The University of Amsterdam at TREC 2005

David Ahn[1] **Leif Azzopardi**[1] **Krisztian Balog**[1] **Sisay Fissaha**[1] **Valentin Jijkoun**[1]
Jaap Kamps[1,2] **Karin Müller**[1] **Maarten de Rijke**[1] **Erik Tjong Kim Sang**[1]

[1] Informatics Institute, University of Amsterdam
[2] Archives and Information Studies, Faculty of Humanities, University of Amsterdam
http://ilps.science.uva.nl/

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

1 Introduction

At TREC 2005, we took part in the Enterprise, Terabyte, and Question Answering tracks. Our aim for the Enterprise track was to adapt the 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 organization's data. Our aim for the Terabyte track was to investigate the effects of larger collections. First, we wanted to test whether our retrieval system scales up to 25 million documents. Second, we hoped to find out whether results from earlier Web Tracks carry over to this task. Third, we wanted to investigate the role of smoothing for collections of this size. For the Question Answering track, we made two major adaptations to the system with which we participated in previous years: first by enabling the table stream to process additional question types and generate more candidate answers, and second by encoding the document collection and its linguistic annotation in XML thus enabling a QA-as-XML-retrieval strategy.

The rest of this paper is organized as follows. In three largely self-contained sections, we describe our work for the Enterprise (§2), Terabyte (§3), and Question Answering (§4) tracks. We summarize our findings in a concluding section.

2 Enterprise 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 (§2.1), whilst we considered the internal structure of an email for known item searching (§2.2). For the expert finding task our focus was on associating a document with a candidate expert to build candidate models (§2.3).

2.1 Email discussion search

The goal of this task was to retrieval 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. 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 and depth. Breadth, indicating the number of replies an email has directly received and depth, indicating the number of consecutive replies. We conducted an informal interviews with two email forum users, who regularly use technical forums. We 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,

[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.

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, that 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 retrieval strategy, the language modeling framework.

2.1.1 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 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) [24]. 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.

2.1.2 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 $p(d) = k$ 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).

2.1.3 Summary of Runs and Results

The following runs were submitted and the results are displayed in Table 2. 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 ($\alpha = 5$).

In Table 2, 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

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 2: Results for Discussion Search

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.

2.2 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 [5]. 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).

2.2.1 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 2.2.2). Each automatically structured query consisted of the set of fields, represented by q_x, which contained the assigned terms.

2.2.2 Fielded language model

The fielded language model is a simple extension of the standard language modeling approach described in §2.1.1. It treats each field of and 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.

2.2.3 Known-item runs

Our experiments examined the hypothesis that automatically inferred queries could be used to improve retrieval performance over unstructured queries (as in [5]). 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[19]. 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 give the best performance and subsequently used this form of smoothing for all runs and models.

2.2.4 Summary of runs and results

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 3: Results for Known Item Finding

The results for the known item finding subtask are shown in Table 3. The second column gives the mean reciprocal rank (MRR) score. The third and forth columns report the percentage of topics for which the known item was found in the top 10 and 100 documents, respectively. Whilst, the last column, reports the percentage of topics where no known item was found in top 100 documents (F@100). From our results, 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 ambigious nature of the queries seriously degraded the performance of our retrieval models, as the structure could not be reliably inferred.

2.3 Expert search

The Expert Search task presents the following scenario into consideration: 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.

2.3.1 Introduction

Our approach focuses on building candidate representations from the corpus, and then identifying the set of experts. To achieve this we apply the language modeling approach to the expert search problem. Under this approach, 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.

2.3.2 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 then can 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:

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

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

2.3.3 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 with 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 have experimented with different name matching methods of which we expect to handle the presented key difficulties. The candidate's name and the documents are represented as a sequence of terms. Moreover we assume that terms are lowercased, accents on letters are replaced and names with dash are considered as two different terms (`Hazael-Massieux` $\Rightarrow$ `hazael massieux`).

- 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 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 4 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 4: Results of different name extraction methods.

2.3.4 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.

2.3.5 Results

Table 5 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.

