Dedicated Backing-Off Distributions for Language Model Based Passage Retrieval

Munawar Hussain, Andreas Merkel and Dietrich Klakow
Spoken Language Systems
Saarland University, Saarbrücken, 66123, Germany
{Munawar.Hussain|Andreas.Merkel|Dietrich.Klakow}@lsv.uni-saarland.de

Abstract

Passage retrieval is an essential part of question answering systems. In this paper we use statistical language models to perform this task. Previous work has shown that language modeling techniques provide better results for both, document and passage retrieval.

The motivation behind this paper is to define new smoothing methods for passage retrieval in question answering systems. The long term objective is to improve the quality of question answering systems to isolate the correct answer by choosing and evaluating the appropriate section of a document.

In this work we use a three step approach. The first two steps are standard document and passage retrieval using the Lemur toolkit. As a novel contribution we propose as the third step a re-ranking using dedicated backing-off distributions. In particular backing-off from the passage-based language model to a language model trained on the document from which the passage is taken shows a significant improvement.

For a TREC question answering task we can increase the mean average precision from 0.127 to 0.176.

1 Introduction

Recently lot of work has been carried out on open-domain Question Answering Systems. These QA Systems include an initial document and/or passage retrieval step. Retrieved passages are then further processed using a variety of techniques to extract the final answers. The passage retrieval method strongly influences the performance of QA System. This is especially true for real systems where computational resources are limited. A good passage retrieval system will mean that only small number of top ranked passages needs to be analyzed to find the answer. In this paper[1] we compare the existing retrieval methods, both traditional and language modeling based, for document and passage retrieval. We have used the AQUAINT document collection as training and test corpus. Out of all methods tested, by choosing the best passage retrieval method as our baseline, we define and test new language models to improve retrieval performance. These language models are defined on different data collections (passage collection, document collection, corpus) and are interpolation based unigram language models.

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 discusses document retrieval. Section 4 presents the passaging of documents and passage retrieval performed. Section 5 explains the process of re-ranking. We conclude the paper by discussing our results and future work in Section 6.

2 Related Work

This section discusses the state of the art in the field of passage retrieval.

Passage retrieval is an important component of QA Systems and it directly influences overall performance.

C. L. A. Clarke et. al. [Clarke *et al.*, 2000] developed the MultiText system, which implements a technique to efficiently locate high-scoring passages.

A language modeling based approach was used by Andres Corrada-Emmanuel et. al. [Corrada-Emmanuel *et al.*, 2003]. They examined the effectiveness of language models in passage ranking for a question answering system.

Dell Zhang et. al. [Zhang and Lee, 2003] also developed a language modeling approach to passage question answering. Their system consists of a question classification component and a passage retrieval component.

Stefanie Tellex et. al. [Tellex *et al.*, 2003] carried out a Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. They evaluated a number of passage retrieval algorithms, and one new algorithm of their own called Voting. They implemented a 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.

Some work has been done to improve the document retrieval by performing passage retrieval.

James P. Callan [Callan, 1994] examined passage level evidence in document retrieval.

Use of language modeling for passage retrieval and comparison with document-based retrieval was done by Xiaoyong Liu et. al. [Liu and Croft, 2002].

Deng Cai et. al. [Cai1 *et al.*, 2004] explored the use of page segmentation algorithms to partition web pages into blocks and investigated how to take advantage of block-level evidence to improve retrieval performance in the web context.

3 Document Retrieval

This section explains our experimental setup for document retrieval. The retrieved document set will be later used for passage retrieval.

[1]This work was partially supported by the BMBF project Smartweb under contract 01 IMD01 M

3.1 Dataset

The training document set or corpus for evaluation is the AQUAINT collection that consists of 1,033,461 documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. We selected AQUAINT as some well established standard task, which is helpful to compare our work with the state of the art. Our question set for evaluation contains 50 factoid questions, from TREC topic 1394 to 1443. In all our experiments, stemming is applied. No stop word removal is performed. Relevance judgments are obtained from the judged pool of top retrieved documents by various TREC participating retrieval systems.

