In Proceedings of the 26th Annual International ACM SIGIR Conference
on Research and Development in Information Retrieval (SIGIR 2003),
July 2003, Toronto, Canada (Best Student Paper)

Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton
MIT Artificial Intelligence Laboratory
Cambridge, Massachusetts, USA
{stefie10,boris,jimmylin,adfernan,gremio}@ai.mit.edu

ABSTRACT

Passage retrieval is an important component common to many question answering systems. Because most evaluations of question answering systems focus on end-to-end performance, comparison of common components becomes difficult. To address this shortcoming, we present a quantitative evaluation of various passage retrieval algorithms for question answering, implemented in a framework called Pauchok. We present three important findings: Boolean querying schemes perform well in the question answering task. The performance differences between various passage retrieval algorithms vary with the choice of document retriever, which suggests significant interactions between document retrieval and passage retrieval. The best algorithms in our evaluation employ density-based measures for scoring query terms. Our results reveal future directions for passage retrieval and question answering.

Categories and Subject Descriptors

H.3.4 [**Information Systems**]: Information Storage and Retrieval, Systems and Software

General Terms

Measurement, performance, standardization, design

Keywords

Question answering, passage retrieval

1. INTRODUCTION

Factoid question answering systems seek to provide concise, succinct answers to natural language questions such as “When did Hawaii become a state?” The aim is to perform fine-grained, targeted information retrieval: instead of retrieving a list of potentially relevant documents, question answering systems attempt to extract only the information requested by the user.

Over the past few years, the question answering tracks at the Text Retrieval Conferences (TREC) [25, 23, 24] have brought formal and rigorous evaluation methodologies to bear on the question answering task. Although the importance of these evaluations cannot be denied, they measure only the end-to-end performance of complex systems that typically involve a combination of information retrieval, information extraction, and natural language processing technologies. Without further ablation studies (e.g., [16, 13]), it is difficult to untangle the performance contributions of the various components. In this study, we present a quantitative evaluation of various passage retrieval algorithms and explore their relationship to document retrieval.

Functionally, most current question answering systems can be decomposed into four components: question analysis, document retrieval, passage retrieval, and answer extraction (cf. [7, 23]). The question analysis component classifies natural language questions by the expected type of the answer, e.g., the expected answer type of “Where was Kennedy assassinated?” is *location*. Typically, queries generated by this component are used by the document retriever to find a set of potentially relevant documents from the corpus. From these documents, the passage retrieval component usually selects a handful of paragraph-sized fragments. Most often, passage retrieval algorithms perform a density-based weighting of query terms, i.e., they favor query terms that appear close together. Finally, the answer extraction component searches the passages for the final answers. A variety of answer extraction techniques ranging in linguistic sophistication have been implemented, from simple answer-type matching with named-entity extractors [21] to complex abductive inferencing [6].

For this study, we focused on passage retrieval algorithms for question answering because, compared to document retrieval, they have not been studied in as much detail. Many passage retrieval techniques have been described in the context of improving document retrieval performance (e.g., [19, 1]), but we are not aware of any attempts to systematically study the performance of passage retrieval for question answering. Passages are an important intermediary between full documents and exact answers, and almost all question answering systems implement some technique for extracting paragraph-sized chunks of text from a large corpus. Furthermore, passages themselves form a very natural unit of response for question answering systems; Lin *et al.* [14] showed that users prefer passages over exact phrase answers in a real-world setting because paragraph-sized chunks provide context.

SIGIR'03, July 28–August 1, 2003, Toronto, Canada.
Copyright 2003 ACM 1-58113-646-3/03/0007 ...$5.00.

