3

Comparing Improved Language Models for Sentence Retrieval in Question Answering

Andreas Merkel and Dietrich Klakow
Saarland University

Abstract

A retrieval system is a very important part in a question answering framework. It reduces the number of documents to be considered for finding an answer. For further refinement, the documents are split up into smaller chunks to deal with topic variability in larger documents. In our case, we divided the documents into single sentences. Then a language model based approach was used to re-rank the sentence collection.

For this purpose, we developed a new language model toolkit. It implements all standard language modeling techniques and is more flexible than other tools in terms of backing-off strategies, model combinations and design of the retrieval vocabulary. With the aid of this toolkit we conducted re-ranking experiments with standard language model based smoothing methods. On top of these algorithms we developed some new, improved models including dynamic stop word reduction and stemming. We also experimented with query expansion depending on the type of a query. On a TREC corpus, we demonstrate that our proposed approaches provide a performance superior to the standard methods. In terms of

Proceedings of the 17th Meeting of Computational Linguistics in the Netherlands
Edited by: Peter Dirix, Ineke Schuurman, Vincent Vandeghinste, and Frank Van Eynde.

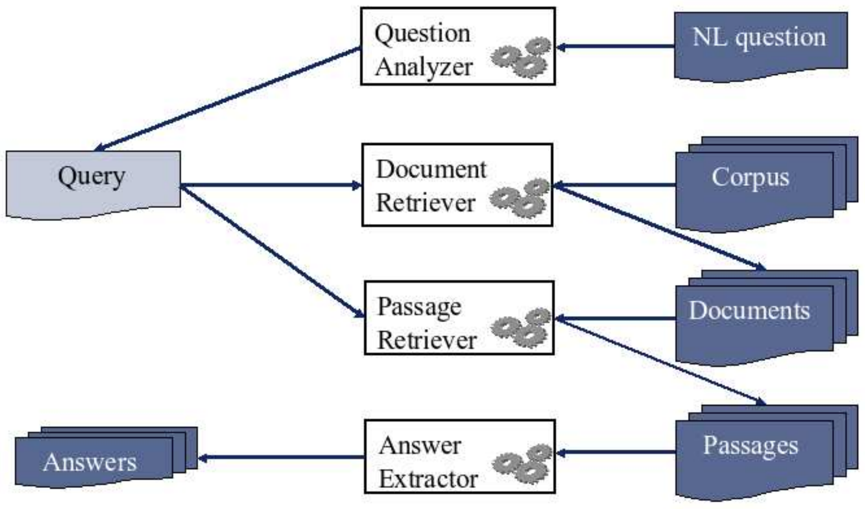

Figure 3.1: A general architecture for question answering systems

Mean Reciprocal Rank (MRR) we can prove a performance gain from 0.31 to 0.39.

3.1 Introduction

The major goal of a question answering (QA) system is to provide an accurate answer to a user question. Compared to a standard document retrieval framework, which just returns relevant documents to a query, a QA system has to respond with an adequate answer to a natural language question. Thus, the process of retrieving documents is just a part of a complex sequence. In order to provide the user with an answer, possible candidates have to be extracted from the documents. To simplify this procedure, the text is segmented into smaller passages and a further retrieval step is done. This process is called sentence retrieval, if the passage contains just one sentence.

In this paper, we describe an experimental setup for comparing different language models to improve sentence retrieval within a question answering context[1]. Figure 3.1 shows the general construction for a question answering system. It starts with the analysis of a natural language (NL) question (upper part). Generally, in this *Question Analyzer*, the expected answer type is determined, but it is also possible to make some other deep analyses like part of speech (POS) tagging or named entity recognition. The result is a processed query, which can be used for the following retrieval steps.

The next step is the document retrieval (*Document Retriever*). In a QA system, the retrieval framework is a very crucial part. It is used to decrease the number of

[1]This work was partially funded by the BMBF project SmartWeb under contract number 01 IMD01 M.

documents in a potential large corpus. This is done in order to reduce the search space in which a correct answer has to be found. It is necessary to reduce the search space because the following components may use long-lasting deep analysis algorithms which strongly depend on the size of the processed corpus. Therefore, it is important to process just the documents which seem relevant to a query to get answers within an appropriate period of time.