3.2 System Architecture for Document Retrieval

The inputs to the system are the corpus and a set of questions. The output is a ranked list of documents for each question. Bellow is an explanation of each of the system components.

KeyFileIndexer & Stemmer: This component builds a key file index of AQUAINT corpus. Stemming is done along with indexing, using the Krovetz stemmer. The generated index is used by each retrieval method.

Question Stemmer: It is responsible for converting questions into queries by stemming them. Again the Krovetz stemmer is used for stemming.

Retriever: This component is responsible for the actual retrieval of documents. Retrieval methods are explained in following section.

3.3 Experimental Methods

A number of popular retrieval techniques exist, which include both traditional and language modeling techniques. We evaluate the performance of some of these techniques on our test data. The retrieval methods evaluated in this section are standard TFIDF, OKAPI, and the language modeling framework. The Dirichlet Prior, Jelinek-Mercer, and Absolute Discounting smoothing methods are the three methods that we have tested. They belong, in general sense, to the category of interpolation-based methods, in which we discount the counts of the seen words and the extra counts are shared by both the seen words and unseen words. The Lemur toolkit is used to run the experiments, because it is efficient and is optimized for fast retrieval. It provides both traditional and language modeling based retrieval algorithms and has been used by many research groups in the IR community.

3.4 Evaluation Methodology

Our goal is to study the behavior of individual retrieval methods and smoothing techniques as well as to compare different methods. Unlike traditional retrieval techniques, in case of language modeling retrieval technique, for each smoothing method we experiment with a wide range of parameter values. In each run, the smoothing parameter is set to the same value across all queries and documents. (While it is certainly possible to set the parameters differently for individual queries and documents through some kind of training procedure, it is beyond the scope of our work.) In order to study the behavior of a particular smoothing method, we examine the sensitivity of non-interpolated average precision to variations in a set of selected parameter values. Along with finding the optimal value of smoothing parameters, we also need to find the optimal number of retrieved documents N. Therefore we first fix the number of retrieved documents by comparing the non-interpolated average precision for varying number of documents retrieved, using each retrieval method. For the purpose of comparing smoothing methods, we first optimize the performance of each method using the non-interpolated average precision as the optimization criterion, and then compare the best runs from each method. The optimal parameter is determined by searching over the entire parameter space.

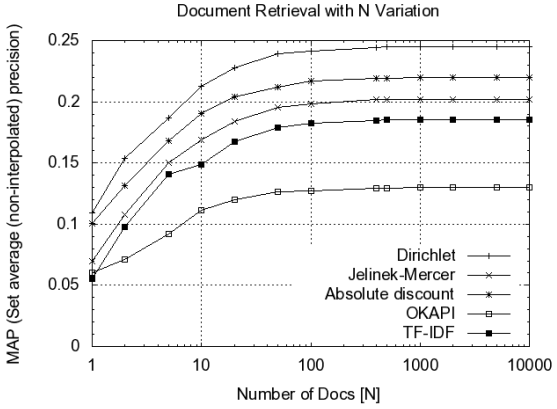

Figure 1: Document Retrieval with varying number of documents retrieved. For Dirichlet Prior the value of prior is set to 2000, for Jelinek-Mercer the value of λ is set to 0.8 and for Absolute Discounting the value of δ is set also to 0.8.

3.5 Experimental Results

This section explains results obtained from different retrieval methods. We first derive the expected influence of number of documents retrieved by plotting the non-interpolated average precision against document number for each retrieval method. We examine the sensitivity of retrieval performance by plotting the non-interpolated average precision at N documents against different values of the smoothing parameter. Following section explains the reason for retrieving a finite number of N documents per query.

Document size tuning

In this section, we study the behavior of each retrieval technique for different numbers of documents retrieved. We examine the sensitivity of retrieval performance by plotting the non-interpolated average precision, with fixed smoothing parameter for this experiment where required, against different number of documents retrieved. The smoothing parameter values are taken from previous work [Zhai and Lafferty, 2001]. The plot in Fig 1 displays the non-interpolated average precision for different number of documents retrieved. It can be seen that with increase in document number, performance also increases. It can also be seen that the increase in performance after 500 documents is relatively marginal. For number of retrieved documents N greater than 500 the cost of computing is significantly larger compared to the gain in performance. Therefore N is fixed at 500. Overall the Dirichlet Prior performed best by far. One reason for this could be that our queries on average are not verbose. Our experiments support the claim that language modeling techniques perform better than traditional ones, as TF-IDF and OKAPI performed worse.