`uams05`	#rel_ret	map	R-prec	bpref	recip_rank
`...run0`	477	0.1225	0.1802	0.3963	0.3942
`...run1`	472	**0.1277**	**0.1811**	0.3925	0.4380
`...run2`	284	0.0918	0.1288	0.2531	0.4975
`...run3`	**479**	0.1158	0.1489	**0.4023**	0.4891
`...run4`	478	0.1177	0.1444	0.4004	**0.5062**

Table 5: Results for the Expert Search task.

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 name matching methods show interesting results; `NAMEMATCH` retrieves more relevant hits while the reciprocal rank of the top relevant document (recip_rank) is much higher for `EMAILMATCH`. This underlines our expectations that the use of `EMAIL MATCH` 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.

3 Terabyte Track

We participated in two of the Terabyte Track's tasks: adhoc and named page finding. Our aim for the Terabyte track was to investigate the effects of larger collections. First, we want to test whether our retrieval system scales up to 25 million documents. Second, we hope to find out whether results from earlier Web Tracks carry over to this task. Third, we

want to investigate the role of smoothing for collections of this size. Our retrieval system is based on the Lucene engine with a number of home-grown extensions [7, 17].

3.1 Experiments

3.1.1 Indexes

The Terabyte track uses the GOV2 test collection, containing 25,205,178 documents (426 Gb uncompressed). We created three separate indexes for (1) the full documents, (2) the text in the title tags, (3) the anchor-texts pointing toward the document. For the anchor-texts index, we ignored relative links and only extracted full links. We normalized URLs, and did not index repeated occurrences of the same anchor-text. This is similar to our earlier experiments in the TREC Web track [13, 14]. As to tokenization, we removed HTML-tags, punctuation marks, applied case-folding, and mapped marked characters into the unmarked tokens. We used the Snowball stemming algorithm [23].

We created a single, non-distributed index for the collection. The size of our full-text index is 61 Gb. Building the full-text index (including all further processing) took a massive 15 days, 6 hours, and 21 minutes.

3.1.2 Retrieval models

For our ranking, we use either a vector-space retrieval model or a language model. Our vector space model is the default similarity measure in Lucene [17], i.e., for a collection D, document d and query q:

$$sim(q,d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight_t,$$

where

$$\begin{aligned} tf_{t,X} &= \sqrt{\text{freq}(t,X)} \\ idf_t &= 1 + \log \frac{|D|}{\text{freq}(t,D)} \\ norm_q &= \sqrt{\sum_{t \in q} tf_{t,q} \cdot idf_t^2} \\ norm_d &= \sqrt{|d|} \\ coord_{q,d} &= \frac{|q \cap d|}{|q|} \end{aligned}$$

Our language model is an extension to Lucene [7], i.e., for a collection D, document d and query q:

$$P(d|q) = P(d) \cdot \prod_{t \in q} ((1-\lambda) \cdot P(t|D) + \lambda \cdot P(t|d)),$$

where

$$\begin{aligned} P(t|d) &= \frac{tf_{t,d}}{|d|} \\ P(t|D) &= \frac{\text{doc\_freq}(t,D)}{\sum_{t' \in D} \text{doc\_freq}(t',D)} \\ P(d) &= \frac{|d|}{\sum_{d' \in D} |d|} \end{aligned}$$

The standard value for the smoothing parameter λ is 0.15.

3.1.3 Official runs

We submitted six runs in total, using only the short topic statement in the title. For the adhoc task, we submitted the following two runs:

UAmsT05aTeVS Vector space model on the full-text index.

UAmsT05aTeLM Language model (λ = 0.15) on the full-text index.

For the named page finding task, we submitted four runs. We submitted a plain language model run:

UAmsT05nTeLM Language model (λ = 0.70) on the full-text index.

We also experimented with web-centric priors [12]. First, we assumed that pages with more inlinks are more likely to be relevant. Since our implementation of the language model calculates the logs of the probabilities, we took the exponent of the retrieval score, and multiplied it with the root of the indegree. We used the incomplete indegree scores we obtained from the anchor-text index. Second, we assumed that pages with shorter URLs are more likely to be relevant. We calculated the number of components in the domain and file path of the URL, e.g, trec.nist.gov/act_part/act_part.html has 3 (domain) plus 2 (file path) components. Again, we took the exponent of the retrieval score, and multiplied it with the reciprocal of the length of the URL.

UAmsT05nTind Language model (λ = 0.70) on the full-text index, with an indegree prior.

