Answer Passage Retrieval for Question Answering

Andres Corrada-Emmanuel, W. Bruce Croft, Vanessa Murdock

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003-9264

{corrada, croft, vanessa}@cs.umass.edu

ABSTRACT

Document or passage retrieval is typically used as the first step in current question answering systems. The accuracy of the answer that is extracted from the passages and the efficiency of the question answering process will depend to some extent on the quality of this initial ranking. We show how language model approaches can be used to improve answer passage ranking. In particular, we show how a variety of prior language models trained on correct answer text allow us to incorporate into the retrieval step information that is often used in answer extraction, for example, the presence of tagged entities. We demonstrate the effectiveness of these models on the TREC9 QA Corpus.

Keywords

Question Answering Systems, Relevance Models, Answer Language Models, Answer Passage Retrieval

1. INTRODUCTION

Current question answering systems typically include an initial document (or passage) retrieval step to simplify the task of identifying good answer passages [1,2]. The answer passages are then analyzed by a variety of techniques to extract the final answers. The accuracy of the final answer will depend to some extent on the quality of the passages retrieved. This will be especially true in environments where computational resources are bounded (i.e. real systems rather than laboratory experiments) and the number of passages that will be examined is limited. If the answer passage retrieval process can be relied on to deliver high-quality results, then the question answering system will only need to process a small number of top-ranked passages.

In this paper, we show how the language model approaches used recently for document retrieval can be applied to answer passage retrieval. We discuss both the query-likelihood and relevance model approaches [3,4,5] and evaluate their performance on the task of finding 250 byte answer passages specified in the TREC-9 question answering track. [6].

In many question answering systems, a variety of additional features, such as the question type, are used in an *ad-hoc* way to filter out passages unlikely to contain answers. We believe that many of these features could be incorporated directly into the passage retrieval model. To demonstrate this, we investigate various approaches for building answer models for particular question types. The answer model specifies likely text patterns that would be found in answers of a given question type. For example, answers to "location" questions will typically contain noun entities of type *location.* We show how these prior answer language models can be incorporated into both the query-likelihood and relevance models and compare the results.

The following section describes the language modeling frameworks that are the basis of the answer passage retrieval algorithms. We also show how answer models can be incorporated. Section 3 discusses the overall system that was used to retrieve passages, including a discussion about different types of passages. Section 4 describes the document, query collections, and evaluation measures used for the experiments. Section 5 presents the results of the experiments. Section 6 gives a brief overview of related work. The final section summarizes the paper and describes future work

2. LANGUAGE MODEL FRAMEWORKS

A statistical language model is a probability distribution over all possible sentences (or other linguistic units) in a language [8]. It can also be viewed as a statistical model for generating text. In most applications of language modeling, such as speech recognition and information retrieval, the probability of a sentence is decomposed into a product of *n-gram* probabilities. A bigram model is estimated using information about the co-occurrence of pairs of words, whereas a unigram model uses only estimates of the probabilities of individual words. For applications such as speech recognition or machine translation, word order is important and higher-order (usually trigram) models are used. In information retrieval, the role of word order is less clear and unigram models have been used extensively.

The basic approach for using language models for IR assumes that the user generates a query as text that is representative of the "ideal" document. The task of the system is then to estimate, for each of the documents in the database, which is most likely to be the ideal document. That is, we calculate:

$$\arg\max_d P(d \mid q) = \arg\max_d P(q \mid d)P(d) \qquad (1)$$

where the prior $P(d)$ is usually assumed to be uniform and a language model $P_d(q)$ is estimated for every document. In other words, we estimate a probability distribution over words for each document and calculate the probability that the query is a sample from that distribution. This *query-likelihood* retrieval model was first proposed by Ponte and Croft [3] and described in terms of a "noisy channel" model by Berger and Lafferty [4]. This approach to retrieval, although very simple, has produced results that are at least comparable to the best retrieval techniques previously available.