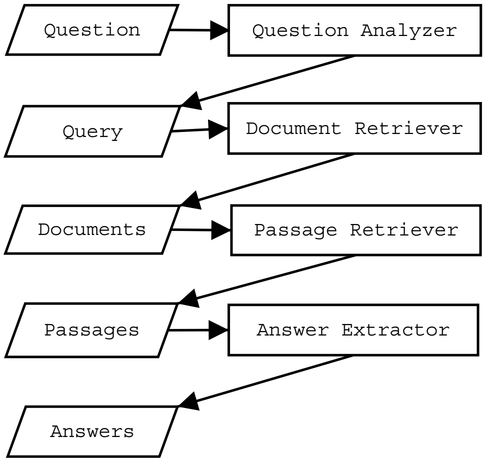

Figure 1: Data Flow in Pauchok.

This work quantitatively studies the effects of different passage retrieval algorithms on a question answering task. However, since the quality of the passage retriever depends in part on the quality of the document retriever, our study also seeks to untangle the interactions between document retrieval and passage retrieval.

2. EXPERIMENTAL DESIGN

In order to quantitatively compare the performance of different passage retrieval algorithms, we have developed a modular testbed called Pauchok. Our infrastructure consists of four main types of modules, along the lines of typical question answering systems (see Figure 1):

- Question analysis modules convert natural language questions into queries for the document retriever. This process can range in sophistication from simply returning the user's question as the query to employing sophisticated question analysis to generate complex, structured queries. Often, this module also detects the expected answer type of a question, e.g., the expected answer type of "When was Albert Einstein born?" is *date*. This information helps guide the answer extraction process.

- Document retrieval modules return ranked lists of potentially relevant documents from the corpus. This process reduces the corpus to a manageable set of documents for additional processing.

- Passage retrieval modules process sets of documents and return ranked lists of passages scored with respect to query terms.

- Answer extraction modules search passages for the final answer to the user's natural language question. Typically, named-entity recognition technology is used to find candidate answers that match the question's expected answer type.

Our study is essentially a matrix experiment for a question answering task that involves three document retrievers and eight passage retrievers.

2.1 Document Retrievers

Currently, we have implemented two document retrievers and an oracle in the Pauchok framework. One is a wrapper around documents that NIST retrieved using the PRISE system. [1] These documents were provided for TREC participants who did not wish to perform document retrieval themselves. The second document retriever is a wrapper around Lucene, a freely available open-source IR engine.[2] Lucene supports a boolean query language, although it performs ranked retrieval using a standard *tf.idf* model. The oracle is guaranteed to return relevant documents, i.e., every document returned has the answer somewhere within it. In Pauchok, the oracle is treated like any other document retriever; it is hardwired to return the NIST-supplied list of known relevant documents.

These document retrievers are good exemplars of the spectrum of document retrieval. PRISE is representative of the state of the art in information retrieval, incorporating many modern advances in query and term weighting, i.e., *bm25* [18, 17]. Lucene is a good example of boolean keyword search engines used by many TREC systems. Its ranked-retrieval functionality is only invoked to sort documents that satisfy the boolean query. As a result, Lucene suffers from well-known problems that plague boolean systems, e.g., very little control over the size of the hit list [4]. We implemented the oracle document retriever to study passage retrieval algorithms in isolation and to assess their performance with perfect document retrieval.

2.2 Passage Retrieval Algorithms

For our study, we implemented eight different passage retrieval algorithms in the Pauchok framework. Most of the algorithms were chosen from top-performing TREC-10 systems that had well-described passage retrieval algorithms.

We did not include passage retrieval algorithms from two notable TREC-10 systems. The top performing TREC-10 system, InsightSoft [20], cuts the retrieved documents into passages around query terms, returning all passages from all retrieved documents. The use of human-generated indicative patterns makes answer extraction on such large amounts of text viable. The second best system, LCC [5], retrieves passages that contain keywords from the question based on the results of question analysis. Their passage retrieval algorithm requires the expected answer type of the question or a bridging inference between the expected answer type and the question. However, neither the answer type ontology nor the bridging inference mechanisms were described well enough for us to implement.

The following subsections provide an overview of the surveyed algorithms.