But within such a limited collection, there might still be large documents. Or within single documents some topic changes occur. If this is the case, again, the following components have to analyse more text than is necessary in order to find the correct answer. To overcome this problem, it is essential to further reduce the size of the collection. This can be done by splitting up the text segments into smaller chunks of passages[2]. After dividing the documents, a second retrieval step is necessary in order to re-rank the new passage collection (*Passage Retriever*) using the pre-processed query. By doing so, the corpus size and thus the search space is reduced again. One can say that the major goal of information retrieval in question answering is to achieve a high precision at a small collection of text segments (Corrada-Emmanuel et al. 2003).

In a final step, the passage collection is processed by the *Answer Extractor*. Here, the single passages are analyzed by computing the part of speech, the named entities and other linguistic features. Finally, the most probable answers are selected and returned by the system.

For our experimental setup we did not use the complete general architecture of a question answering system but just the upper part of fig. 3.1. So we skipped the extraction of the most relevant answers.

In the query construction, we used a language model driven method so as to find the expected answer type (Merkel and Klakow 2007) and some simple techniques to optimize the question for the following retrieval steps. The document retrieval was also done, using a language modeling approach.

Nevertheless, in this paper, we used a special case of passage retrieval where we directly split the documents into single sentences. In a next step, a language model (LM) based approach with unigram distributions was used to re-rank the text chunks. For these purposes, we developed a new language model toolkit. It implements all standard language modeling techniques, like linear interpolation and backing-off models. Its advantage is that it is more flexible than other tools in terms of model combinations, design of the retrieval vocabulary and the smoothing strategies. By means of this toolkit we conducted re-ranking experiments with standard language model based smoothing methods like Jelinek-Mercer linear interpolation, Bayesian smoothing with Dirichlet priors and absolute discounting as well as some new, improved models. We focused on investigating refinements which are easy to implement such as ignoring query words, dynamic stopword lists and stemming. We also experimented by modeling the expected answer type of a query into the LM approach.

To make our results comparable to current literature, we evaluated our algo-

[2]Popular methods to make passages and research into the effects on retrieval can be found in Clarke et al. (2000) and Tellex et al. (2003).

rithms on a news texts corpus from the Text REtrieval Conference (TREC) – the Aquaint corpus. Here, we demonstrate that our proposed algorithms outperform the standard methods in terms of mean reciprocal rank (MRR) by 25%. We can also show that we need to return fewer sentences to achieve equal or even better accuracy. So it is possible to say that we attained our goal, namely to reduce the search space for the following components in a QA framework.

The rest of paper is organized as follows: The next section presents some related work. Sect. 3.3 shows the language model based smoothing methods we used for our experiments. Sect. 3.4 presents the used datasets as well as the experiments we performed in order to achieve optimal results. Sect. 3.5 concludes the results.

3.2 Related Work

In the area of passage retrieval for question answering, one comes across a lot of secondary literature. For example, Clarke et al. (2000) introduces a passage retrieval system for the TREC *Question Answering track*. They use a question pre-processing, a passage retrieval and a passage post-processing step to select the top five text sections out of a set of documents. In the pre-processing step, the question is parsed and "selection rules" (patterns) are defined. Each text block for the passage retrieval algorithm can start and end with any query term. The score of such a passage is calculated by the text size and the number of occurring query terms. Then a new passage with a required length around the center point of the original passage is produced. Finally, the patterns are used to post-process the passage retrieval results.

Tellex et al. (2003) provides an overview of various state-of-the-art passage retrieval systems for question answering. They built a framework for the use of different document retrieval and passage making systems to compare the results. The text selection methods are re-implementations of famous TREC systems like MITRE, bm25, IBM and MultiText. The most important findings of their work is that boolean querying performs well for question answering, that the choice of the document retriever is very important and, that the best algorithms use density based scoring.

There is also some literature in the field of language model based passage retrieval for QA. In Zhang and Lee (2004) LM based question classification and a LM based passage retrieval approach is shown. To optimize their text selection, they first look at an initial set of relevant passages and construct a language model. Then, relevant web data is used to built a second language model. Finally, they mix the two models and include some further constraints like answer type and answer context information.

Another interesting approach is presented by Corrada-Emmanuel et al. (2003). In their paper, three methods to score relevant passages are shown. They compare the famous query likelihood, relevance modeling and a bigram answer model. In this model, the expected answer type is taken into account. Therefore, text selections are replaced by their named entity tag and an answer model is trained. Then three different methods for backing-off the bigram are used. They show that the bigram method provides a performance superior to the other approaches.