The classical probabilistic model of retrieval [9] is described in terms of a Bayesian classification of documents into the classes

relevant (*R*) and non-relevant (N) for each query. In this case, we rank documents by the ratio $P(d|R)/P(d|N)$. A generative approach to this *document-likelihood* model requires that we estimate the language models $P_R(d)$ and $P_N(d)$. This means that we must estimate probabilities for words in the relevant (and non-relevant) classes of documents. Given that the only information available initially about relevance is the query, this is a challenging task. Lavrenko and Croft [5] show that if the *relevance model* is estimated by:

$$P(w \mid R) \approx P(w \mid q) \qquad (2)$$

where *w* is a word, then the estimation process involves a version of query expansion. This results in better effectiveness than the simple query-likelihood model. Lavrenko and Croft introduce two ways of estimating $P(w|q)$. In this work, we use

$$p(w,q_1,...,q_m) = \sum_D p(d)p(w/d)\prod_{i=1}^{m} p(q_i/d) \qquad (3)$$

It seems reasonable, given these two models, to propose that the documents whose models are most similar to the relevance model should be retrieved. Relative entropy (also known as Kullback-Leibler divergence) is a standard metric for comparing distributions that has worked well in IR experiments. The relative entropy between the relevance model *R* and a document model *D* is defined as:

$$KL(R \parallel D) = \sum_w P(w \mid R) \log \frac{P(w \mid R)}{P(w \mid D)} \qquad (4)$$

This measure produces consistently better retrieval results than either the query-likelihood or document-likelihood approaches. It also simplifies to the query-likelihood model when maximum-likelihood estimates are used for the query terms.

Both the query-likelihood and relevance model approaches have been used for passage-based retrieval of documents [10]. In this case, document language models are replaced with passage language models. Relevance models can be constructed from either documents or passages. Liu and Croft [10] show that passage retrieval is approximately as effective for retrieving relevant documents, and more robust when searching databases containing very heterogeneous documents.

In this paper, we are developing a passage retrieval model specifically for retrieval of answer passages, not relevant documents. The specific passage type used for this work is described in the next section. The baseline for the query-likelihood model is equation (1) assuming uniform prior probabilities. In other words, we rank passages by

$$P(A \mid Q) \propto \prod_{i=1}^{n} P(q_i \mid A) \qquad (5)$$

where A is an answer passage. As with all language model approaches, the smoothing of probabilities is a major issue. In our experiments, we used interpolation with a collection model and Dirichlet smoothing [11]. The collection probabilities were estimated using the whole TREC-9 collection.

In the case of the relevance model, equation (2) is used with a uniform prior (*p(d)*). The relevance model is built from the top ranked passages rather than documents (in our case, the top 30 passages).

In order to incorporate additional information about an answer model, we used the prior probability in both the query-likelihood and relevance model equations. In other words, *p(d)* (or *p(a)* in this case) is modified based on the probability of the text given a particular question class. To calculate these probabilities, we constructed answer models from training data for the main TREC question classes.

Our answer models are of two types. The first type is the bigram model familiar from other NLP applications were the training data is used to learn the 2-gram and 1-gram statistics of words in the text. The second type is "template" based. Templates are text patterns strongly associated with particular question types. These templates can either be incorporated manually [14] or learned automatically in a supervised learning system [13, 16].

3. SYSTEM ARCHITECTURE

The goal of our research is to effectively rank passages directly from the query. Currently, however, the toolkit we are using (Lemur[1]) does not support direct passage retrieval. For these experiments, therefore, we first retrieve documents, then split these documents into passages. The passages are then ranked using language model techniques.

Passages were created using the following procedure. The top 20 retrieved documents were selected (early tests showed that increasing this number had no effect in system performance). These selected documents were split into sentences using a heuristic sentence segmenter.

The sentences were sequentially formed into passages that were at most 250 bytes in length and possibly overlapping with neighboring passages. If a sentence was longer than 250 bytes, as did occur, it was dropped. We call the passages produced by this procedure "sentenced" passages. Our procedure yielded an average of 434 passages per question that needed to be ranked. We use these overlapping sentenced passages for various reasons.

We know from preliminary studies that passages constructed with a sliding window that does not respect sentence boundaries performed better than these sentenced ones. But for a given document we get many more windowed passages than sentenced ones. By using overlapping sentenced passages we have developed a system complex enough to resolve issues related to overlap between passages (see discussion in section 5.1) but simple enough to quickly perform our experiments.