2.2.1 *MITRE*

The word overlap algorithm presented by Light *et al.* [13] simply counts the number of terms a passage has in common with the query, where each sentence is treated as a separate passage. This algorithm represents the simplest reasonable passage retrieval technique and serves as a good baseline for comparison. Although the version described by Light *et al.* makes use of stemming, we implemented both a stemming and non-stemming version of the algorithm.

[1] `www.itl.nist.gov/iad/894.02/works/papers/zp2/zp2.html`

[2] `jakarta.apache.org/lucene/docs/index.html`

2.2.2 bm25

The well-known Okapi *bm25* weighting scheme [18, 17] represents the state of the art in document retrieval. We implemented a simple passage retrieval algorithm based on a sliding window scored with *bm25* to serve as another baseline for comparison.

2.2.3 MultiText

The MultiText algorithm [2, 3] is a density-based passage retrieval algorithm that favors short passages containing many terms with high *idf* values. Each passage window in the algorithm starts and ends with a query term, and its score is based on the number of query terms in the passage as well as the window size. Once the highest scoring passage has been identified, our implementation creates a new window of the required length around the center point of the original passage.

Due to the structure of their index, Waterloo's implementation of the MultiText algorithm uses a variant of *idf* for the term weights. However, our implementation uses the standard definition of *idf*.

2.2.4 IBM

IBM's passage retrieval algorithm [10, 9] computes a series of distance measures for the passage. The "matching words measure" sums the *idf* values of words that appear in both the query and the passage. The "thesaurus match measure" sums the *idf* values of words in the query whose WordNet synonyms appear in the passage. The "mis-match words measure" sums the *idf* values of words that appear in the query and not in the passage. The "dispersion measure" counts the number of words in the passage between matching query terms, and the "cluster words measure" counts the number of words that occur adjacently in both the question and the passage. These various measures are linearly combined to give the final score for a passage.

2.2.5 SiteQ

SiteQ's passage retrieval algorithm [12] computes the score of an n-sentence passage by summing the weights of the individual sentences. Sentences are weighted based on query term density. This algorithm weights query terms based on their part of speech; however the version implemented in Pauchok uses the *idf* weight instead. Our experiments show an optimal passage length of three sentences.

2.2.6 Alicante

Alicante's passage retrieval algorithm [22, 15] computes the non-length normalized cosine similarity between query terms and the passage. It takes into account the number of appearances of a term in the passage and in the query, along with their *idf* values. The Alicante system reported an optimal passage size of twenty sentences, but our experiments show highest performance using a six sentence window.

2.2.7 ISI

ISI's passage retrieval algorithm [8] ranks sentences based on their similarity to the question by weighing various features: exact match of proper names, match of query terms, and match of stemmed words. Their passage scoring function includes a term whose sole purpose is to offset scoring performed by the answer extractor; we did not implement this part of the algorithm.

2.2.8 Voting

We designed a new passage retrieval algorithm by combining the results from our implemented collection of algorithms. We implemented a simple voting scheme that scored each passage based on its initial rank and also based on the number of answers the other algorithms returned from the same document. More precisely, given the results from various passage retrieval algorithms, the score for each passage is calculated as follows:

$$\begin{aligned}
A &= \text{number of algorithms} \\
R &= \text{number of passages returned} \\
\mathit{docids} &= A \times R \text{ matrix of document ids} \\
&\quad\ \text{returned by each algorithm} \\
\mathit{docscore}(\mathit{doc}) &= \sum_{a=1}^{A}\sum_{r=1}^{R}\begin{cases} 1/r & \text{if } \mathit{docids}[a,r] = \mathit{doc} \\ 0 & \text{otherwise} \end{cases} \\
\mathit{score}(a,r) &= \frac{1}{r} + \frac{1}{2}\mathit{docscore}(\mathit{docids}[a,r])
\end{aligned}$$

We ran this voting algorithm using IBM, ISI, and SiteQ, the best performing algorithms under the PRISE IR engine.