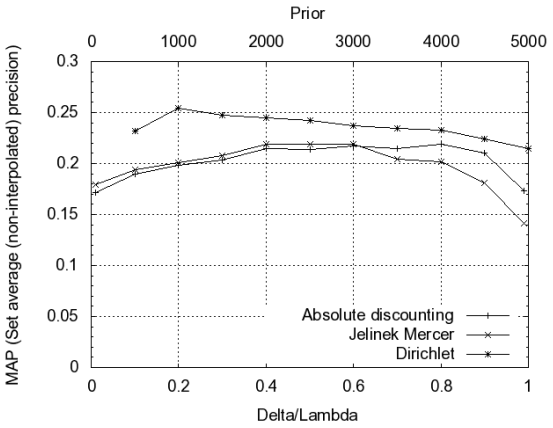

Figure 2: Plot of non-interpolated average precision against smoothing parameter, with delta/lambda varying from 0.01 to 0.99 and prior varying from 500 to 5000. Number of retrieved documents fixed at 500.

Parameter tuning for language modeling techniques
In this section, we examine the sensitivity of retrieval performance by plotting the non-interpolated average precision at 500 retrieved documents against different values of the smoothing parameter. Following is the analysis of our results.

For Jelinek-Mercer, the value to λ is varied between zero and one. The plot in Fig 2 shows non-interpolated average precision for different settings of λ. As depicted in plot, optimal value of λ is near 0.5, which indicates that our queries are of mixed length. According to [Zhai and Lafferty, 2001], for short queries optimal point is around 0.1 and for long queries optimal point is generally around 0.7. This is because long queries need more smoothing and less emphasis is placed on the relative weighting of terms.

Dirichlet Prior: For Dirichlet Prior, the value of prior μ is varied between 500 and 5000 with intervals of 500. The plot in Fig 2 illustrates the non-interpolated average precision for different settings of the prior sample size. As mentioned in [Zhai and Lafferty, 2001], the optimal prior μ vary from collection to collection and depends on query lengths. For our dataset and questions it is around 1000.

Absolute Discounting: For Absolute Discounting, the value to δ is varied between zero and one. The plot in Fig 2 shows non-interpolated average precision for different settings of δ. The optimal value of δ is near 0.8, which fortifies the claim by [Zhai and Lafferty, 2001] that the optimal value for δ tends to be around 0.7.

Overall the Dirichlet Prior performed best using prior of 1000 and 500 retrieved documents. Then came Absolute Discounting, which is better than Jelinek-Mercer. The good performance of Dirichlet Prior is relatively insensitive to the choice of μ. Indeed, many non-optimal Dirichlet runs are also significantly better than the optimal runs for Jelinek-Mercer and Absolute Discounting. This is because our queries are not long. As for long queries, Jelinek-Mercer is supposed to perform the best. As displayed by Table 1, TF-IDF performed slightly worse than Jelinek-Mercer, while OKAPI performed even worse.

4 First Pass Passage Retrieval

Passage retrieval is mainly used for three purposes. Firstly, passage retrieval techniques have been extensively used in standard IR settings, and have proven effective for document retrieval when documents are long or when there are topic changes within a document. Secondly, from an IR system user's standpoint, it may be more desirable that the relevant section of a document is presented to the user than the entire document. Thirdly, passage retrieval is an integral part of many question answering systems. We are performing passage retrieval for question answering systems. This section explains our methodology to establish a baseline using existing techniques developed for passage retrieval. For our experiments, we first retrieve documents (Section 3), then split these documents into passages.

Method	Parameter	MAP
Dirichlet Prior	$\mu = 1000$	0.254
Jelinek-Mercer	$\lambda = 0.5$	0.219
Absolute Discounting	$\delta = 0.8$	0.219
TFIDF	-	0.185
OKAPI	-	0.130

Table 1: Non-interpolated average precision for best run of each retrieval methods. With μ of 1000, δ of 0.8, and λ of 0.5

4.1 Experimental Setup

This section explains our setup for passage retrieval.