In addition, we were interested in developing a system that could be seen as a pre-processor for an NLP system that needed grammatical sentences while at the same time respecting the 250-byte limit so our performance could be compared to that of other systems that have performed the same task.

These sentenced passages were subsequently indexed with no stemming and a stop list consisting of single characters only. We then performed retrieval on these passages applying the techniques we described above. Our retrieval for each question was limited to only those passages that came from its retrieved documents. Doing otherwise would be contrary to our approach: start with retrieved documents for a question and then identify

[1] www.cs.cmu.edu/~lemur/

passages within those documents that are likely to contain its answer.

4. EXPERIMENTAL SETUP

4.1 Dataset

The TREC-9 QA Dataset was used for our experiments. It consists of 979,000 documents with about 3GB of text from various news sources (AP newswire, Wall Street Journal, San Jose Mercury News, Financial Times, LA Times, and FBIS) [4].

The questions set for TREC-9 consists of 693 questions that are generally characterized as being of the "factual" type: "What is the longest river in the world?", "Who is Colin Powell?", etc. It was found during the TREC-9 evaluation that some questions were too ambiguous or did not have an answer within the corpus [4]. Thus, 682 questions were finally used to report evaluation metrics for the TREC-9 QA tasks, and it is this restricted set that we use in the experiments reported here.

The TREC-9 questions have been classified into "question types" by Thomas S. Morton of the University of Pennsylvania (private email communication). We used six of his question type classifications:

- Amount (A): "How many zip codes are there in the U.S.?"
- Famous (F): "What is Francis Scott Key best known for?"
- Location (L): "Where is Glasgow?"
- People (P): "Who invented basketball?"
- Time Point (T): "What year did Montana become a state?"
- Other(X): This is a generic, catch-all for questions that did not fit the other types

There are various reasons for using these question types. Many QA systems have a question classification component. Thus it was reasonable to investigate if our statistical approach benefits from knowing a question's type. In addition, we were interested in seeing if training data improves the performance of a statistical approach. The A questions are the smallest set in our group with 52 questions. We took this as the practical limit for testing approaches that divided the questions into training and testing portions. And finally, this AFLPTX set contains 589 out of the 682 questions evaluated in TREC-9.

So the results we present here are for each individual question in the AFLPTX set, the AFLPTX set combined and all of the TREC-9 evaluated questions. The individual question type results allows us to see if performance varies across questions types. The combined result (referred to as AFLPTX in the tables) would be the performance of a system that had perfect question classification performance. And finally the results for all questions allow an approximate comparison between our work and previously reported results for the TREC-9 evaluation.

4.2 Task and performance metric

The TREC-9 QA track had two experimental conditions: returning answer strings limited to 50 bytes, and answer strings limited to 250 bytes. The results of the TREC-9 evaluation make it clear that performance correlated with answer string length – longer strings improved a system's performance. All experiments reported were done for the 250-byte task [4].

The performance metric for TREC-9 was the Mean Reciprocal Rank (MRR) measured down to the top 5 answer strings returned for each question. So systems were rewarded for returning a correct answer within the top 5 ranked answer strings, but received no credit for answers returned below the 5th position.

In lieu of using human judges to decide if retrieved answer passages did answer the questions, we utilized the regular expressions developed by NIST in an attempt to develop a "reusable test collection". These regular expressions were constructed so that "almost all" strings judged to be correct by the original judges of the TREC-9 evaluation would be matched [4]. The Kendall τ association between system performance measured by human judges and that measured using regular pattern matching was found to be 0.94 for the systems that participated in the TREC-9 250-byte task [4].

5. EXPERIMENTAL RESULTS

5.1 Query Likelihood Baseline

The query likelihood baseline was described in section 2. It returns a ranked list of passages using equation (3).

After examining the initial results, we found that just using the query likelihood ranking may not the most effective approach. Passages at the top of a ranked list may be overlapping and come from the same document. To test the effect this has on the results, we tested two strategies for removing passages from the same document. The results from the original ranking are shown in the first column of Table 1 under the heading "Unvetted". The second column, "one per document", only allows the top ranked passage from a given document to remain in the ranked list. The second strategy, "non-overlapping", allows multiple passages from the same document by removing passages that overlap with a higher ranked passage. It is clear from these results that vetting the list returned by the query likelihood ranking produces a small improvement for every question class, although there is not much difference between the two strategies for removing passages.

