Language Modeling Approaches for Enterprise Tasks

Leif Azzopardi* **Krisztian Balog** **Maarten de Rijke**

ISLA, University of Amsterdam
http://ilps.science.uva.nl/

Abstract: We describe our participation in the TREC 2005 Enterprise track. We provide a detailed account of the ideas underlying our language modeling approaches to these tasks, report on our results, and give a summary of our findings so far.

1 Introduction

Our aim for the TREC 2005 Enterprise track was to adapt our existing language modeling framework to the specific needs of each task. A key goal was to incorporate, and make use of, the structure and structured content housed within the W3C data used in the track.

Using generative language models, we tailored the framework to address the particular needs of the three sub tasks: Email Discussion Search, Known Email Search, and Expert Finding. These tasks were executed on the enterprise collection which contained six different types of web pages created from a crawl of the W3C website. These were lists (email forum), dev, www, esw, other, and people. The former two tasks used only the email forum where structure was a main theme of our research. Here, we examined the link structure generated by replies for the discussion search (Section 2), whilst we considered the internal structure of an email for known item searching (Section 3). For the expert finding task our focus was on associating a document with a candidate expert to build candidate models (Section 4).

2 Email Discussion Search

The goal of this task was to retrieve emails which contained a discussion about the query topic, where highly relevant documents would introduce a new point to the discussion (such as pro or con given the topic). Consequently, for this task only the email forum (lists) documents were considered. This subset contains pages which are not only emails, but pages for the navigation of the forums as well. The collection comprised of 198,275 documents of which approximately 174,413 were emails to the forum lists.[1] Each email page contains links to the other emails that are related to it (i.e., the response(s) an email attracts, the email that was responded to, and next/previous emails in the listing). Of these emails only 75,422 emails had attracted a response.[2] This left 99,991 emails without any replies.

2.1 Discussions

We assumed a 'discussion thread' consisted of a set of emails, linked by replies, which formed a graph of emails. The number of discussion threads within the 75,422 emails was approximately 19,917, where the average number of emails in a discussion thread was 3.8.

Within a discussion thread, there were two main quantitative characteristics of interest: *breadth*, indicating the number of replies an email has directly received, and *depth*, indicating the number of consecutive replies. We conducted informal interviews with two email forum users, who regularly use technical forums, and asked them about how they used the email forums and about the shape of the discussion threads. The main points ascertained were as follows. They rarely ever searched for discussions, when they did, they would favor the use of navigational methods as opposed to search facilities. When replying to an email, it was important to respond to a point in that email, and that the email should only respond to that point. Further, a separate email should be sent in response to the different points in that email. Hence, an email attracting multiple replies would probably discuss multiple points, whilst consecutive replies would probably discuss one point in detail.

To follow up this intuition, we examined ten discussion graphs (about 50 emails in total), five of which contained a breadth of at least three and five with a depth of at least three. From this set of graphs, consecutive replies tended to discuss a point in detail, whilst multiple replies would usually present different points, but would sometimes include emails referring to other replies. This suggested that long chains of emails would be indicative of arguments for and against the topic of discussion, and may have been useful for retrieval. We endeavored to encode some of these findings within our

*Current affiliation: Department of Computer and Information Sciences, University of Strathclyde.

[1] This is an estimate of the number of emails in the collection and maybe slightly inaccurate due to parsing errors.

[2] Note that "maybe replies" were treated as replies.

retrieval strategy, the language modeling framework.

2.2 Language Model

The standard language modeling approach computes the probability of a query q being generated from a document model θ_d on behalf of the document d as follows:

$$p(q|\theta_d) = \prod_{t \in q} \left\{(1-\lambda)p(t|d) + \lambda p(t)\right\}^{n(t,q)}, \quad (1)$$

where $p(t|d)$ is the maximum likelihood estimate of term t in document d, $p(t)$ is the unconditional probability of t (also determined using the maximum likelihood estimate), $n(t,q)$ is the number of times term t occurs in query q, and λ is the smoothing parameter. If λ is set to $\frac{\beta}{n(d)+\beta}$, where $n(d)$ is the size of the document, Bayes Smoothing with a Dirichlet prior of the document model is obtained (instead of Jelinek-Mercer Smoothing) [3]. Ranking according to the joint probability of a query and document $p(q,d)$, involves the multiplication of the document prior $p(d)$ to both sides of the equation such that $p(q,d) = p(q|\theta_d)p(d)$. This represents a natural extension to the framework for encoding external evidence.

