( #combine[target]( scored #max( #combine[./arg1]( #any:person )) #max( #combine[./arg2]( #max( #combine[target]( reach #max( #combine[./arg1]( south pole ))))))))`
Top-ranked result (relevant)	[ARG1 *Norwegian explorer* [PERSON *Roald Admundsen*]] [TARGET *becomes*] [ARG2 [ARG0 *first man*] *to* [TARGET *reach*] [ARG1 [LOCATION *South Pole*]]]

Figure 2: Retrieval comparison between OpenEphrya's existing text retrieval component, and the high-precision version it was a replaced with, for question 1475, *Who was the first person to reach the South Pole?* Note that the top two results retrieved by the existing text retrieval component are not relevant, and the top result from the high-precision component is relevant. The existing component does retrieve this answer-bearing sentence, but ranks it third.

pared by manually determining whether each sentence matching the TREC-provided answer pattern for a given question was *answer-bearing* according to the definition that an answer-bearing sentence completely contains and supports the answer to the question, without requiring inference or aggregation outside of that sentence. Questions without any answer-bearing sentences were removed from the test collection, leaving 91 questions.

Questions were manually reformulated so that they contain predicates. For example, question 1432, *Where is Devil's Tower?* was changed to *Where is Devil's Tower located?*, because ASSERT does not cover verbs, including *be* and *have*, that do not occur in its training data. Hand-corrected ASSERT parses for each question were were cached in the question analysis module. Reformulated questions are used as input to both the existing and high-precision text retrieval modules, to avoid advantaging one system over the other.

2.3 High-Precision Text Retrieval

OpenEphyra's existing text retrieval module was replaced with a high-precision text retrieval system based on a locally-modified version of the Indri (Strohman et al., 2005) search engine, a part of the open-source Lemur toolkit[4]. While the existing version of the text retrieval component supports querying on keyterms, phrases and placeholders for named entity types, the high-precision version also supports retrieval-time constraint-checking against the semantic representation based on verb predicate-argument structures, PropBank-style semantic role labels, and named entity recognition.

To make use of this expanded text retrieval capability, OpenEphyra's query formulation module was changed to source pre-prepared Indri queries that encode using structured query operators the predicate-argument and named entity constraints that match the answer-bearing sentences for each question. If questions have multiple queries associated with them, each query is evaluated individually, with the resulting ranked lists fused by Round Robin (Voorhees et al., 1994). Round Robin, which merges ranked lists by taking the top-ranked element from each list in order followed by lower-ranking elements, was chosen because Indri, the underlying retrieval engine, gives different queries scores that are not comparable in general, making it difficult to choose a fusion method that uses retrieval engine score as a feature.

Figure 2 shows a comparison of querying and retrieval behavior between OpenEphyra's existing text retrieval module and the high-precision version with which it is being replaced for question 1475, *Who was the first person to reach the South Pole?* The bottom of the figure shows an answer-bearing sentence with the correct answer, *Roald Admundsen*. The predicate-argument structure, se-

[4]See: `http://www.lemurproject.org`

mantic role labels and named entities are shown.

The high-precision text retrieval module supports storing of extents representing sentences, target verbs and arguments and named entity types as fields in the index. At query time, constraints on these fields can be checked using structured query operators. The queries in Figure 2 are shown in Indri syntax. Both queries begin with `#combine[sentence]`, which instructs Indri to score and rank sentence extents, rather than entire documents. The query for the existing text retrieval component contains keyterms as well an `#any:`*type* operator that matches instances of the expected answer type, which in this case is *person*. The high-precision query encodes a verb predicate-argument structure. The nested `#combine[target]` operator scores a sentence by the predicate-argument structures it contains. The `#combine[./`*role*`]` operators are used to indicate constraints on specific argument roles. The dot-slash syntax tells Indri that the argument extents are related to but not enclosed by the target extent. Throughout, the `#max` operator is used to select the best matching extent in the event that more than one satisfy the constraints.