All results mentioned in the rest of the paper will quote the MRR scores derived using the non-overlapping strategy.

Table 1. Query Likelihood MRR performance (%)

Question Set	Unvetted	One per Document	Non-overlapping
A (52)	20.7	22.5	22.4
F (84)	58.5	58.8	59.2
L (109)	43.6	44.1	44.9
P (109)	53.4	55.4	55.4
T (71)	27.6	27.8	27.7
X (164)	34.9	36.3	36.4
AFLPTX (589)	41.2	42.2	42.5
All(682)	39.3	40.6	40.8

5.2 Relevance Model Results

The relevance model baseline was done using equation (2). As mentioned before, the relevance models are built using the top-ranked passages from an initial retrieval. As one can see from the baseline results for query likelihood, however, the best performance is obtained when we vet the ranked list by requiring that only non-overlapping sentenced passages be allowed in the list. This implies that the relevance model results could also be improved by vetting. Table 2 shows that this is the case for all question types.

In general, the relevance model was more effective than the query-likelihood model, sometimes substantially. This is a little surprising given that relevance models are equivalent to a massive query expansion [5], and answer passage retrieval is often done in other systems using strict matching criteria. One of the question classes, People, did not perform better with relevance models (although the results were not worse either). This suggests that the effectiveness of query expansion may depend on the type of question.

Table 2. QA Relevance Models MRR performance (%)

Question Set	Query Likelihood	RM	% improvement
A (52)	22.4	23.9	6.7
F (84)	59.2	62.5	5.6
L (109)	44.9	51.6	14.9
P (109)	55.4	56.2	1.4
T (71)	27.7	32.0	13.4
X (164)	36.4	41.3	13.5
AFLPTX (589)	42.5	46.3	8.9
All (682)	40.8	42.9	5.1

It is interesting to note that the relevance models always had their best performance at a smaller value for the Dirichlet constant than the query likelihood results. This is encouraging since it means that the probability estimates derived from the passages were weighted more than in the query likelihood baseline. That is, relevance models depend less on the use of smoothing. In addition, whereas the number of words used to estimate the relevance models for document retrieval is in the order of hundreds, for this task the best value was obtained when the words used to estimate the models were in the order of ten words.

5.3 Bigram Answer Model Results

Both of the results reported so far, query likelihood and relevance models have used the assumption that the prior probability of an answer passage is constant. We wanted to investigate how both methods would be affected by the use of probability priors for the answer passages. We investigated this approach by looking at obtaining passage priors from bigram language models for correct answer passages.

For example, one would expect that location questions (L type) would invariably have a location within the correct answers. This suggests that answer models trained on data where various entities are abstracted from the text could lead to improvements in the query likelihood baseline.

We tested this hypothesis by limiting ourselves to the 6 question types that had more than 50 questions (A, F, L, P, T, X).

Our goal was to train answer models that captured the "structure" of answer text rather than the actual words themselves. To that end, we normalized the candidate answer passages using a noun entity tagger. The text for all candidate answer passages for the AFLPTX questions were processed using BBN's IdentiFinder [12] to tag for the following entities:

- PERSON
- PERCENT
- DATE
- MONEY
- LOCATION
- ORGANIZATION

These tags were used to replace the tagged text itself. Thus, for example, eliminating most mentions (the tagger is not perfect) of specific locations to a single token: _location_. This procedure yielded a vocabulary of about 53K words for our AFLPTX candidate answer passages versus a vocabulary of about 94K words if no tagging with replacement had occurred.

We then proceeded to train answer models specific to each question type with the following procedure. We used a 10-fold cross-validation separation of questions into training and testing sets. The question set for each question type was divided into ten random partitions. Each partition assigned 90% of the questions to a training set and 10% to a testing set. So each question was used nine times in a training set and once in a test set.

There is one subtlety related to how this partitioning was done. The TREC9 QA Corpus contains many examples of question variants – groups of questions that are essentially reworded versions of the same question. For example, questions 408, 701-4 are variants of the question: "What kind of an animal is Winnie The Pooh?" The effect of these variants is that questions can share the same passage as a correct answer. Thus we randomly partitioned questions with the constraint that variant groups were not split between testing and training sets. This guaranteed that no correct answer used for training was ever a correct answer for a test question.