UAmsT05nTurl Language model (λ = 0.70) on the full-text index, with a URL prior.

Finally, we have not yet implemented a proper mixture language model incorporating different document representations. Instead, we combined separate runs made on the different full-text, anchor-text, and titles indexes.

UAmsT05n3SUM CombSUM of language model (λ = 0.70) runs on the full-text index (relative weight 0.8), anchor-text index (relative weight 0.8), and titles index (relative weight 0.8).

Task	#Topics	Model	Total		Avg.Q
Efficiency	50,000	VS	23,976	(6h39m36s)	0.480
Adhoc	50	VS	43	(43s)	0.862
Adhoc	50	LM	798	(13m18s)	15.962
Named Page	272	VS	594	(9m54s)	2.184
Named Page	272	LM	12,180	(3h23m)	44.781

Table 6: Performance measurements in seconds for max. 20 results per topic using the full-text index.

3.2 Results

3.2.1 Efficiency task

We created a run for the efficiency task as a post-submission experiment. Table 6 shows the total and average query processing times for the 50,000 efficiency task topics. We used a non-dedicated, dual processor machine running Linux with the retrieval system running as a single Java process with a heap size of 1 Gb. For the efficiency task we used the vector-space model. Total processing time for the 50,000 queries was 6 hours and 39 minutes. On average, it took 0.480 seconds to produce the top 20 results for a single query. For comparison, we also list the system performance for the other Terabyte tasks.

3.2.2 Adhoc task

There are in total 50 adhoc task topics. The number of relevant documents per topics varies from 4 to 559, with an average of 208 and a median 171. Table 7 shows the results for the adhoc task. We see an interesting comparison between

UAmsT05	#rel_ret	map	R-prec	bpref	recip_rank
...aTeVS	**5717**	**0.1996**	**0.2696**	**0.2290**	0.4437
...aTeLM	4180	0.1685	0.2097	0.1737	**0.6023**

Table 7: Results for the adhoc task.

the two retrieval models. First, we see that the vector-space model is superior on the overall measures (map, r-prec, and bpref). Second, we see that the language model is superior at early precision (recip_rank). The outcome deviates from results on the training data—the Terabyte track 2004 adhoc task topics—, where the language model outperformed the vector space model with a map score of 0.1562 versus 0.1413. Below, we will further zoom in on the language model and experiment with the amount of smoothing.

3.2.3 Named page finding task

In total there are 252 named page finding topics (20 topics have been deemed adhoc topics, and have been retracted from the qrels). The minimal number of relevant documents per topic is 1 and the maximum is 4525. For 187 topics there is a unique relevant page, the few topics with thousands of relevant pages are caused by page-duplicates in the collection. This leads to a skewed distribution with a mean of 47 and a median of 1 relevant page. Table 8 shows the results for the named page finding task (note that the 20 retracted topics are shown here as 'not found'). We make a number

UAmsT05	recip_rank	top 10		not found	
...nTeLM	0.3364	112	44.44%	78	30.95%
...nTind	0.2649	99	39.29%	78	30.95%
...nTurl	0.3251	115	45.63%	78	30.95%
...n3SUM	**0.3653**	**123**	48.81%	**77**	30.56%

Table 8: Results for the named page finding task.

of observations. First, the indegree prior results in a loss of performance. Second, the URL prior leads to mixed results: a loss of mean reciprocal rank, but a gain in the number of topics with the relevant page in the top 10. Third, the combination run leads to improved performance on all measures.

The success of the combination run shows the value of different document representations. On the Web Track data, mixture language models proved far more effective than straightforward run combination [13, 14]. This may also explain, in part, the mixed results for the link and URL priors. Other factors such as the incompleteness of the extracted links may also play an important role.

3.3 Smoothing experiments

In the language modeling framework, smoothing plays an important role: it helps to overcome data-sparseness, it introduces an inverted document frequency effect, and it expresses the relative importance of query terms [24]. In practice, smoothing is also a handle to tune a run toward recall (much smoothing) or precision (little smoothing). It is known that collection size is a factor influencing precision measures [6]. Hence, collection size may also be a factor influencing the amount of smoothing needed in the language modeling framework. Here, we focus on linear or Jelinek-Mercer smoothing, and investigate the effect of varying the smoothing parameter.