2.3 Discussion Runs

For our discussion run submissions, we applied two priors; one to filter our non-discussion emails, the other to bias towards larger/longer threads of discussion. For this, the lists collection was indexed using LEMUR. No stemming was applied but standard stop words were removed. We developed a set of five training discussion topics with 54 highly relevant emails and 29 relevant documents, after assessing a total of 169 documents. These topics were used to select our baseline run, to which we applied two different document priors that adjusted the scores according to our intuitions. For our baseline, we examined a host of different parameter settings with both Jelinek Mercer Smoothing and Bayes Smoothing, but Bayes smoothing was found to perform the best when $\beta = 350$. The two priors were then applied:

Document Type Filter. We removed all messages which were not identified as an email in the collection. This can be considered as a document prior where $p(d) = k > 0$ if the document is an email, else $p(d) = 0$. If the document contained the structured fields subject, author and date then it was considered an email.

Thread Size Prior. The probability of a document, $p(d)$, was proportional to the size of the graph from which that email came. Such that:

$$p(d) = \frac{g(d) + \alpha}{\sum_{d'}(g(d') + \alpha)}, \quad (2)$$

where $g(d)$ is the size of the graph given d and α is a smoothing parameter to adjust the influence of the prior (known as Laplace smoothing). This encoded our intuition that discussions are a group of messages, and that an email in a discussion is more likely to be relevant than a email that is not. This prior was applied to the top 1000 documents retrieved documents to re-rank the result set.

We also considered augmentation of the ranked list, instead of re-ranking. This approach re-structured the results such that if an email appeared in the ranked list, then all related emails in its graph were given the same rank. This was to provide the user with a coherent view of the result list (i.e., grouped by discussion thread). Unfortunately, this run was not successfully submitted due to time constraints. However, given the evaluation scheme used, we would not have expected the results to fare significantly better, because the evaluation is based on a ranked list, and not the representation presented to the user (i.e., a graph).

2.4 Summary of Runs and Results

The following runs were submitted and the results are displayed in Table 1. All submitted runs were automatic and only the query field of the topic was used.

`ToNsBs350` Baseline run using Bayes Smoothing ($\beta = 350$).

`ToNsBs350F` Same as ToNsBs350, but with Document Type Filter.

`ToNsBs350FT` Same as ToNsBs350F, but with the Thread Size Prior ($\alpha = 1$).

`ToNsBs350FT5` Same as ToNsBs350F, but with the Thread Size Prior ($\alpha = 5$).

Run identifier	MAP	p@10	p@20	p@100
`ToNsBs350`	0.2907	0.4441	0.4034	0.2047
`ToNsBs350F`	**0.3518**	**0.5407**	**0.4449**	**0.2147**
`ToNsBs350FT`	0.1947	0.3559	0.2873	0.1442
`ToNsBs350FT5`	0.1988	0.3610	0.2975	0.1480

Table 1: Results for Discussion Search

In Table 1, the first column displays the run identifier, the second reports the mean average precision (MAP), then the following three columns display the precision at 10, 20 and 100. From these results, we can see that the influence of the filter increased the MAP by about 6% over the baseline run. However, applying the filter resulted in a loss of 21 relevant documents. This was due to either non-emails judged as relevant or documents not being parsed correctly. The application of the thread size prior introduced too much bias and resulted in a massive loss of MAP. Further work is required to examine the influence of the prior on performance.

3 Email Known-Item Search

The goal of this task was to find the known (relevant) email given the query topic. The intuition that motivated our research was that users would pose queries for these known emails, based on what they remembered about the known email. We assumed that such query terms would invariably be the most salient features of the email. Our retrieval strategy for the known-item email search used these features and consisted of two components. First, we automatically inferred the structure of a query (with respect to the email's structure) similar to [1]. Then, we execute the structured query on a fielded language model to utilize this structure in the retrieval process. So, instead of treating each email as a whole document, we broken the email into four structured fields. These were: author, date, subject and body of the email. All other text was disregarded. These fields were chosen because they represented the key fields from which query terms appeared to be generated (or, recalled).