2.3 Procedure

Our study was a matrix experiment involving the document retrievers and passage retrievers described above. The TREC-9 data was used for training purposes, and the data from TREC-10 was used in our evaluation.

For passage retrieval algorithms that required parameter tuning (Alicante, IBM, ISI, and SiteQ), we used the TREC-9 test set on Lucene to explore the parameter space. Starting with parameters reported by the authors of the passage retrieval algorithms, we automatically refined each parameter in a hill-climbing fashion. The step size was chosen by initial ad-hoc experimentation. We informally verified that passage ranking performance remained relatively constant with respect to perturbations in parameters. Overall, the performance of the tuned algorithms was comparable to the results reported by the original authors.

As mentioned previously, our experiments involved three different document retrieval modules. For the PRISE document retriever, NIST presented the entire question verbatim as the query. For Lucene, Pauchok composed a conjunctive boolean query after removing stopwords from the question. The oracle document retriever ignored the input query terms (taking into account only the question number) because it only returned known relevant documents.

Pauchok ran each passage retrieval algorithm on the first two hundred documents returned by the document retriever, although the document retriever sometimes returned fewer documents. Each algorithm extracted the best passage from every document, and returned up to twenty passages. Our implementations ignored the original document rank and document score. For each question, algorithms returned up to twenty passages of 1000 bytes each. In an end-to-end question answering system, an answer extraction algorithm would then select the final answers from the passages; in TREC-10, systems returned up to five 50 byte answers.

Due to the differences in each algorithm, passage lengths varied dramatically. To normalize for these variations, Pauchok expanded or contracted the initial passage returned by the algorithm to fit within the space allotted. If a passage re-

	Strict					Lenient				
	Lucene		**PRISE**		**TREC**	**Lucene**		**PRISE**		**TREC**
Algorithm	MRR	% Inc.	MRR	% Inc.	% Inc.	MRR	% Inc.	MRR	% Inc.	% Inc.
IBM	0.326	49.20%	0.331	39.60%	44.3%	0.426	39.60%	0.421	30.80%	43.1%
ISI	0.329	48.80%	0.287	41.80%	41.7%	0.413	40.20%	0.396	32.20%	39.8%
SiteQ	0.323	48.00%	0.358	40.40%	56.1%	0.421	40.20%	0.435	32.60%	52.8%
MultiText	0.354	46.40%	0.325	41.60%	43.1%	0.428	38.60%	0.398	34.80%	40.7%
Alicante	0.296	50.00%	0.321	42.60%	60.4%	0.380	41.80%	0.391	35.20%	59.6%
bm25	0.312	48.80%	0.252	46.00%	n/a	0.410	40.80%	0.345	38.00%	n/a
stemmed MITRE	0.250	52.60%	0.242	58.60%	n/a	0.338	44.20%	0.312	39.20%	n/a
MITRE	0.271	49.40%	0.189	52.00%	n/a	0.372	42.20%	0.265	42.00%	n/a
Averages	0.309	49.15%	0.297	45.33%	n/a	0.399	40.95%	0.370	35.60%	n/a
Voting with IBM, ISI, SiteQ	0.350	39.80%	0.352	39.00%	n/a	0.410	31.00%	0.430	30.00%	n/a

Table 1: Performance on TREC-10 using both Lucene and PRISE, ordered by the lenient percentage incorrect under PRISE. Mean reciprocal rank (MRR) and percentage incorrect (% Inc.) are shown for both the strict and lenient scoring conditions. The performance of the associated TREC-10 systems is also provided.