3.3.1 Named page finding task

First, we focus on the named page finding task. Since finding a 'unique' page requires precision rather than recall, we choose a relatively high value for the smoothing parameter(i.e., $\lambda = 0.7$). Table 9 shows the results while varying the smoothing parameter over the interval between 0 and 1. We make a few observations. As expected, we see that the named page finding topics do not require much smoothing. In fact, the less smoothing the better. This is in contrast with results on the Web Track data, where performance actually drops at the highest values of the smoothing parameter.

3.3.2 Adhoc task

Next, we focus on the adhoc task. Since adhoc topics require a delicate balance between precision and recall, we choose

λ	recip_rank	top 10		not found	
0.1	0.1684	57	22.62%	155	61.51%
0.2	0.2124	78	30.95%	127	50.40%
0.3	0.2417	83	32.94%	110	43.65%
0.4	0.2753	98	38.89%	99	39.29%
0.5	0.3046	103	40.87%	90	35.71%
0.6	0.3158	110	43.65%	84	33.33%
0.7	0.3364	112	44.44%	78	30.95%
0.8	0.3447	115	45.63%	77	30.56%
0.9	**0.3557**	**118**	46.83%	**74**	29.37%

Table 9: Smoothing for the named page finding task.

the standard relatively low value for the smoothing parameter (i.e., $\lambda = 0.15$). Table 10 shows the results while varying the smoothing parameter over the interval between 0 and 1. A few observations present themselves. We see that performance increases if we apply less smoothing. In fact, the gain is substantial; already the improvement for $\lambda = 0.2$ is statistically significant (99.9%, one tailed) over $\lambda = 0.15$. In sum, the adhoc task evaluated by mean average precision behaves like an early precision task.

λ	MAP	Prec@10
0.1	0.1467	0.3620
0.2	0.1856	0.4120
0.3	0.2138	0.4600
0.4	0.2355	0.4900
0.5	0.2540	0.5060
0.6	0.2707	0.5380
0.7	0.2863	0.5420
0.8	0.2998	0.5620
0.9	**0.3107**	**0.5680**

Table 10: Smoothing for the adhoc task using the full-text index.

3.4 Conclusions

Our participation in the Terabyte track was inspired by a number of aims related to the size of the Terabyte track collection, we now draw some initial conclusions.

Our retrieval system did scale up to the 25 million documents in the GOV2 collection. Performance at query time is impressive, especially for the optimized implementation of the vector space model. With respect to indexing time, with over two week to build a full-text index we seem to have reached the limits of building a non-distributed index.

For the adhoc task, we saw that standard IR techniques on a full-text index lead to good performance. We found that, on the 2005 topics, the vector space model outperformed the language model, although the language model can be substantially improved by using less smoothing.

For the named page finding task, we found that, on the one hand, indegree and URL priors did not promote retrieval effectiveness, but that, on the other hand, the combination of different document representations improved retrieval effectiveness.

Last, but not least, we zoomed in on the role of smoothing and found that less smoothing leads to the best performance. Whereas this is roughly according to expectation for the named page finding topics, it is unexpected for the adhoc topics. In addition to known relations between retrieval effectiveness and collection size [6], there also seems to be a relationship between collection size and the appropriate amount of smoothing in the language modeling framework.

4 Question Answering Track

This year, we took part in the main task of the question answering track as well as in the relationship finding task. We describe the results of our participation in the main task in Sections 4.1, and the results of the relationship finding task in Section 4.3.

4.1 QA: main task

We built on the multi-stream QA architecture that we have been developing over the past few years as part of our work for the TREC and CLEF QA tasks. The architecture has several streams run in parallel: each is based on a different approach to QA and is a self-contained QA system in itself. No new streams were added this year, leaving us with a total of seven streams: a table stream (detailed in §4.1.1 below), pattern matching and ngram mining (both against the collection and against the web), a Wikipedia stream (which gets answers out of Wikipedia), and Tequesta (which was replaced updated to Xquesta this year, see §4.1.2 below). For a more detailed description of our multi-stream approach we refer to [1, 2, 8, 10, 11].

Our QA efforts for the main task were concentrated in two areas. First, we enabled the table module to handle more question types and generate more candidate answers. Second, we upgraded the Tequesta stream by encoding the document collection as well as the different linguistic annotations in XML, thus enabling a "QA-as-XML-retrieval" strategy.