Passage Making
Passages are created using the following procedure. The top 500 retrieved documents are selected, see Section 3 for details of the document retrieval. The selected documents are then split into passages by a "passage maker". Our passage making technique is based on document structure [Berger and Lafferty, 1999] [Agichtein and Gravano, 2000] [Clarke*et al.*, 2000]. This entails using author-provided marking (e.g. period, indentation, empty line, etc.) as passage boundaries. Examples of such passages include paragraphs, sections, or sentences. Since our corpus is nicely structured (SGML form), we used paragraphs as passages.

Dataset
The query topics are the same as used for document retrieval (Section 3). For each query we have a distinct corpus consisting of passages created from the top 500 retrieved documents.

Experimental Methods
For passage retrieval we used the same set of retrieval methods as for document retrieval explained in Section 3. Likewise, the evaluation methodology is the same as for document retrieval (Section 3).

4.2 Experimental Results

Following subsections discuss results.

Passage document size tuning
In this section, we study the behavior of each retrieval technique for different number of retrieved passages, which is similar to what we did for document retrieval in Section 3. The plot in Fig 3 shows the non-interpolated average precision for different number of retrieved passages. It can be seen that with increase in number of retrieved passage documents, performance also increases. But the increase in performance after 500 passages is relatively marginal. Therefore the passage document number N is fixed at 500. Overall Dirichlet Prior performed best. Our experiments also show that there is no significant performance difference between retrieval methods, i.e. the curves are pretty

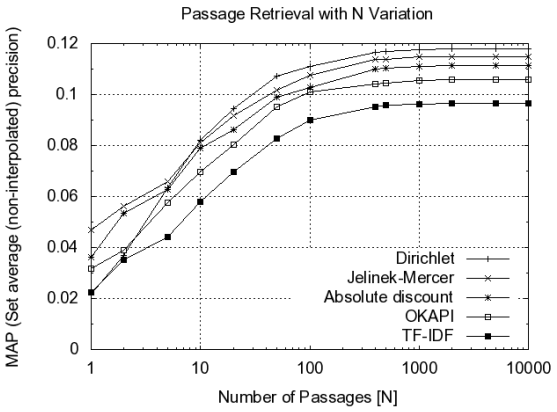

Figure 3: Passage Retrieval with varying number of retrieved passages. For Dirichlet Prior the value of prior is set to 1000, for Jelinek-Mercer the value of λ is set to 0.4 and for Absolute Discounting the value of δ is set also to 0.4.

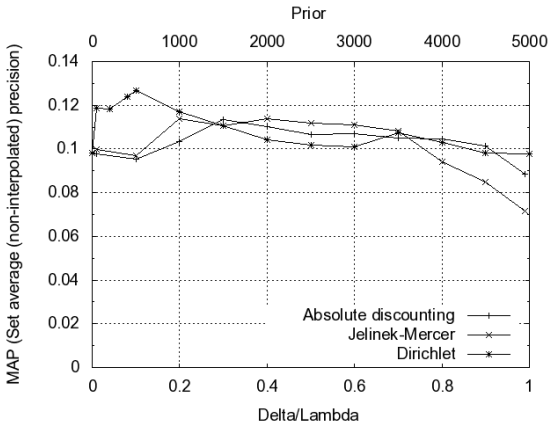

Figure 4: Plot of non-interpolated average precision against smoothing parameter,, with delta/lambda varying from 0.01 to 0.99 and prior varying from 500 to 5000. Number of retrieved passages fixed at 500.

close to each other. Performance of OKAPI is slightly worse than language modeling techniques. TF-IDF showed worse performance. Another noticeable fact is that Dirichlet Prior performance improves significantly for N between 1 and 10.

Parameter tuning for language modeling techniques
In this section, we study the behavior of individual smoothing methods, as we did for document retrieval in Section 3. Below is an analysis of our results.