As mentioned above, sentence retrieval is just a special case of passage retrieval where the text selection has the size of one sentence. There is also some related work in the area of sentence retrieval for QA systems. One example is Murdock and Croft (2004). They understand the meaning of retrieving sentences as the translation of a user query to a (more or less complex) answer. With this idea, they suppose to overcome the problem of the shortness of sentences to compute a multinomial distribution. Their approach is based on the IBM Model 1 and is smoothed with the corresponding document in addition to the collection. They show a performance gain to the original query-likelihood scoring.
In Losada (2005) language model based approaches for sentence retrieval are compared. They define multinomial and multiple-Bernoulli distributions on top of the query-likelihood approach. Their motivation is the shortness of a sentence. In a multiple-Bernoulli framework, also the non-query terms are taken into account. So, they show a significantly performance increase compared to a multinomial approach.

An other application for sentence retrieval is the TREC Novelty track (Harman 2002). Here, the task is to reduce the amount of redundant and non-relevant information in a given document set. Normally, this is done in a two-step approach. The first part is to find the relevant sentences according to a query[3]. In a second part, those sentences are selected which contain novel information compared to the retrieved set in the first part. Larkey et al. (2003) and Allan et al. (2003) give some examples of how to build such a system. They use three different methods for extracting relevant sentences; a vector based approach using *tf-idf*, a version using the *Kullback-Leibler divergence (KLD)* and an approach using a *Two-Stage Smoothing* model. Because they do not find any significant differences between these methods[4], they decide to use the *tf-idf* approach. From their point of view, the selection of the relevant sentences is the major challenge, so they try to further improve the performance by using known techniques like query expansion, pseudo-relevance feedback and other features. But, again in contrast to our observations, just pseudo-feedback helps to improve the performance.
A major difference to the open-domain question answering is that in the Novelty track, a set of relevant documents is given. So, there is no need to find some relevant documents out of a large corpus first. Allan et al. (2003) also show the negative effects when using a real information retrieval system instead of a given document set.
A last significant difference is the kind of processing the retrieved data. In a question answering system, further steps are the extraction and selection of possible answers out of the sentences. This task is very hard and time-consuming, so it is necessary to keep the set of returned sentences as small as possible.

A partial implementation of the system can be found in Shen et al. (2006). There, a complete statistically-inspired QA system in context of the TREC 2006

[3]This is what we call *Sentence Retrieval*.
[4]In contrast to our experiments.

question answering track is developed.

Merkel and Klakow (2007) give a more specific description of the language model based query classification part we used in our experiments. This work mainly depicts the methods of how to obtain the expected answer types.

3.3 Methods

In this section, the general idea behind language model based information retrieval is presented. Furthermore, we describe the smoothing methods we used for our experiments in Sect. 3.4.3.

3.3.1 Language Models for Sentence Retrieval

First, we want to introduce the language model based approach proposed by Ponte and Croft in 1998 (Ponte and Croft 1998) as our information retrieval framework for sentence retrieval. They rank the user query using a query model, whereas a language model for each document is determined. Then the probability of producing the query with those models is calculated. Following Zhai and Lafferty (2001), applying Bayes rule results in

$$P(D|Q) \propto P(Q|D)P(D) \tag{3.1}$$

where $P(D)$ is the prior belief of a document and $P(Q|D)$ is the probability of the query given a document.

We act on the assumption that the prior $P(D)$ is a uniform distribution, so it is equal for all documents and therefore irrelevant for ranking the query. Thus, it will be ignored in further computations. The probability of $P(Q|D)$ is calculated by using language models. This conversion means that we just have to calculate the conditional probability of the user query and the document we intend to rank. This task seems easier than calculating $P(D|Q)$.

Formula (3.1) has a data sparsity problem. Generally, there isn't enough training data to compute language models for a complete query[5]. To overcome this problem we act on the assumption that all words in the query are independent. This independence assumption results in unigram language models as proposed in Zhai and Lafferty (2001):

$$P(Q|D) = \prod_{i=1}^{N} P(q_i|D) \tag{3.2}$$

whereas N is the number of terms in a query. In our approach the documents are sentences, so we used $P(q_i|S)$ as our experimental baseline, where S is the sentence we intend to score.

In the next sections we will describe how to calculate those probabilities. Because we use a maximum likelihood estimate to calculate $P(w|S)$, it is necessary to smooth them in order to avoid zero probabilities[6].

[5]Let's suppose that a query has 7 words in average. Then 7–gram language models have to be computed.
[6]See Zhai and Lafferty (2001) for further information.