Figure 3 compares average precision at the top twenty ranks over the entire question set between OpenEphyra's existing text retrieval module and the high-precision text retrieval module, showing that the latter performs better.

2.4 Results

To determine what effect improving text retrieval quality has on the end-to-end QA system, it suffices to run the system on the entire test collection,

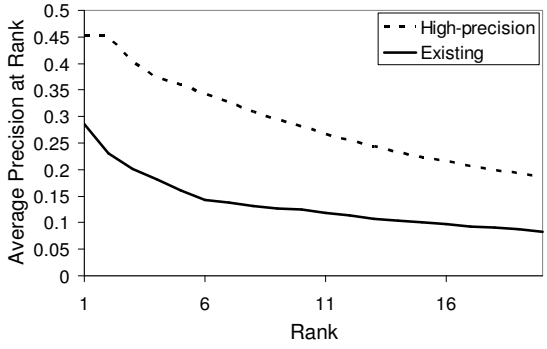

Figure 3: Comparison of average precision at top twenty ranks between OpenEphyra's existing text retrieval module, and the high-precision version that took its place.

replace the text retrieval component with the high-precision version while holding the other modules constant, and repeat the test run. Table 1 summarizes the MAP, average end-to-end system accuracy (whether the top-ranked returned answer is correct), and the mean reciprocal rank (MRR) of the correct answer (one over the rank at which the correct answer is returned). If the correct answer to a question is returned beyond rank twenty, the reciprocal rank for that question is considered to be zero.

Table 1: Summary of end-to-end QA system accuracy and MRR when the existing text retrieval module is replaced with a high-precision version

Retrieval	MAP	Accuracy	MRR
Existing	0.3234	0.1099	0.2080
High-precision	0.5487	0.1319	0.2020

Table 1 shows that, despite the improvement in average precision, the end-to-end system did not realize a significant improvement in accuracy or MRR. Viewed in the aggregate, the results are discouraging, because it seems that the performance gains realized after the text retrieval stage of the pipeline are lost in downstream answer extraction components.

Figure 4 compares OpenEphyra both before and after the integration of the high-precision text retrieval component on the basis of average precision and answer MRR. The horizontal axis plots the difference in average precision; a value of positive one indicates that the high-precision version of the module was perfect, ranking all answer-bearing sentences at the top of the ranked list, and that the existing version retrieved no relevant text at all. Negative one indicates the reverse. The vertical axis plots the difference in answer MRR. As before, positive one indicates that the high-precision component led the system to rank the correct answer first, and the existing component did not, and negative one indicates the reverse. The zero point on each axis is where the high-precision and existing text retrieval components performed equally well.

The expectation is that there will be a positive correlation between average precision and answer MRR; when the retrieval component provides higher quality results, the job of the answer extraction module should be easier. This is illustrated in the bottom portion of Figure 4, which was cre-

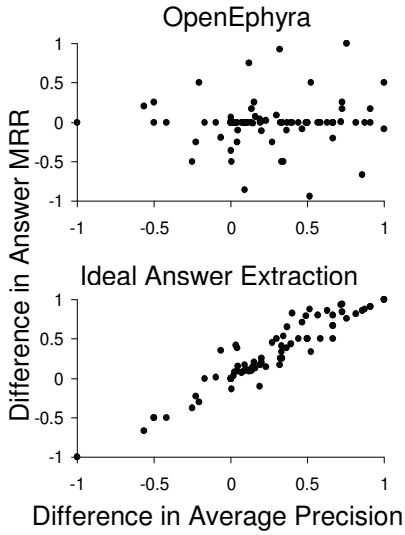

Figure 4: Scatter plot comparing the difference in average precision between the high-precision retrieval component and the existing retrieval component on the horizontal axis, to the difference in answer MRR on the vertical axis. Ideally, there would be a high correlation between the two; as average precision improves, so should answer MRR.

ated by assuming that the answer extraction module could successfully extract answers without error from all answer-bearing sentences returned by the text retrieval component.