Each random partition of a question type now had the correct answer passages for 90% of the questions to train on. The main distinction between all the answer models we constructed and report here is the extent to which they used this training text.

The first model we tested is called the "whole passage" model. It utilized all of the sentences available in passages that were identified as correct for the training questions. The second model is called the "matching line" model. It used only the sentence that satisfied the regular expression for the correct answer. And lastly, we investigated what type of information from templates could be incorporated into answer models.

All of our answer models were trained as bigram models that used absolute discounting to deal with unseen words and count one

words since our training data was so small and it is generally assumed that absolute discounting works better than other schemes such as Turing smoothing for small amounts of training text.

Our procedure for calculating an estimate of p(a) given a candidate answer passage and a prior language model was as follows: The inverse of the perplexity of the candidate answer passage under the answer model was calculated. The inverse of the perplexity is just the geometric mean of the word probabilities under the model, so we took this as our initial estimate for the prior:

$$p(a) \approx \left(\prod_{i=1}^{n} p(w_i \mid w_{i-1})\right)^{1/n} \quad (6)$$

By taking the geometric mean, we compensate for the varying word lengths of the candidate passages. In addition, we smoothed the resultant probability estimate by raising it to a power between one and zero. By varying the exponent to which we raised the prior probability we could tune for our use of answer models. An exponent of zero gives a uniform prior (equivalent to query likelihood ranking). Since the prior probabilities can be rescaled to sum up to one, we view raising to a power as a smoothing procedure necessary to deal with noisy training data.

We began by studying how answer models performed under the query likelihood approach shown in equation 1.

5.3.1 Whole passage answer models

Whole passage answer models were trained on all of the sentences present in a passage that was part of the training correct answers. All classes improved but overall the improvement is small and it suggests that better models can be trained by limiting further the text we use for training.

Table 3. Whole passage answer models MRR performance (%)

Question Set	Query Likelihood	Whole Passage AM	% relative improvement
A (52)	22.4	23.9	6.7
F (84)	59.2	59.3	0.2
L (109)	44.9	47.8	6.5
P (109)	55.4	55.7	0.5
T (71)	27.7	30.6	10.5
X (164)	36.4	37.8	3.7
AFLPTX (589)	42.5	43.9	3.3

5.3.2 Matching Line Answer Model

We constructed a second version of a prior answer model by limiting our training text further. Matching line answer models were trained using only the sentences that matched a correct answer pattern. Their performance is shown in table 4. We can see that although the text amount used for training diminished, the quality of the models improved. Albeit, the Person question class showed no improvement.

Table 4. Matching line answer models MRR performance (%)

Question Set	Query Likelihood	Match Line AM	% relative improvement
A (52)	22.4	23.5	4.9
F (84)	59.2	59.6	0.7
L (109)	44.9	49.4	10.2
P (109)	55.4	56.2	1.4
T (71)	27.7	31.2	12.6
X (164)	36.4	39.0	7.1
AFLPTX (589)	42.5	44.7	5.2

5.3.3 Template Answer Model

A number of systems have used templates for question answering [13,14,15,17]. We generated templates for the "location" class using the *Snowball* system [16]. *Snowball* generates a set of tuples, which consist of the terms to the left of the first seed, the terms between the seeds, the terms to the right of the second seed, the seeds themselves, and whether they should be ordered or not. An example of a seed is "<Exxon, Irving>", which expresses the relation "<ORGANIZATION, LOCATION>". We extracted a total of 722 templates for the location class expressing organization-location, location-location, location-person, and location-date relations. A template is a cluster of similar patterns. Each term in the template has a score that is the similarity between the pattern the term was in, and the pattern that it matched. An example of a template is:

order="true" tag1="PERSON" tag2="LOCATION" left="<the:0.268><is:0.268>" middle="<at:1.0>" right="" belief= 0.0685

In addition, the template, itself, has a "belief" score that is an indication of its quality.