3.3.2 Jelinek–Mercer smoothing

The Jelinek-Mercer smoothing method is just a linear interpolation between the maximum likelihood probability and a background collection model. It is defined by

$$P_\lambda(w|S) = (1-\lambda)\frac{c(w,S)}{\sum_w c(w,S)} + \lambda P(w|C) \tag{3.3}$$

where $c(w,S)$ is the count of word w in sentence S and λ is the smoothing parameter. $P(w|C)$ is the collection model. In our experiments the background collection always consists of the set containing all sentences.

3.3.3 Absolute Discounting

This smoothing method has its origin in the task of speech recognition. There, it is the most efficient and thus the most commonly used technique. But it was also introduced to the task of information retrieval by Zhai and Lafferty (2001). It results in

$$P_\delta(w|S) = \frac{\max\left(c(w,S)-\delta,0\right)}{\sum_w c(w,S)} + \frac{\delta B}{\sum_w c(w,S)}P(w|C) \tag{3.4}$$

whereas $c(w,S)$ are the frequencies of w in S and $P(w|C)$ is the collection model of all sentences. δ defines the smoothing parameter to redistribute some probability mass to unseen events. The parameter B counts how often $c(w,S)$ is larger than δ.

3.3.4 Bayesian smoothing with Dirichlet priors

Bayesian smoothing using Dirichlet priors is the approach which performs best according our question answering task in document retrieval as well as according our sentence retrieval framework. It is also described by Zhai and Lafferty (2001) and is defined by

$$P_\mu(w|S) = \frac{c(w,S)+\mu P(w|C)}{\sum_w c(w,S)+\mu} \tag{3.5}$$

where $c(w,S)$ is the frequency of observations of the word w in sentence S. μ is the smoothing parameter. Again, $P(w|C)$ is the collection model containing all sentences. A special case of this method is the *add-epsilon smoothing,* i.e. when a uniform collection model is used.

3.4 Experiments

In this section, we describe the dataset and the experimental setup we used for our experiments. Furthermore, we discuss the experimental results.

3.4.1 Dataset

As dataset for our experiments we used the *TREC 2004 QA collection*. It consists of the AQUAINT[7] document collection with more than one million[8] text documents from various news agencies (the Xinhua News Service (People's Republic of China), the New York Times News Service, and the Associated Press Worldstream News Service).

The question set for TREC 2004 consists of 351 questions, which are further divided into subsets. Each subset has a unique topic and a set of "factoid", "list" and "other" question. For example, a typical "factoid" question is "When was James Dean born?" whereas a "list" question would be "What movies did James Dean appear in?". The task of the "other" question is mainly to find as many different information concerning the topic as possible.

As evaluation metrics for the results, the Mean Reciprocal Rank (MRR) and the accuracy of the system was used. In this context, accuracy means the percentage of answerable questions using a specific number of returned sentences. For testing the parameters in the query construction, we used the Mean Average Precision (MAP).

3.4.2 Experimental Setup

For efficiency reasons we chose a three-step approach for our experiments. First, the user question was analyzed. Therefore, we used the approach described in Merkel and Klakow (2007) to extract the expected answer type. It specifies a language model based query classification, using a simple Bayes classifier as paradigm. The taxonomy of the classifier takes 6 coarse and 50 fine grained classes into account.
In addition to this, some simple methods were used to further optimize the query for the following retrieval task.

In a second step, the *Lemur Toolkit for Language Modeling and Information Retrieval*[9] was used to carry out a language model based document retrieval. As suggested in Hussain et al. (2006), we performed Bayesian smoothing with Dirichlet priors. We fetched the top 50 relevant documents because Shen et al. (2006) showed that this number is sufficient to answer about 90% of questions. After the extraction, we split them up into sentences using the sentence boundary detection algorithm provided by LingPipe[10]. We also used larger passages (Hussain et al. (2006)), but the sentence-based approach is much more efficient.

The third step was the re-ranking of sentences using the language model based methods described in Sect. 3.3. For these purposes a new language modeling toolkit was developed by our chair[11]. It implements all standard language modeling techniques and is more flexible than other tools in terms of backing-off strate-

[7] `http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/`
[8] 1,033,461 documents.
[9] `http://www.lemurproject.org/`
[10] `http://www.alias-i.com/lingpipe/`
[11] *LSVLM*.