Interestingly, actual extraction performance, shown in the top portion of Figure 4, bears little resemblance to the ideal. Note the large concentration of data points along the line representing zero difference in answer MRR. This indicates that, regardless of improvement in average precision of the results coming out of the retrieval module, the downstream answer extraction performance remains the same as it was when the existing text retrieval component was in use. This occurs because the answer extraction module does not know how to extract answers from some of the types of answer-bearing sentences retrieved by the high-precision version of the retrieval module and not by the existing version.

There are several data points in the top right-hand quadrant of the top half of Figure 4, indicating that for some questions, answer extraction was able to improve as average precision improved. This is likely due to better rankings for types of answer-bearing sentences that answer extraction already knows how to handle. Data points occurring in the lower right-hand portion of the graph indicate depressed answer extraction performance as average precision is increasing. This phenomenon can be explained by the higher-precision text retrieval module ranking answer-bearing sentences that answer extraction can not handle ahead of those that it can handle.

3 Failure Analysis

The results presented in the previous section confirm that an improvement made to the text retrieval component, in isolation, without a corresponding improvement to the downstream answer extraction modules, can fail to translate into a corresponding improvement in end-to-end QA system accuracy. The increased average precision in the retrieved results is coming in the form of answer-bearing sentences of types that the answer extraction machinery does not know how to handle. To address this gap in answer extraction coverage, it is first necessary to examine examples of the types of errors made by the OpenEphyra answer extraction module, summarized in Table 2.

Question 1497, *What was the original name before "The Star Spangled Banner"?* is an example of a question for which OpenEphyra's answer extraction machinery failed outright. An answer-bearing sentence was retrieved, however, containing the answer inside a quoted phrase: *His poem was titled "Defense of Fort M'Henry" and by November 1814 had been published as "The Star-Spangled Banner"*. The expected answer type of this question does not match a commonly-used named entity type, so OpenEphrya's named entity-based answer extractor found no candidates in this sentence. Predicate-argument structure-based answer extraction fails as well because the old and new names do not appear within the same structure. Because OpenEphyra does not include support for positing quoted phrases as answer candidates, no answer to this question can be found despite the fact that an answer-bearing sentence was retrieved.

Question 1417, *Who was the first person to run the mile in less than four minutes?* is an example of a question for which average precision improved greatly, by 0.7208, but for which extraction quality remained the same. The existing text retrieval module ranks 14 sentences ahead of the first answer-bearing sentence, but only one contains a named entity of type person, so despite the improvement in retrieval quality, the correct answer

Table 2: Summary of end-to-end QA system results on the question set

Result Type	Count
Extraction failure	42
Retrieval better, extraction same	20
Retrieval better, extraction worse	13
Retrieval better, extraction better	10
Retrieval worse, extraction better	3
Retrieval worse, extraction worse	3
Total	91

moves up only one rank in the system output.

For ten questions, extraction performance does improve as average precision improves. Question 1409, *Which vintage rock and roll singer was known as "The Killer"?* For each of these questions, OpenEphyra's existing text retrieval module could not rank an answer-bearing sentence highly or retrieve one at all. Adding the high-precision version of the text retrieval component solved this problem. In each case, named entity-based answer extraction was able extract the correct answer. These eleven questions range over a variety of answer types, and have little in common except for the fact that there are relatively few answer-bearing sentences in the corpus, and large numbers of documents matched by a bag-of-words query formulated using the keyterms from the question.

There are three questions for which extraction performance degrades as retrieval performance degrades. Question 1463, *What is the North Korean national anthem?* is an example. In this case, there is only one relevant sentence, and, owing to an annotation error, it has a predicate-argument structure that is very generic, having *North Korea* as the only argument: *Some of the North Korean coaches broke into tears as the North's anthem, the Patriotic Song, played.* The high-precision retrieval component retrieved a large number of sentences matching the that predicate-argument structure, but ranked the one answer-bearing sentence very low.