Jelinek-Mercer smoothing: For Jelinek-Mercer, the value of λ is varied between zero and one. The plot in Fig 4 shows non-interpolated average precision for different settings of λ. As depicted in plot, optimal value of λ is near 0.4, which indicates that our queries are of mixed length. According to [Zhai and Lafferty, 2001], for short queries optimal point is around 0.1 and for long queries optimal point is generally around 0.7. As long queries need more smoothing and less emphasis is placed on the relative weighting of terms.

Dirichlet Prior: The value of Dirichlet Prior μ is varied between 1 and 5000 with intervals of 500. The plot in Fig 4 illustrates the non-interpolated average precision for different settings of the prior sample size. As mentioned in [Zhai and Lafferty, 2001], the optimal priors μ vary from collection to collection and depends on query lengths. For our dataset and questions it is around 500.

Absolute Discounting: The value of δ is varied between zero and one. The plot in Fig 4 shows the non-interpolated average precision for different settings of δ. The optimal value of δ is near 0.3.

Overall the Dirichlet Prior performed best using μ of 500 and 500 retrieved passage documents. Then came Jelinek-Mercer, which is slightly better than Absolute Discounting. But the performance difference is not very significant. OKAPI performed slightly worse than Absolute Discounting while TF-IDF performed even worse.

Table 2 gives a comparison of the best run by each technique.

Method	Parameter	MAP
Dirichlet Prior	$\mu = 500$	0.127
Jelinek-Mercer	$\lambda = 0.4$	0.114
Absolute Discounting	$\delta = 0.3$	0.113
TFIDF	-	0.105
OKAPI	-	0.096

Table 2: Non-interpolated average precision for best run of each retrieval methods

5 Passage Re-Ranking

This section explains our language models, which are based on an interpolation smoothing scheme. Since Lemur is not flexible enough to implement such custom models, we shifted to our own language modeling toolkit. This toolkit is very flexible in generating custom language models. It uses perplexity to rank the documents. To check the similarity between the two toolkits, an experiment was carried out using Jelinek-Mercer smoothing technique to regenerate the results produced by Lemur. These results confirmed the validity of results generated by our toolkit.

5.1 Experimental Setup

Our experimental setup consists of document collections generated by experiments explained in previous sections. Following sections explain our datasets, experimental methods, and the system architecture.

Dataset
The query topics are the same as used for document retrieval (Section 3). The corpus C for evaluation is the AQUAINT collection that consists of documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. Also, we have the document collection dc and the passage collection pc obtained from our previous experiments. All these collections are stemmed and no stop word removal is performed.

Evaluation Methodology
Our toolkit uses perplexity to rank the documents. For the purpose of studying the behavior of an individual language model, we select a set of representative parameter values and examine the sensitivity of non-interpolated average precision MAP to the variation in these values. In question answering mean reciprocal rank (MRR) is also widely used. We checked the correlation of MRR and MAP on question answering tasks. For consistency with the document retrieval, we report MAP throughout the paper.

Experimental Methods
Our experimental methods are language modeling based. We have defined a number of language models using Jelinek-Mercer smoothing techniques.

5.2 System Architecture for Passage Re-Ranking

This section explains complete architecture of our experimental setup. Language models explained in this section utilize the vocabulary closed with the query and the value of interpolation parameter is varied between zero and one. The main difference between these models is the background collection.

Language Model I (pdclm)
This language model is defined as linear interpolation between unigram language models defined on passage and related document collection. Where each passage is taken from related retrieved passages (Section 4) and related document collection consists of 500 top ranked documents retrieved (Section 3), for a given query. As perplexity is given by the formula

$$PP = exp[-\sum_{w} f(w) \log P(w)]$$

where *f(w)* is the relative frequency of words in the query and the probability is

$$P(w) = (1 - \lambda)P_{ml}(w|p) + \lambda P(w|dc),$$

where P_{ml} is maximum likelihood of word *w* in passage *p* and *dc* is document collection.

Language Model II (ppclm)
This language model is similar to pdclm explained above with the related passage collection consisting of 500 top ranked passages retrieved as the background collection. For this language model the probability is

$$P(w) = (1 - \lambda)P_{ml}(w|p) + \lambda P(w|pc),$$

where *pc* is passage collection.