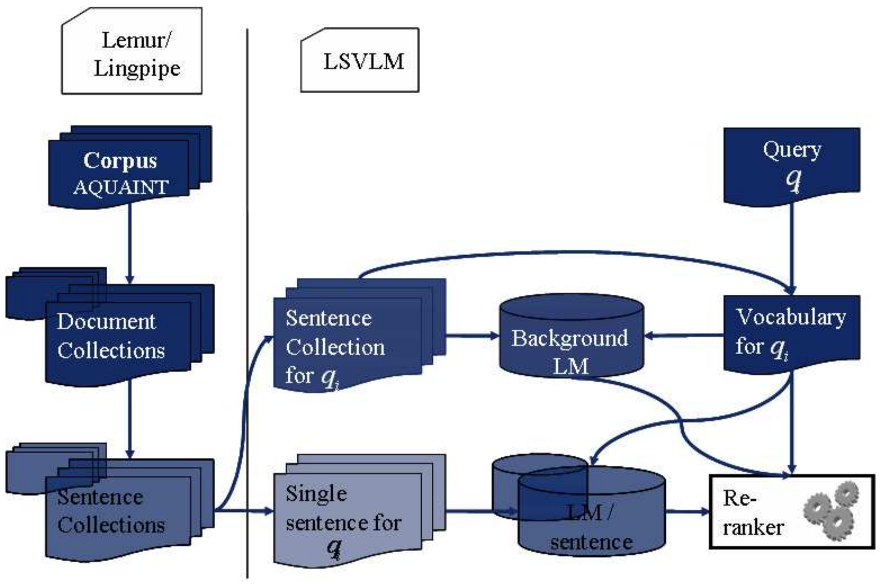

Figure 3.2: The sentence retrieval architecture for our experiments

gies, model combinations and design of the retrieval vocabulary.

Figure 3.2 shows the architecture of the sentence retrieval experiments we made. On the left-hand side, the pre-processing steps with standard software are shown. On the top one finds the Aquaint corpus we described in Sect. 3.4.1. Then the document collections, we gain by using Lemur, can be seen. On the bottom one finds the sentence collections we receive by using the LingPipe toolkit.

On the right-hand side, we show the experimental setup for the *LSVLM* framework. In the middle the background model for each experiment is presented. It consists of the complete sentence collection for a given user query q_i. Out of this collection, a background language model is build. As vocabulary for this model we use the union of the vocabulary build from the user query q_i and the corresponding sentence collection. So, the vocabulary is closed over the query.
On the bottom left, the single sentences for a query q_i can be seen. These are the sentences we want to re-score in our experiments. Then, again language models are created for each individual sentence by using the closed vocabulary.
And finally, the two language models are used to calculate a new score. Because of the flexibility of our toolkit, it is possible to easily change the used smoothing algorithms and parameters to get the optimal setting.

# included topics	Mean Average Precision
0	0.0900
1	0.2440
2	0.2995
3	0.2997
4	0.2876
6	0.2799

Table 3.1: Number of included topics and corresponding MAP for TREC 2004 dataset

In our sentence retrieval experiments, we used TF–IDF and OKAPI[12] as standard baseline approaches and linear interpolation (Jelinek-Mercer), absolute discounting and Dirichlet priors as language model based smoothing algorithms (see Sect. 3.3).
The smoothing parameters for the different methods were experimentally defined on the TREC 2003 dataset. For absolute discounting, we took the discounting parameter $\delta = 0.1$, for linear interpolation the smoothing parameter was set to $\lambda = 0.8$ and for Dirichlet prior we set $\mu = 100$.

3.4.3 Results

In this section, we discuss the results we achieved by using the query construction, the document retrieval and the optimized sentence retrieval steps.

3.4.3.1 Query Construction and Document Retrieval

As already mentioned in Sect. 3.4.2, we first analyzed the user query by extracting the expected answer type[13]. This answer type is used in a later step to optimize the language models in the sentence retrieval step.

In addition to this approach, we used further methods to optimize the query for document and sentence retrieval. In a first step, the topic of the query[14] was included for multiple times. This inclusion was done because, within our language model approach, the repeating of a specific term for multiple times results in a higher score for that term. A higher score means that the included term gets greater importance in that context.
Table 3.1 shows the impact of including the topic on the document retrieval. The performance increases until the topic was added three times. This means, if we add the topic too often, it gets too much weight and other possible relevant keywords are scored too lowly. This would result in a worse retrieval performance. Due to

[12]See `http://www.lemurproject.org`
[13]Results can be found in Merkel and Klakow (2007).
[14]See Sect. 3.4.2 for a definition of “topic”.

the fact that there is just a very small performance gain between adding the topic twice or three times, we decided to include the topic in our experiments only twice.