3.1 Automatic Querying Structuring

The query was structured by classifying each query term t according to the probability of the field x given t (i.e., $p(t|x)$). This was evaluated by applying Bayes theorem and then using a generative model, such that

$$p(x|t) = \frac{p(t|x)p(x)}{\sum_{x'} p(t|x')p(x')}, \tag{3}$$

where $p(t|x)$ is the probability of t given x, which was estimated with Laplace estimator and proportional to the count of the number of times t occurred in x plus the Laplace constant ($\alpha = 0.00001$). Query terms were assigned to the field x, if $p(x|t) > \delta$, where δ was a tuning parameter of the system. This was set to $\delta = 0.1$ after training the system on the 25 known-item training topics and selecting δ with respect to the mean reciprocal rank of the fielded language model (Section 3.2). Each automatically structured query consisted of the set of fields, represented by q_x, which contained the assigned terms.

3.2 Fielded Language Model

The fielded language model is a simple extension of the standard language modeling approach described in Section 2.2. It treats each field of an email document as an independent source of evidence, from which each of the fields in the query are generated. Formally, this can be represented as

$$p(q|d) = \prod_x p(q_x|\theta_d^x), \tag{4}$$

where $p(q_x|\theta_d^x)$ is the probability of the query field q_x being generated from the model of the document field θ_d^x. This probability is computed as above for standard documents, but for each of the four fields instead.

We also considered an alternative approach, where we assumed that the sources of evidence were linearly independent and weighted by $p(x)$, such that by marginalizing over x the query likelihood could be expressed as:

$$p(q|d) = \sum_x p(x)p(q_x|\theta_d^x), \tag{5}$$

where $p(x)$ denotes the importance of the query field in the document.

3.3 Known-Item Runs

Our experiments examined the hypothesis that automatically inferred queries could be used to improve retrieval performance over unstructured queries (as in [1]).

The email fields selected were indexed separately in LEMUR, with Porter stemming applied and standard stop words removed. The 125 known-item queries were processed in a similar fashion. Our first submission was a baseline run, that used the standard language modeling approach on the entire email document using the original unstructured query and then we submitted four further runs using the field language model. Two runs used the query likelihood as shown in Eq. 4 and Eq. 5. However, we were concerned that the differences in length between query fields affected retrieval performance, and thus tried two runs where the normalized query likelihood was used [2]. This is equivalent to computing the odds ratio of the query being generated by the document versus the query being generated from the collection). For the model defined in Eq. 5, the prior $p(x)$ was set on a query by query basis. $p(x)$ was proportional to the number of query terms that were assigned to that field x, which we assumed would correlate to its importance.

The 25 training topics were used to tune the free model parameters and compare smoothing methods. We found Jelinek Mercer smoothing tended to give the best performance and subsequently used this form of smoothing for all runs and models.

3.4 Summary of Runs and Results

The following runs were submitted:

`qdFlat` Baseline run using language modeling approach with Jelinek Mercer smoothing ($\lambda = 0.1$).

`qdC` Automatically structured queries ($\delta = 0.1$), using the model in Eq. 4 with Jelinek Mercer smoothing ($\lambda = 0.5$ for all fields).

`qdWcEst` Same as qdC, but using the model in Eq. 5.

`OddsC` Same as qdC, but normalized.

`OddsWcEst` Same as qdWcEst, but normalized.

Run identifier	MRR	S@10	S@100	F@100
qdFlat	0.494	75.2%	91.2%	8.8%
qdC	0.423	56.8%	78.4%	21.6%
qdWcEst	**0.579**	**79.2%**	**92.0%**	**8.0%**
OddsC	0.423	56.8%	78.4%	21.6%
OddsWcEst	0.547	56.8%	89.6%	10.4%

Table 2: Results for Known-Item Finding

The results for the known-item finding subtask are shown in Table 2. The second column gives the mean reciprocal rank (MRR) score. The third and fourth columns report the percentage of topics for which the known-item was found in the top 10 and 100 documents, respectively. The last column reports the percentage of topics where no known-item was found in top 100 documents (F@100).