Language Model III (pdlm)
Here again the language model differs from pdclm in the background collection. The background collection is the single document from witch the passage was extracted i.e. the document containing the passage being ranked. For this language model, the probability for calculating the perplexity is

$$P(w) = (1 - \lambda)P_{ml}(w|p) + \lambda P(w|d),$$

where *d* is single document. Fig 5 explains complete setup to re-rank passages using this language model.

Re-ranker: It is responsible to re-rank the collection of 500 related passages per query. It utilizes standard tree and background tree containing statistical information required by language models.

Vocabulary: Our word list consists of all the words in the single document containing the passage being ranked, closed with words from query.

Standard Tree: It contains statistical information for given passage being ranked. We build one standard tree per passage.

Background Standard Tree: It consists of statistical information for the single document containing the passage being ranked. We build one standard tree per document.

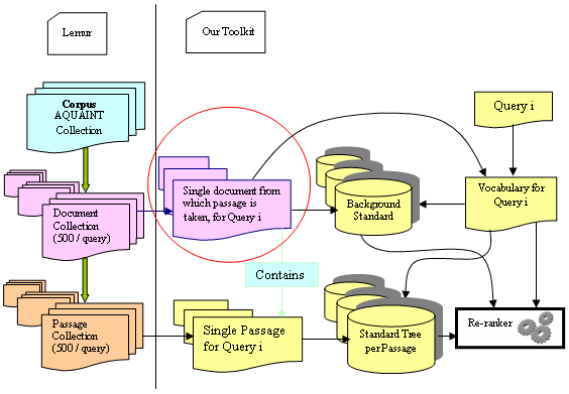

Figure 5: Dataset flow diagram for the pdlm language model.

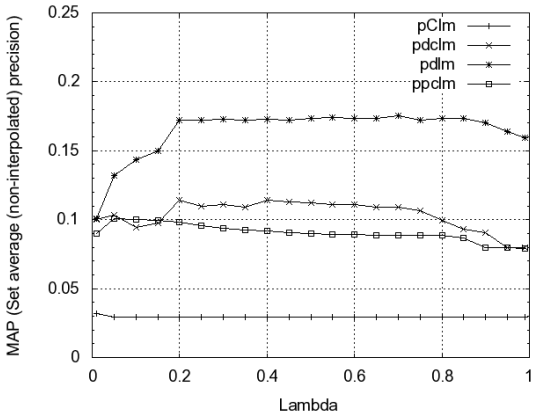

Figure 6: Plot of non-interpolated average precision against λ. Jelinek-Mercer b/w passage and different background collections with λ varying from 0.01 to 0.99.

Language Model IV (pClm)
For this language model the background collection is the complete corpus (AQUAINT document collection *C*). The probability for calculating the perplexity is

$$P(w) = (1 - \lambda)P_{ml}(w|p) + \lambda P(w|C),$$

5.3 Experimental Results

This section discusses the results of our experiments.

Language Model I (pdclm)
This language model is a reproduction of the language model with Jelinek-Mercer smoothing used in Section 4. It reproduces our previous results, which confirmed the validity of results generated by our language modeling toolkit. The plot in Fig 6 shows non-interpolated average precision for different settings of λ. It illustrates that the optimal value of λ is near 0.4.

Language Model II (ppclm)
Fig 6 shows the results using this language model by line with cross as points. The optimal value for λ is 0.05. According to [Zhai and Lafferty, 2001] small λ means more emphasis on relative term weighting, which means that the passage collection has a smaller role in ranking than passage itself. This might be due to small size of passages and variety in topics they discuss. With this language model we observe a 20% decrease over the baseline.

Method	Lambda	MAP
pdclm	0.40	0.114
ppclm	0.05	0.101
pdlm	0.70	0.176
pClm	0.01	0.032

Table 3: Non-interpolated average precisions for the best run of each language model. Passage re-ranking using the document language model for smoothing improves MAP by 39% over the best result from Lemur.