Some questions actually worsened in terms of the reciprocal rank of the correct answer when average precision improved. An example is question 1504, *Where is the Salton Sea?* The high-precision text retrieval module ranked answer-bearing sentences such as *The combination could go a long way to removing much of the pesticides, fertilizers, raw sewage carried by the river into the Salton Sea, the largest lake in California*, but a failure of the named entity recognition tool did not identify *California* as an instance of the expected answer type, and therefore it was ignored. Sentences describing other seas near other locations provided answers such as *Central Asia*, *Russia*, *Turkey* and *Ukraine* that were ranked ahead of *California*, which was eventually extracted from another answer-bearing sentence.

And finally, for some questions, high-precision retrieval was more of a hindrance than a help, retrieving more noise than answer-bearing sentences. A question for which this is true is question 1470, *When did president Herbert Hoover die?* The high-precision text retrieval module uses a predicate-argument structure to match the target verb *die*, theme *Hoover* and a *date* instance occurring in a temporal adjunct. Interestingly, the text collection contains a great deal of *die* structures that match partially, including those referring to deaths of presidents of other nations, and those referring to the death of J. Edgar Hoover, who was not a U.S. president but the first director of the U.S. Federal Bureau of Investigation (FBI). False positives such as these serve to push the true answer down on the ranked list of answers coming out of the QA system.

4 Improving Answer Extraction

The answer extraction and selection algorithms packaged with OpenEphyra are widely-accepted baselines, but are not sophisticated enough to extract answer candidates from the additional answer-bearing text retrieved by the high-precision text retrieval module, which can check linguistic and semantic constraints at query time.

The named-entity answer extraction method selects any candidate answer that is an instance of the expected answer type, so long as it co-occur with query terms. Consider question 1467, *What*

year did South Dakota become a state? Given that the corpus consists of newswire text reporting on current events, years that are contemporary to the corpus often co-occur with the question focus, as in the following sentence, *Monaghan also seized about $87,000 from a Santee account in South Dakota in 1997*. Of the top twenty answers returned for this question, all but four are contemporary to the corpus or in the future. Minimal sanity-checking on candidate answers could save the system the embarrassment of returning a date in the future as the answer. Going one step further would involve using external sources to determine that *1997* is too recent to be the year a state was admitted to the union.

OpenEphyra's predicate-argument structure-based answer extraction algorithm can avoid some of these noisy answers by comparing some constraints from the question against the retrieved text and only extracting answers if the constraints are satisfied. Consider question 1493, *When was Davy Crockett born?* One relevant sentence says *Crockett was born Aug. 17, 1786, in what is now eastern Tennessee, and moved to Lawrenceburg in 1817*. The SRL answer extraction algorithm extracts *Aug. 17, 1786* because it is located in an argument labeled *argm-tmp* with respect to the verb, and ignores the other date in the sentence, *1817*. The named entity-based answer extraction approach proposes both dates as answer candidates, but the redundancy-based answer selection prefers *1786*.

The predicate-argument structure-based answer extraction algorithm is limited because it only extracts arguments from text that shares the structure as the question. The high-precision text retrieval approach is actually able to retrieve additional answer-bearing sentences with different predicate-argument structures from the question, but answer extraction is not able to make use of it. Consider the sentence, *At the time of his 100 point game with the Philadelphia Warriors in 1962, Chamberlain was renting an apartment in New York*. Though this sentence answers the question *What year did Wilt Chamberlain score 100 points?*, its predicate-argument structure is different from that of the question, and predicate-argument structure-based answer extraction will ignore this result because it does not contain a *score* verb.