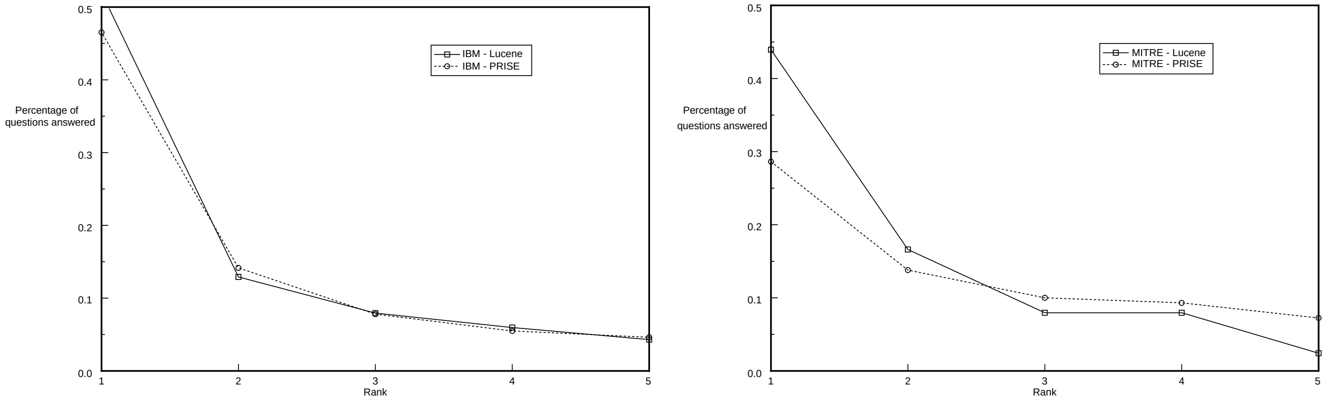

Figure 2: Graphs of performance IBM and MITRE (using both Lucene and PRISE).

trieval algorithm returned an answer shorter than the limit, Pauchok expanded the passage by adding words surrounding the passage from the document on both ends. Similarly, if an algorithm returned a passage that was too long, Pauchok trimmed words from both ends accordingly.

All results, in the form of $[question, docid]$ pairs were automatically scored using NIST-supplied scripts designed to simulate human judgments with regular expression patterns. We modified the scoring scripts to provide both strict and lenient scores. For strict scoring, an answer was correct if it matched one of the answer patterns and its associated document was listed as one of the known relevant documents (also supplied by NIST). A correct answer only needed to match the pattern for lenient scoring, regardless of the supporting document. We collected two standard performance metrics: mean reciprocal rank (MRR) and percentage of questions with no correct answers. MRR was measured over all twenty response passages, as opposed to the usual five in formal TREC evaluations.

3. RESULTS

Table 1 shows the overall performance of the passage retrieval algorithms with Lucene and PRISE, under strict and lenient conditions. ANOVA (over all runs except for voting) revealed that the performance differences for the PRISE set of results were statistically significant under both strict and lenient scoring (lenient: $F(7, 3992) = 3.25$, $p = 0.001$; strict: $F(7, 3992) = 3.71$, $p = 0.0005$). Intuitively, this means that chance alone could not account for the performance differences among the algorithms. However, the differences in the performance of passage retrieval algorithms with Lucene were not significant under both strict and lenient scoring, as demonstrated by ANOVA (over all runs except for voting): (lenient: $F(7, 3696) = 0.71$, *ns*; strict $F(7, 3696) = 0.68$, *ns*).[3] Intuitively, this means that we could not rule out that chance alone accounted for the differences in passage retrieval performance.

For reference, Table 1 also shows the percentage of incorrect questions for the end-to-end TREC-10 system associated with the passage retrieval algorithms we studied (data taken from Voorhees [23]). The complete end-to-end systems included answer extraction modules, and returned up to five 50 byte answers.

Overall, the passage retrieval algorithms achieved a higher

[3] The ANOVA excluded questions for which Lucene returned no documents (although we included those questions in Table 1). Excluding these values makes the ANOVA more sensitive; yet, the results were still not significant.