Language Model III (pdlm)

The line with squares in Fig 6 shows the results using this language model. The optimal value for λ is 0.70. The value of λ near middle of the parameter space suggests that both passage and document collection are equally important for ranking. The document is given a bit more importance than the passage, which is quite understandable as passages are of small size and sometimes they miss some related terms from query. With this language model we have more then 38% improvement over the baseline, which is quite a significant improvement. This is no surprise as both document and passage being used discuss the same topic. The related document size is relatively small compared to the document or passage collection, which also contributes to the improvement in results.

Language Model IV (pClm)

Fig 6 shows, using line with diamonds, the results using this language model. The optimal value of λ is 0.01. A small λ means more emphasis on relative term weighting, which means that corpus have nearly no role in ranking the passages. This is because of large size of corpus, with lots of irrelevant terms. It is also clear from Fig 6 that this language model performed worse than all our proposed models.

Table 3 display best results by each language model.

6 Conclusion and Future Work

We have studied the problem of language model smoothing in the context of passage retrieval for QA Systems and compared it with traditional models, including TF-IDF and OKAPI. We then examined three popular interpolation-based smoothing methods (Jelinek-Mercer, Dirichlet Prior, and Absolute Discounting), and evaluated them using the AQUAINT retrieval testing collection.

First we performed document retrieval. Our experiments showed that the Dirichlet Prior performed the best with prior of 1000. Then we carried out passage retrieval and observed that again the Dirichlet Prior performed the best with a prior of 500. With these experiments we established a baseline value. We have defined a number of language models based on the Jelinek-Mercer smoothing technique, and found out that interpolation between language model for passage and single document from which passage is extracted provided more then 38% improvement, which is quite significant for QA Systems.

Table 4 gives list of best runs for document retrieval, passage retrieval and re-ranking experiments. Our best performing language model can be used for real QA Systems. We have used one of the basic approaches to passage generation. One problem with our approach is that it does not take care of the topic shift within a passage. It also does not consider topics which spread over multiple passages. Other more sophisticated passaging techniques could further improve our proposed language model. The language models we have proposed and tested are all unigram models. As previous work depicts, higher order language models will improve retrieval performance.

Step	Method	Parameter	MAP
Document Retrieval	Dirichlet Prior	$\mu = 1000$	0.254
Passage Retrieval	Dirichlet Prior	$\mu = 500$	0.127
Re-ranking	pdlm	$\lambda = 0.70$	0.176

Table 4: Summary of results.

References

[Clarke *et al.*, 2000] 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.

[Corrada-Emmanuel *et al.*, 2003] A. Corrada-Emmanuel, W. Bruce Croft and Vanessa Murdock. Answer Passage Retrieval for Question Answering. In *CIIR Technical Report IR-283*, 2003.

[Berger and Lafferty, 1999] A. Berger and J. Lafferty. Information Retrieval as statistical translation. In *Proceedings of ACM SIGIR*, pp. 275-281, 1999.

[Zhang and Lee, 2003] A. Berger and J. Lafferty. A Language Modeling Approach to Passage Question Answering. In *Proceedings of the Text REtrieval Conference (TREC 2003)*, pp. 489, 2003.

[Tellex *et al.*, 2003] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes and Gregory Marton. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. In *Proceedings of the 26th annual ACM SIGIR conference*, pp. 41 - 47, 2003.

[Callan, 1994] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th annual ACM-SIGIR conference*, pp. 302-310, 1994.

[Liu and Croft, 2002] Xiaoyong Liu and W. Bruce Croft. Passage Retrieval Based On Language Models. In *CIKM conference*, pp. 375-382, 2002.

[Cai1 *et al.*, 2004] Deng Cai1, Shipeng Yu2, Ji-Rong Wen and Wei-Ying Ma. Block-based Web Search. In *Proceedings of the 27th annual ACM SIGIR conference*, pp. 456-463, 2004.

[Zhai and Lafferty, 2001] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th annual ACM SIGIR conference*, pp. 334-442, 2001.

[Agichtein and Gravano, 2000] E. Agichtein and L. Gravano. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 85-94, 2000.

[Clarke*et al.*, 2000] C. Clarke, G. Cormack and E. Tudhope. Relevance ranking for one to three term queries. In *Information Processing and Management*, pp. 291-311, 2000.