The last step in the query construction was the subtraction of the query word. In general, this term has no positive effects on the retrieval system and can therefore be ignored. Here, the same argumentation holds as for including the topic for multiple time. By removing the query word, this score will be zero and other, possibly more relevant terms get a higher score.

The effects of these methods on the sentence retrieval framework can be found in Sect. 3.4.3.3.

As mentioned above, the *LEMUR* toolkit was used to perform document retrieval. Therefore, the queries as well as the AQUAINT corpus were stemmed and no stop-words were removed. Then we chose a language model based approach to retrieve the documents. As smoothing method, we used Bayesian smoothing with Dirichlet priors because Hussain et al. (2006) illustrated that this approach performs best for this task[15]. They also suggests an optimal smoothing parameter for this question set which we also used for our experimental setup.
After doing the retrieval, the 50 most relevant documents were fetched and split up into sentences.

3.4.3.2 Baseline Experiments

This section describes the baseline experiments we conducted before starting our optimization approaches. Figure 3.3 shows the results of those experiments. It presents on the x-axis the number of returned sentences by the system on a logarithmic scale. On the y-axis the accuracy of the system is shown. For example, an accuracy of 0.5 means that 50% of the queries are answerable by the system.

For the standard TF–IDF and OKAPI baseline experiments we used the *LEMUR* toolkit. The figure shows that the TF–IDF performs better than OKAPI regarding this task. Both approaches were not optimized for these experiments.

In a next step, we used our *LSVLM* toolkit to conduct the baseline experiments with the three standard language model based smoothing approaches (as described in Sect. 3.4.2).

For a small number of returned sentences (1–50), the linear interpolation (Jelinek–Mercer) and the absolute discounting smoothing perform comparably bad. In this part the Bayesian smoothing with Dirichlet priors obviously performs better.
In the last segment (50–100 sentences) the Dirichlet prior approach performs somewhat worse than absolute discounting. The best smoothing method for this part is the Jelinek–Mercer interpolation. But this performance gain is not visibly significant.

But the figure also shows that all baseline language model based approaches perform better than the standard TF–IDF and OKAPI methods for this task by large margin.

[15]They show that it even provides a performance superior to standard approaches like TF–IDF and OKAPI.

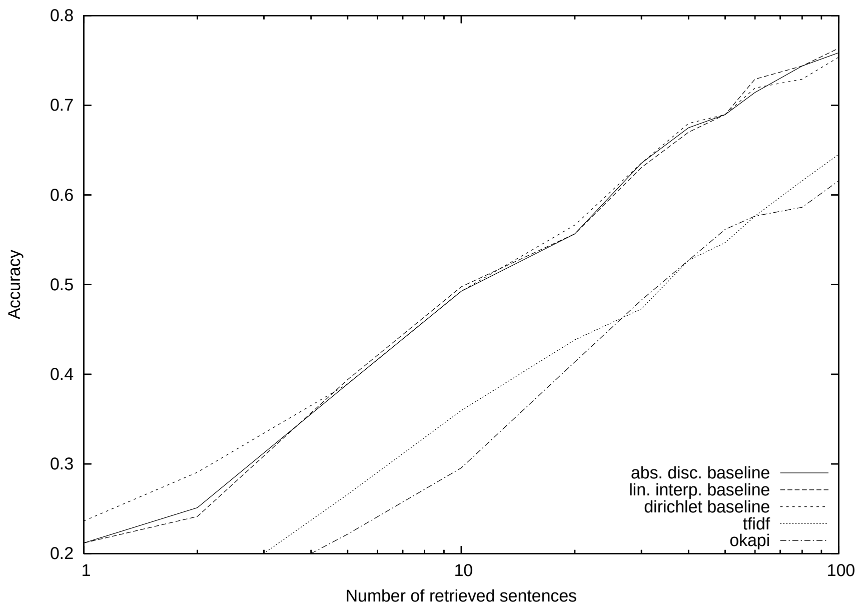

Figure 3.3: Number of retrieved sentences vs. accuracy for baseline experiments

As already mentioned in Sect. 3.1, in a question answering system we are most interested in getting a high accuracy at a small number of returned sentences. That means, the following modules need to process just smaller sets of sentences to reach the same level of accuracy. Thus, the Bayesian smoothing with Dirichlet priors was chosen as a optimization baseline for further experiments.

3.4.3.3 Improved Smoothing Methods

Figure 3.4 shows the results of the experiments we carried out with optimized language models for the question answering task. Again, on the axis of abscissae the number of returned sentences is plotted on a logarithmic scale, whereas on the ordinate the accuracy of the system is shown (as described in Sect. 3.4.1).