5.3.3.1 Incorporating Templates

Templates and patterns are, by their construction, heuristic. The similarity and belief scores assigned to them are attempts to establish the likelihood of an answer candidate passage matching a given template. Matching a template is strong evidence for being a correct answer so we wanted to utilize them in our statistical framework. The following experiments show how they can be used to create answer models that can produce prior probabilities.

We trained a unigram language model on the words from the templates. Since we didn't have word frequency statistics, the similarity score for each term and the belief scores for each template were massaged into a probability distribution, and then those "probabilities" were used in place of word frequencies. This approach yielded an MRR of 49.1.

Since we didn't have actual word frequency statistics, but the words themselves were a highly targeted vocabulary, we trained a bigram language model from the texts of answers known to be correct, but used the words from the templates as the vocabulary. Restricting our vocabulary in this focused way is resulted in an average MRR of 50.6.

We created bigrams from the templates and trained the model on those, as if they had been gleaned from training data. All possible bigrams were created from the templates, respecting the left/middle/right substring boundaries. The language model was

trained with the pattern vocabulary. The average MRR from this approach was 51.0.

We selected sentences from our known correct answer texts that had "location-location" entity pairs, "location-organization", "location-date", and "location-person", and created a distribution of the bigrams appearing in these sentences. The language model trained on these targeted sentences produced an MRR score of 52.0. In each case the vocabulary used was restricted to the vocabulary of the training set.

Table 5. MRR performance of "L" questions with four ways of incorporating templates

Question Set	Query Likelihood	Pat. Scores	Pat. Vocab	Pat. bigram	LM
L	44.9	49.1	50.6	51.0	52.0

One criticism of templates is that they are inflexible. We would like a more flexible system, for question types for which we don't have entity relations or template information.

For each question type, we defined the entity relations by choosing pairs of entities that occurred more frequently for a given question type. We trained bigram language models using sentences that contained any of the entity relations for a given question type. The vocabulary was restricted to the training set. Table 6 shows the results in average MRR score for this approach.

Incorporating relevance models showed improvement for most question types. In a template based system, incorporating relevance models has the effect of smoothing for terms not found in the template. This improves the situation for answers that are not captured in templates, but decreases the overall effects for template matches. Thus relevance models would be most useful for question types whose answers have a low degree of match to the templates.

Table 6. Relation Answer Models MRR performance (%)

Question Set	Query Likelihood	AM	% relative improvement
A	22.4	23.1	3.1
F	59.2	63.7	7.6
L	44.9	54.0	20.2
P	55.4	54.8	-1.2
T	27.7	32.7	18.1
X	36.4	44.9	23.4
AFLPTX	42.5	47.7	12.2

5.4 Answer Models with Relevance Models

Since bigram answer models improved the baseline query-likelihood performance, we also tested how the use of the model priors would improve the relevance model performance. This was done using the prior in equation (2).

The results are shown in Table 4. We show how two of our models: the "matching line" and "template" models combine with Relevance Models. We show the relative improvement in the baseline query likelihood results in parentheses.

Table 7. RMs with bigram AM priors MRR performance (%)

Question Set	Query Likelihood	Match Line AM + RM	Template AM + RM
A (52)	22.4	24.1 (+07.6)	23.6 (+5.4)
F (84)	59.2	63.9 (+07.9)	65.9 (+11.3)
L (109)	44.9	53.1 (+15.4)	56.3 (+25.4)
P (109)	55.4	55.4 (+00.0)	56.0 (+1.1)
T (71)	27.7	31.6 (+14.1)	30.1 (+8.7)
X (164)	36.4	41.4 (+13.7)	45.7 (+25.5)
AFLPTX (589)	42.5	46.7(+09.9)	48.6 (+14.4)

6. RELATED WORK

The use of passage retrieval for question answering has been studied before. The IR-n system [13] from the University of Alicante has focused on this task. It uses a heuristic measure to retrieve passages with a fixed number of sentences, with best performance obtained at around 20 sentences. Besides our use of language models, this work differs from ours because we only considered passages that conformed to the 250-byte TREC-9 QA task.

The University of Sheffield [14] has also used passage retrieval in their QA system. But as in the IR-n system, performs the passage retrieval with a heuristic measure and retrieves passages longer than the 250-byte limit.