Algorithm	# Incorrect	% Incorrect	MRR
IBM	31	7.18%	0.851
SiteQ	32	7.41%	0.859
ISI	37	8.56%	0.852
Alicante	39	9.03%	0.816
MultiText	44	10.19%	0.845
bm25	45	10.42%	0.810
MITRE	45	10.42%	0.800
stemmed MITRE	63	14.58%	0.762

Table 2: Performance of different passage retrieval algorithms using the oracle document retriever.

	t-test results		
Algorithm	PRISE	Lucene	Oracle
ISI	$t(499) = 0.94$, $p = 0.35$	$t(462) = 0.44$, $p = 0.66$	$t(431) = 1.23$, $p = 0.22$
SiteQ	$t(499) = 1.15$, $p = 0.25$	$t(462) = 0.45$, $p = 0.66$	$t(431) = 0.19$, $p = 0.85$

Table 3: T-test results comparing both ISI and SiteQ with IBM. The results for lenient scoring shown here are not statistically significant; the t-tests for strict scoring were also not significant.

MRR with Lucene as the document retriever, while passage retrievers using PRISE had fewer questions with incorrect answers. Intuitively, passage retrievers using PRISE answer more questions correctly, analogous to higher recall, but passage retrievers using Lucene tend to rank correct answers higher, analogous to higher precision. Figure 2 shows the percentage of questions answered correctly as a function of rank for IBM's and MITRE's passage retrievers. The top-performing IBM algorithm is relatively invariant to the choice of document retriever, whereas the baseline MITRE algorithm shows two distinctly different behaviors depending on the choice of document retrievers. Regardless, the algorithms exhibit the same precision *vs.* recall tradeoff in both PRISE and Lucene.

Manual examination of the results revealed that neither strict nor lenient scoring was perfect. Strict scoring displayed many false negatives, i.e., valid answers scored as incorrect, because the list of known relevant documents supplied by NIST was not exhaustive. Conversely, lenient scoring displayed many false positives, i.e., wrong answers scored as correct, because some of the answer patterns were not discriminating enough. However, we believe that both scoring conditions establish realistic upper and lower bounds on performance.

Although our voting algorithm resulted in a slight performance increase, the improvement was not statistically significant. For Lucene, ANOVA over all runs including voting was not significant (lenient: $F(8, 4158) = 0.79$, *ns*; strict: $F(8, 4158) = 0.69$, *ns*). For the PRISE results, although the ANOVA over all runs was significant (lenient: $F(8, 4990) = 3.27$, $p = 0.0006$; strict: $F(8, 4991) = 3.64$, $p = 0.0003$), a t-test between IBM and the voting algorithm was not. (lenient: $t(499) = -0.67$, *ns*; strict: $t(499) = -0.54$, *ns*)

Table 2 shows the results using the oracle document retriever: every document returned is guaranteed to have at least one instance of the correct answer.[4] This condition tests the performance of passage retrieval algorithms under optimal document retrieval. ANOVA revealed that the difference in performance between the algorithms is statistically significant ($F(7, 3448) = 2.71$, $p = 0.008$).

Focusing on the three passage retrievers that correctly answered the most questions (IBM, ISI, and SiteQ), we found that their performance was not significantly different. The results of our pairwise t-tests under lenient scoring are shown in Table 3.

4. DISCUSSION

In this section, we discuss our three important findings: Boolean querying schemes perform well in the question answering task. The performance differences between various passage retrieval algorithms vary with the choice of document retriever, which suggests significant interactions between document retrieval and passage retrieval. The best algorithms in our evaluation employ density-based measures for scoring query terms.

4.1 Boolean Querying

An immediate conclusion from our study is that in terms of passage retrieval, the performance obtained using the Lucene document retriever is comparable to the performance obtained using the PRISE document retriever. In fact, passage retrieval algorithms using Lucene actually achieve a higher MRR on average. We found this result surprising because in terms of pure document retrieval, boolean query models have been consistently outperformed by more modern approaches. Yet, for the passage retrieval task, the effectiveness of these different document retrievers is quite similar. This result confirms the intuition of many in the question answering community: boolean queries can supply a reasonable set of documents for down-stream components in a question answering system. Many of the top-performing systems in the TREC competitions (e.g., [8, 16]) employ simple boolean queries for this reason, and because a boolean model allows for finer-grained control over query expansion.