For better comparison between the improvements of the optimization steps, the Dirichlet prior smoothing method is shown as baseline (curve (1)). The other lines show the performance gain of each individual method we added to the baseline. Each new experiment is based on the previous optimization method.

Our first approach is already discussed in Sect. 3.4.3.1. It is the simple removal of the query word and therefore belongs to the query construction step. Figure 3.4 shows the resulting effects on the system. The new curve (2) provides a performance superior to the Dirichlet baseline.

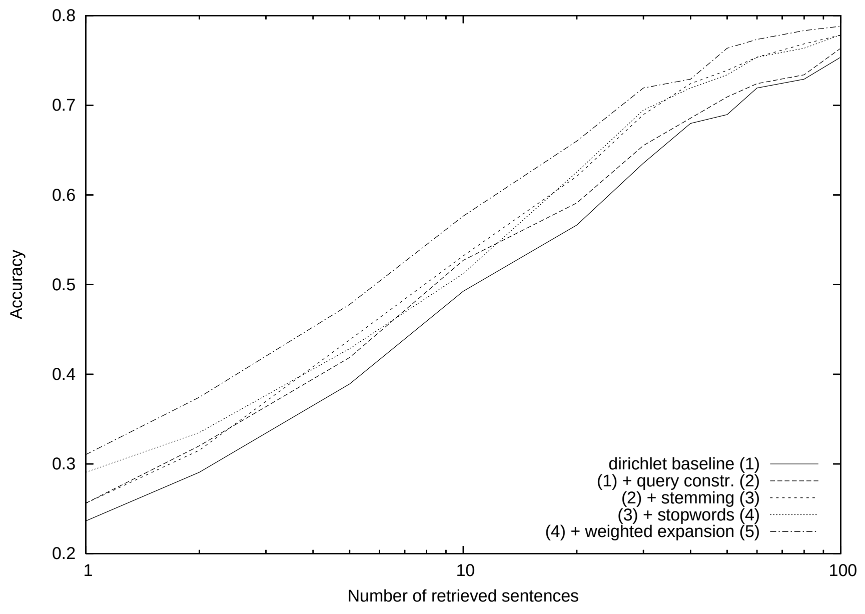

Figure 3.4: Number of retrieved sentences vs. accuracy for optimized smoothing methods

In a second step, we added the Porter stemmer[16] to our experiments. The results are shown in curve (3). As described in relevant literature, this addition also results in a further advance of system accuracy of this specific kind of task.

As a next optimization criterion we used a dynamic stopword list. It was created by selecting the four most commonly used terms of the complete sentence collection. However, those terms were not removed as usual for stop-words but they got just a smaller weight in the language model. This re-weighting is based on the same findings we already discussed in Sect. 3.4.3.1. The result in Fig. 3.4 shows in curve (4) a small performance gain, when looking at a very small number of returned sentences. Besides, the accuracy is nearly equal to the previous step.

For the last optimization experiment, the expected answer type of an user query we gained in the query construction, was used to expand the language models with this additional information (see Sect. 3.4.3.1). This was done by expanding the query and a sentence in dependency of the extracted question type. Thereby, sentences, which match the query type, are ranked higher.

This means, for example, if the expected answer type is *Date*, the term *DATE* is added to the question. Then patterns are used to identify dates expressions in a sentence. If such a date expression also occurs in a sentence, it is expanded with the *DATE* term as well. After this step, the additional terms are weighted and thus the language model based approach gives a higher rank to sentences which match

[16]`http://www.tartarus.org/˜martin/PorterStemmer/`

Distribution	MRR
OKAPI	0.16
TF–IDF	0.18
Jelinek–Mercer	0.29
Absolute Discounting	0.29
Dirichlet Baseline	0.31
Dirichlet Combined	0.39

Table 3.2: Mean Reciprocal Rank of baseline and optimized experiments

the corresponding question.

The resulting effects of this last optimization step is also shown in Fig. 3.4. Here, curve (5) demonstrates the improvement of performance by adding the weighted expansion. The distribution outperforms all other combined methods by a large margin.