In terms of using answer templates, a variety of systems have used heuristic methods of deriving and applying templates. One example of such a system is the AskMSR system described in [17]. This system generates a hand-crafted set of query rewrite rules, and answer templates and uses an heuristic approach to scoring the answer candidates. Ravichandran and Hovy [13] utilize the approaches in [14] and [15], using a bootstrapping technique to learn regular expressions from the web, using suffix trees to find the optimal substring length.

7. CONCLUSIONS AND FUTURE WORK

The results shown in this paper demonstrate that it is possible to significantly improve answer passage ranking by incorporating additional features related to passage quality into the retrieval model. Table 7 summarizes the percentage improvements of different approaches compared to the baseline query likelihood model. This shows that query expansion via the relevance model generally improves performance, but the incorporation of answer models is better for some question classes. The incorporation of answer models into the priors for the relevance model produces the best results with consistent and substantial improvements in all question classes. The incorporation of template information was previously demonstrated to improve performance, and we showed several ways this information can be incorporated into a statistical system.

The best results in this table are competitive with the best results achieved in TREC-9, but they are somewhat difficult to compare directly because answer models were not constructed for all question classes. It should be emphasized that this high level of performance was achieved in what is effectively a single retrieval pass with no additional *ad-hoc* filtering done afterwards, as is common with most other systems.

In future work, we plan to investigate other features that could be incorporated into the retrieval model. To do this, we may use maximum entropy language models, which are more flexible with regard to the types of features in the model [8]. We also plan to further explore the automatic learning of answer templates, and the most appropriate way to incorporate the information into a statistical QA system.

8. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903, and in part by Advanced Research and Development Activity under contract number MDA904-01-C-0984. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] Ittycheriah et al. IBM's Statistical Question Answering System. In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2000.

[2] Elworthy, D. Question Answering using a large NLP System. In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2000.

[3] J. Ponte, W.B. Croft, "A language modeling approach to information retrieval", *Proceedings of ACM SIGIR 1998*, 275-281, 1998.

[4] A. Berger, J. Lafferty, "Information Retrieval as statistical translation", *Proceedings of ACM SIGIR 1999*, 222-229, 1999

[5] V. Lavrenko, W.B. Croft, "Relevance-based language models", *Proceedings of ACM SIGIR 2001*, 120-127, 2001

[6] Voorhees, E. Overview of the TREC-9 Question Answering Track. In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2000.

[7] R. Rosenfeld, "Two decades of statistical language modeling: where do we go from here?", Proceedings *of the IEEE*, 88(8), 1270-1288, 2000.

[8] K. Sparck Jones, S. Walker, S E. Robertson, "A probabilistic model of information retrieval: development and comparative experiments", Parts 1 and 2, *Information Processing and Management*, 36(6): 779-840, 2000.

[9] Liu, X. and Croft, W.B., "Passage Retrieval Based On Language Models," *Proceedings of CIKM '02 conference*, 375-382, 2002.

[10] C. Zhai, J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval", *Proceedings of ACM SIGIR 2001*, 334-342, 2001.

[11] Bikel, Daniel M., Schwartz, Richard L., and Weischedel, Ralph M. An Algorithm that Learns What's In a Name. Machine Learing vol. 34, p. 211-231, 1999.

[12] Llopis, Fernando, Vicedo, Jose Luis, and Fernandez, Antonio. Passage Selection to Improve Question Answering. In Workshop on Multilingual Summarization and Question Answering, 2002.

[13] Ravichandran, D. and E.H. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002) Conference*. Philadelphia, PA, July 7-12.

[14] M.M Sabboutin, and S.M. Sabboutin. "Patterns of Potential Answer Expressions as Clues to the Right Answer." *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD.

[15] G. Lee, et al. "SiteQ: Engineering High Performance QA system Using Lexico-Semantic Pattern Matching and Shallow NLP." In *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD.

[16] E. Agichtein and L. Gravano. "*Snowball*: Extracting Relations from Large Plain-Text Collections." In *Proceedings of the 5th ACM International Conference on Digital Libraries*. 2000.

[17] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. "Web Question Answering: Is More Always Better?" In *Proceedings of ACM SIGIR 2002*. p. 291-298.