In addition to answer extraction, end-to-end performance could be improved by focusing on answer selection. OpenEphyra does not include support for sanity-checking the answers it returns, and its default answer selection mechanism is redundancy-based. As a result, nonsensical answers are occasionally retrieved, such as *moon* for question 1474, *What is the lowest point on Earth?* Sophisticated approaches, however, do exist for answer validation and justification, including use of resources such as gazetteers and ontologies (Buscaldi and Rosso, 2006), Wikipedia (Xu et al., 2002), the Web (Magnini et al., 2002), and combinations of the above (Ko et al., 2007).

5 Conclusions

This paper set out to challenge the assumption of compositionality in pipelined modular QA systems that suggests that an improvement in an individual module should lead to an improvement in the overall end-to-end system performance. An attempt was made to validate the assumption by showing an improvement in the end-to-end system accuracy of an off-the-shelf QA system by substituting its existing text retrieval component for a high-precision retrieval component capable of checking linguistic and semantic constraints at query time. End-to-end system accuracy remained roughly unchanged because the downstream answer extraction components were not able to extract answers from the types of the answer-bearing sentences returned by the improved retrieval module.

The reality of QA systems is that there is a high level of coupling between the different system components. Ideally, text retrieval should have an understanding of the kinds of results that answer extraction is able to utilize to extract answers, and should not offer text beyond the capabilities of the downstream modules. Similarly, question analysis and answer extraction should be agreeing on a common representation for what constitutes an answer to the question so that answer extraction can use that information to locate answers in retrieved text. When a retrieval module is available that is capable of making use of the semantic representation of the answer, it should do so, but answer extraction needs to know what it can assume about incoming results so that it does not have to re-check constraints already guaranteed to hold.

The coupling between text retrieval and answer extraction is important for a QA system to perform well. Improving the quality of text retrieval is essential because once the likely location of

the answer is narrowed down to a subset of the text collection, anything not retrieved text can not be searched for answers in downstream modules. Equally important is the role of answer extraction. Even the most relevant retrieved text is useless to a QA system unless answers can be extracted from it. End-to-end QA system performance can not be improved by improving text retrieval quality in isolation. Improvements in answer extraction must keep pace with progress on text retrieval techniques to reduce errors resulting from a mismatch in capabilities. Going forward, research on the linguistic and semantic constraint-checking capabilities of text retrieval systems to support the QA task can drive research in answer extraction techniques, and in QA systems in general.

References

Bikel, D., R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3):211–231.

Bilotti, M., B. Katz, and J. Lin. 2004. What works better for question answering: Stemming or morphological query expansion? In *Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR 2004*.

Bilotti, M., P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Buscaldi, D. and P. Rosso. 2006. Mining knowledge from wikipedia for the question answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Cui, H., R. Sun, K. Li, M. Kan, and T. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Graff, D. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium (LDC). Cat. No. LDC2002T31.

Hirschman, L. and R. Gaizauskas. 2001. Natural language question answering: The view from here. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter.

Kingsbury, P., M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*.

Ko, J., L. Si, and E. Nyberg. 2007. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Lin, J. and B. Katz. 2006. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, 57(7):851–861.

Magnini, B., M. Negri, R. Pervete, and H. Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIIIo Convegno AI*IA*.

Narayanan, S. and S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*.

Pradhan, S., W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*.

Reynar, J. and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

Schlaefer, N., P. Gieselmann, and G. Sautter. 2006. The ephyra qa system at trec 2006. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC)*.

Schlaefer, N., J. Ko, J. Betteridge, G. Sautter, M. Pathak, and E. Nyberg. 2007. Semantic extensions of the ephyra qa system for trec 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC)*.

Strohman, T., D. Metzler, H. Turtle, and W. B. Croft. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*.

Sun, R., J. Jiang, Y. Tan, H. Cui, T. Chua, and M. Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC-14)*.

Voorhees, E., N. Gupta, and B. Johnson-Laird. 1994. The collection fusion problem. In *Proc. of TREC-3*.

Xu, J., A. Licuanan, J. May, S. Miller, and R. Weischedel. 2002. Trec 2002 qa at bbn: Answer selection and confidence estimation. In *Proceedings of the Text REtrieval Conference*.