4.2 Passage Retrieval Differences

A striking result of our study was the answer to the question "Are the performance differences between passage re-

[4] Note that the strict and lenient measures are identical under the oracle document retriever.

trieval algorithms significant?" The answer to this question depends on the document retriever and has important implications for question answering:

PRISE	significant
Lucene	not significant
oracle	significant

Our results with Lucene show that the performance differences between passage retrieval algorithms were not statistically significant. It may be possible to attribute the differences in performance to pure chance. Based on our experience, we recommend that question answering systems utilizing boolean keyword querying schemes focus more on improving document retrieval. Compared to PRISE results, Lucene appears to have lower recall (consistent with conventional wisdom); as such, methods for boosting recall (e.g., query expansion techniques) should be a priority in the development of question answering systems built on boolean keyword search engines. In fact, at least one system [16] has implemented "feedback loops" which expand or contract boolean queries to improve the quality of a hit list.

In contrast, our results with the PRISE document retriever show that the performance differences attributed to the passage retrieval algorithms are statistically significant; here the passage retriever does make a difference. However, passage retrieval algorithms using PRISE tend to place relevant passages at lower ranks compared to Lucene. Generalizing from this result, we believe that different passage retrieval techniques are still worth exploring, with a focus on better confidence ranking in the context of an IR engine like PRISE.

Finally, our results with the oracle document retriever reveal that performance differences among the various algorithms were still statistically significant even when document retrieval was perfect. This observation suggests that document and passage retrieval technology can be developed independently and still be combined for better end-to-end performance.

4.3 Density-Based Scoring

Examining specific passage retrieval algorithms, we discovered that IBM, ISI, and SiteQ are statistically indistinguishable in terms of performance. Common to all three algorithms is a non-linear boost to query terms that occur very close together in a candidate passage. For example, IBM's use of the cluster word score has a comparable effect to ISI's boost for exact proper names matching. In general, a scoring function based on how close keywords appear to each other is common among passage retrieval algorithms. In contrast, the baseline MITRE algorithm, which performs worse than the others, does not employ any measure of keyword density. We believe that density-based scoring is a critical aspect of passage retrieval.

5. FUTURE DIRECTIONS

For some questions, our implemented passage retrieval algorithms do not return *any* correct answers in the top twenty passages; as a result, any subsequent processing by an answer extraction module is useless. Error analysis of these "hard" questions suggests that improvements in precision could come from a number of simple linguistic strategies: incorporating question type analysis and extracting relations between question terms in the passage.

An example of a useful question type analysis is recognizing definition questions. Many passage retrieval failures can be attributed to the lack of query terms in questions such as "What is an ulcer?", where "ulcer" is the only relevant query term in the question. By using specific query expansion strategies to look for patterns commonly used in definitions (e.g. apposition), performance could be significantly improved.

Another source of error stems from the failure to recognize that crucial syntactic relations are missing in the candidate passages. In some cases, passages that contain all the correct keywords may still not answer a question. Consider the following examples taken from wrong answers returned by actual TREC systems:

> **(Q1003) What is the highest dam in the U.S.?**
> (1) Extensive flooding was reported Sunday on the Chattahoochee River in Georgia as it neared its crest at Tailwater and George *Dam*, its *highest* level since 1929. (AP900319-0047)
> (2) A swollen tributary the Ganges River in the capital today reached its *highest* level in 34 years, officials said, as soldiers and volunteers worked to build *dams* against the rising waters. (AP880902-0066)
> (3) Two years ago, the numbers of steelhead returning to the river was the *highest* since the *dam* was built in 1959. (SJMN91-06144185)

Katz and Lin [11] identified this phenomenon as *ambiguous modification*, where words in a query are found in the candidate answers, but in the wrong modification relationship (in this case, adjective-noun modification). In this specific example, "highest" is the ambiguous modifier, because it could potentially modify many head nouns that co-occur with dam, e.g., "highest level". Katz and Lin's solution extracted syntactic relations (using a parser) from both candidate answers and queries and ensured that the proper relations existed in both. We believe the integration of linguistic processing techniques into passage retrieval algorithms could increase their overall precision.