From the result table we see that, using the model in Eq. 5 (runs qdWcEst and OddsWcEst), we were able to obtain an improvement over our baseline run, with a sizeable increase in the MRR. Once we obtained the corresponding set of known-items, we tagged the query terms according to the fields in the known emails. We found that the accuracy of our automatic query structuring procedure was just over 50%, whilst if we had simply assumed all terms were from the subject accuracy would have been just under 50%. During the training phase, better classification accuracy led to substantially improvements over the baseline regardless of the type of fielded model. However, here the ambiguous nature of the queries seriously degraded the performance of our retrieval models, as the structure could not be reliably inferred.

4 Expert Search

The Expert Search task presents the following scenario: Given the document repositories of the organization, find the experts in a particular topic, field or area. Our approach employs language modeling, information retrieval, and name entity recognition techniques. Our results indicate that the latter is especially important for the task.

Our approach focuses on building representations of candidate experts from the corpus, and then identifying the set of actual experts for a given topic. To achieve this we apply the language modeling approach to the expert search problem. Each candidate is represented by the documents that are found to be the most relevant given the topic and the candidate is associated with. When a query is issued, we select the subset of the collection, containing documents, found to be the most relevant given the query. To obtain these cut-offs we apply information retrieval techniques over the document set and against the topic as a query. Then, the candidates are ranked according to the probability of the query being generated by the candidate model.

The main research problem within this work is to find the associations between documents and candidates.

4.1 Modeling

Our method is a direct application of standard language modeling techniques, where we infer a candidate model θ_{ca} for each candidate ca, such that the probability of a term given the candidate model is $p(t|\theta_{ca})$. Using this model, we can then estimate the probability of a query by taking the product across terms in the query.

$$p(q|\theta_{ca}) = \prod_{t \in q} p(t|\theta_{ca})^{n(t,q)}. \quad (6)$$

Here, the standard term independence assumption is made. The candidate model is constructed by using a mixture model:

$$p(t|\theta_{ca}) = (1-\lambda) \sum_d p(t|d)p(d|ca) + \lambda p(t), \quad (7)$$

where $p(t)$ is the maximum likelihood estimate of the unconditional probability of the term occurring in the collection. The final estimation of the probability of a query given the candidate model is:

$$p(q|\theta_{ca}) = \prod_{t \in q} \left\{(1-\lambda)(\sum_{d \in S} p(t|d)p(d|ca)) + \lambda p(t)\right\}^{n(t,q)}, \quad (8)$$

where S is a subset of documents that are found to be the most relevant given the query q.

4.2 Candidate Document Associations

As pointed out before, the W3C corpus is a heterogeneous document repository containing a mixture of different document types (technical reports, emails, web pages, etc). A document d in this collection, is assumed to be associated with a candidate ca, if there is a non-zero association $a(d, ca) > 0$. The forming of these associations is vital to the performance of our methods and overall performance. Here we introduce four different methods that we used for associating documents with candidates.

Extracting candidates is a special named entity recognition task where the list of possible candidates—with names and e-mail addresses—are given.

Extract Candidates by Name. Identification based on the candidates' name might be the most natural approach. In spite of the apparent simplicity, one has to face different challenges when solving this task. Some cases when a simple *exact-matching* test on the candidate's name may fail:

- different name length: middle name(s),
- accentuated letters,
- dash in the name,
- only initials of one or more names.

We experimented with different name matching methods, each addressing only some of the key issues. The candidate's name and the documents are represented as a sequence of terms, lowercased, accents on letters are replaced, and names with dashes are considered as two different terms (e.g., `Hazael-Massieux` ⇒ `hazael massieux`). The following matching methods were considered for our TREC 2005 experiments:

- M_0 `EXACT MATCH`: returns $true$ if the name appears in the document exactly as it is written.
- M_1 `NAME MATCH`: returns $true$ if the last name and at least the initial of the first name appears in the document.
- M_2 `LAST NAME MATCH`: returns $true$ if the last name appears in the document.