Table 3.2 shows the MRR of the baseline experiments and combination of all optimization steps. Standard OKAPI and TF–IDF achieved the worst MRR. The Jelinek–Mercer interpolation and absolute discounting baseline perform better with a MRR of 0.29. We found out that the Dirichlet prior baseline again performs a little bit better with a MRR of 0.31. This was the reason for the fact why we developed the improved language models on top of this distribution. The table also shows that the combination of all optimization steps (Dirichlet Combined) performs best with a MRR of 0.39. This means that there is an improvement of more than 25% compared to the Dirichlet baseline and, that there is an improvement of more than 34% compared to the other LM based experiments.

3.5 Conclusion

In this paper, we showed a language model based framework to perform improved sentence retrieval in a question answering context. The major goal was to improve the accuracy of the system in order to return just a smaller number of relevant sentences. This reduces the search space of the following components in a QA system. Because these components are typically deep-analysis approaches which strongly depend on the size of processing documents, such a step is necessary.

For this purpose, we first analyzed the user query by extracting the expected answer type and doing some other simple text manipulations.
After a language model based document retrieval step, we split up the documents into smaller text passages in the size of sentences.

Then, the *LSVLM* toolkit, a language model based framework we developed at our department, was introduced. With this toolkit, we were able to conduct sentence retrieval experiments in a more flexible way than with other state-of-the-art information retrieval frameworks. We conduct baseline experiments with standard TF–IDF and OKAPI as well as with language model based smoothing methods like Jelinek–Mercer interpolation, Bayesian smoothing with Dirichlet priors and absolute discounting.

We proved that Dirichlet priors baseline performs best for our task, so we developed our optimization steps on top of this approach.
In several experiments we illustrated that using query word removal, dynamic stopword list weighting and stemming results in a performance gain. In the last experiment, we modeled the expected answer type of a user query into the used language models. This approach performs better than the LM baselines by at least 25%.

We also proved that we need to return fewer sentences in order to achieve equal or even better performance in terms of system accuracy. So we attained our goal to reduce the search space for the following components in a QA framework.

References

Allan, J., Wade C., and Bolivar, A. (2003). Retrieval and Novelty Detection at the Sentence Level, *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR) 2003*, Toronto, 2003.

Clarke, C.L.A., Cormack, G.V., Kisman, D.I.E., and Lyman, T.R. (2000). Question Answering by Passage Selection (MultiText experiments for TREC 9), *in The 9th Text REtrieval Conference (TREC-9)*, Gaithersburg: NIST, 2000.

Corrada-Emmanuel, A., Croft, W.B., and Murdock, V. (2003). Answer Passage Retrieval for Question Answering, *in CIIR Technical Report*, Amherst, 2003.

Harman, D. (2002). Overview of the TREC 2002 Novelty Track, *in The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2002, pp. 17–28.

Hussain, M., Merkel, A., and Klakow, D.(2006). Dedicated Backing-Off Distributions for Language Model Based Passage Retrieval, *in Hildesheimer Informatik-Berichte, LWA 2006*, Hildesheim, 2006.

Larkey, L.S., Allan, J., Connell, M.E., Bolivar, A., and Wade, C. (2003). UMass at TREC 2002: Cross Language and Novelty Tracks, *in The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2003.

Losada, D.E. (2005). Language Modeling for Sentence Retrieval: A Comparison between Multiple-Bernoulli Models and Multinomial Models, *in Information Retrieval and Theory Workshop*, Glasgow, 2005.

Merkel, A. and Klakow D. (2007). Language Model Based Query Classification, *to appear in Proceedings of 29th European Conference on Information Retrieval (ECIR)*, Rome, 2007.

Murdock, V. and Croft, W.B. (2004). Simple Translation Model for Sentence Retrieval in Factoid Question Answering, *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR) 2004*, Sheffield, 2004.

Ponte, J.M. and Croft, B. (1998). A Language Modeling Approach to Information Retrieval, *in Proceedings of the Special Interest Group on Information*

Retrieval (SIGIR) 1998, Melbourne, 1998.

Shen, D., Leidner, J.L., Merkel, A., Klakow, D. (2006). The *Alyssa* System at TREC 2006: A Statistically-Inspired Question Answering System, *in The Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Tellex, S., Katz, B., Lin, J., Fernandes, A., and Marton G. (2003). Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering, *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR) 2003*, Toronto, 2003.

Zhai, C. and Lafferty J. (2001). A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval, *in Proceedings of the Special Interest Group on Information Retrieval (SIGIR) 2001*, New Orleans, 2001.

Zhang, D. and Lee, W.S. (2004). A Language Modeling Approach to Passage Question Answering, *in NIST Special Publication 500–255: The Twelfth Text REtrieval Conference (TREC 2003)*, Gaithersburg: NIST, 2004.