Overall, we believe that recall is more important than precision at the earlier stages of question answering; in the later stages, however, precision becomes increasingly important. The current paradigm of question answering is based on iterative extraction of shorter and fewer candidates, i.e., first candidate documents, then candidate passages, then candidate answers. In the earlier stages, such as document retrieval, recall is crucial because a relevant document that is not retrieved cannot be processed by subsequent modules. In the later stages of question answering, i.e., answer extraction, precision is paramount—after all, TREC questions usually have a single correct answer, and only one instance is necessary to answer a question. These tradeoffs should be taken into account when designing passage retrieval algorithms in conjunction with specific document retrievers.

6. CONCLUSIONS

Although passage retrieval is an important component of question answering systems, end-to-end performance depends on a variety of other factors. Measuring passage retrieval performance in isolation shows that density-based measures of query terms are important in passage ranking. In addition, our investigations of the effects of document retrieval systems suggest that the interaction between document retrieval and passage retrieval is just as important.

7. ACKNOWLEDGMENTS

This work was supported by DARPA under contract number F30602-00-1-0545. We would like to thank Michael Ernst, Sue Felshin, Anant Saraswat, and Vineet Sinha for their helpful comments on earlier drafts of this paper. All other errors are, of course, our own.

8. REFERENCES

[1] J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-1994)*, 1994.

[2] C. Clarke, G. Cormack, D. Kisman, and T. Lynam. Question answering by passage selection (Multitext experiments for TREC-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.

[3] C. Clarke, G. Cormack, and E. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36:291–311, 2000.

[4] W. S. Cooper. Getting beyond Boole. *Information Processing and Management*, 24:243–248, 1988.

[5] S. Harabagiu, D. Moldovan, M. Paşca, M. Surdeanu, R. Mihalcea, R. Gîrju, V. Rus, F. Lăcătuşu, P. Morărescu, and R. Bunescu. Answering complex, list, and context questions with LCC's Question-Answering Server. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

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

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

[8] E. Hovy, U. Hermjakob, and C.-Y. Lin. The use of external knowledge in factoid QA. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[9] A. Ittycheriah, M. Franz, and S. Roukos. IBM's statistical question answering system—TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[10] A. Ittycheriah, M. Franz, W.-J. Zhu, and A. Ratnaparkhi. IBM's statistical question answering system. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, 2000.

[11] B. Katz and J. Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, 2003.

[12] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[13] M. Light, G. S. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter 2001.

[14] J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D. R. Karger. What makes a good answer? The role of context in question answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT-2003)*, 2003.

[15] F. Llopis and J. L.Vicedo. IR-n: A passage retrieval system at CLEF-2001. In *Proceedings of the Second Workshop of the Cross-Language Evaluation Forum (CLEF 2001)*, 2001.

[16] D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, 2002.

[17] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, 1995.

[18] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, 1994.

[19] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-1993)*, 1993.

[20] M. M. Soubbotin and S. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[21] R. Srihari and W. Li. Information extraction supported question answering. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 1999.

[22] J. L. Vicedo and A. Ferrández. University of Alicante at TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[23] E. M. Voorhees. Overview of the TREC 2001 question answering track. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[24] E. M. Voorhees. Overview of the TREC 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, 2002.

[25] E. M. Voorhees and D. M. Tice. Overview of the TREC-9 question answering track. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.