Note that each method M_i $(i = 1, 2)$ keeps, and improves upon, the results achieved by the preceding M_{i-1}.

Extract Candidates by Email Address. This method (`EMAIL MATCH`) simply extracts all email addresses appearing in a document. Email addresses were identified using a regular expression. According to our experiments this technique is less effective in terms of the number of identified candidates while the associations found by this method look like stronger relationships. See Table 3 for detailed results.

method	#candidates	#assoc	#docs
`EXACT MATCH`	696	324,258	136,627
`NAME MATCH`	757	354,315	139,801
`LAST NAME MATCH`	924	945,518	212,425
`EMAIL MATCH`	456	73,747	59,355

Table 3: Results of Name Extraction Methods.

4.3 Runs

We submitted the following 5 runs:

uams05run0 $m = 500$, `EXACTMATCH`

uams05run1 $m = 200$, `EXACTMATCH`

uams05run2 $m = 200$, `EMAILMATCH`

uams05run3 $m = 200,\ 0.5 \cdot$ `EXACTMATCH` $+ 0.5 \cdot$ `EMAILMATCH`

uams05run4 $m = 200,\ 0.375 \cdot$ `EXACTMATCH` $+ 0.208 \cdot$ `NAMEMATCH` $+ 0.416 \cdot$ `EMAILMATCH`,

where $m = |S|$ is the number of documents retrieved as most relevant given a topic. In the last two runs we experimented with a linear combination of different candidate-document association methods.

4.4 Results

`uams05`	#rel	map	R-prec	P@10	P@20	RR1
`...run0`	477	0.1225	0.1802	**0.240**	**0.202**	0.3942
`...run1`	472	**0.1277**	**0.1811**	0.222	0.200	0.4380
`...run2`	284	0.0918	0.1288	0.194	0.139	0.4975
`...run3`	**479**	0.1158	0.1489	0.194	0.138	0.4891
`...run4`	478	0.1177	0.1444	0.192	0.141	**0.5062**

Table 4: Results for the Expert Search task, where #rel is the number of relevant experts retrieved, and RR1 is the reciprocal rank of the first expert found.

Table 4 gives our overall results for the Expert Search task, using the various evaluation measures proposed by the task organizers; the best score per measure is indicated in bold face.

The results of `uams05run0` and `uams05run1` show no significant difference. We conclude that using different m values for cut-offs does not really affect overall performance.

At the same time, the association methods show interesting results; `EXACTMATCH` (`uams05run0, uams05run1`) retrieves more relevant hits while the reciprocal rank of the top relevant document (RR1) is much higher for `EMAILMATCH` (`uams05run2`). This underlines our expectations that the use of `EMAILMATCH` results in fewer but stronger associations. Combining different matching methods (`uamsrun3`, `uamsrun4`) shows promising results and suggests experimenting with more sophisticated estimation of candidate-document associations.

5 Conclusions

In this paper we described our participation in the TREC 2005 Enterprise track. We found that structured information was not particularly useful for either email search task. In the case of the discussion search, this was because too much bias was introduced by the thread size prior, whilst in the known-item task the ambiguity of the queries meant that the classification accuracy was mediocre, which influenced the quality of retrieval.

As to the expert search task, we found that the performance depends crucially on the ability to recognize names of experts. A textual representation of candidates' knowledge has been created according to the documents with which they are associated. In follow-up work we are exploring a second approach which does not create a candidate model directly, but assumes conditional independence between the query and the candidate and builds a so-called aspect model. Initial results from experiments aimed at comparing the two approaches seem to favor the this second, and they confirm, yet again, that the quality of document-candidate associations has great influence on the performance in case of both models.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.006, 640.-001.501, and 640.002.501

References

[1] M. A. Gonçalves, E. A. Fox, A. Krowne, P. Calado, A. H. F. Laender, A. S. da Silva, and B. Ribeiro-Neto. The effectiveness of automatically structured queries in digital libraries. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 98–107, New York, NY, USA, 2004. ACM Press.

[2] K. Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 1999.

[3] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 49–56, Tampere, Finland, 2001. ACM Press.