Question processing is the first stage in our system architecture which is common to all the streams. Each of the questions is tagged, and a question class is assigned based on our question classification module. The question types correspond to WordNet words and their senses. For instance, the question type of (9) is COACH%1. To determine the expected answer types, we also use the WordNet hierarchy. In our example, the question type COACH%1 is mapped to the expected answer type PERSON.

(9) *Q69.6: Who was the coach of the French team?*

Our question analysis was extended by exploiting the syntactic structure of the questions. Thus, we parse the questions using Charniak's parser [3]. The parses provide information

about the NP/PP-chunks which are used to determine the focus of the sentence, here *French team*.

4.1.1 Table stream modifications

An important part of our QA system is a table stream which relies on question-specific tables with answers which were extracted offline rather than during question processing [9]. The tables contain, among other, information on abbreviation expansions, birthdays, country leaders, and event dates. Evaluation of this stream on the 2004 factoid questions showed that while the accuracy of the returned answers was low (28% lenient evaluation), its coverage was even worse (7%). The work described in this section was intended to improve these scores.

The first step we took was to add extra tables. The topics of the tables were chosen based on TREC 2004 factoid questions which the system had been unable to answer: birthplaces of people, definitions, groups and their members, nicknames, and organizations, their founders and founding dates. The tables were filled by applying to the document collection hand-crafted extraction rules which utilized available syntactic and named entity annotation of the documents. Most of the tables were small (about 20k entries or less) with the group table (100k) and the definition tables (5M) as exceptions. The latter grew this large because the extraction rules included rules that derived definitions for arbitrary noun phrases. The new tables enabled the stream to handle previously unanswered questions like *What kind of animal is an agouti?* and their positive effects were larger than the side effect of pattern overgeneration.

Next, the question analysis and the table processing modules were updated. This work included defining new question topics, creating new question templates and making new links between question topics and tables. We also added a filter to the output of the table stream to make sure that when named entity answers from a certain category (person, location or organization) were requested by the question, ill-typed answers were removed. The filtering step could not be as precise as we would have liked it to be since the categories returned by our named entity analysis are more coarse-grained than the categories of the question analysis.

The adaptations of the table stream had a positive effect on its performance on the TREC 2004 factoid questions. The recall of the top answers went up from 7% (lenient evaluation) to 21% and even the precision of the answers went up (from 28% to 35%).

For factoid questions, we highly depend on the correct output of our named entity recognizer. We find some problems when assigning the correct named entities to a sentence. One improvement was to post-process the output of the recognizer by correcting obvious inconsistencies of named entity sequences. The following two examples exemplifies the sort of errors corrected in the output. The named entity recognizer often failed to assign the correct tag to names which are included in the name of an organization such as in (10).

(10) *the*/O *director*/O *of*/O *the*/O *Rose*/E-PER *Institute*/I-ORG *of*/I-ORG *State*/I-ORG *and*/I-ORG *Local*/I-ORG *Government*/I-ORG

In such cases, the named entity tags are changed to the most common one. Moreover, film titles and quotes in quotation marks are hard to detect for the named entity recognizer such as in (11). They are often misclassified as ORG or PER instead of MISC.

(11) *you*/O *'re*/O *in*/O *"*/O *The*/I-ORG *Sixth*/I-ORG *Sense*/E-ORG *./O "*/O

These errors have also been corrected with a postprocessing filter.

4.1.2 Semi-structured information retrieval

The system used in our previous participations in TREC-QA [1, 11] contained a text retrieval stream named Tequesta. This year we changed the task of the stream to XML element retrieval. The document collection was enriched automatically with token boundaries, syntactic and named entity annotation. Both the annotation and the original documents were stored in stand-off XML format. Our new stream XQuesta is able to query the collection with XPath and to retrieve elements that satisfy lexical, syntactic and named entity constraints. For this purpose the document collection was divided in non-overlapping sequences of paragraphs containing at least 400 characters. We found that having access to the corpus annotation improved the quality of the text snippets and allowed more elaborate answer filtering.

4.1.3 Handling "other" questions

The system changes described in the previous two section dealt with factoid questions. For list questions we only made a small modification: we return the same number of answers for each question (eight, when available) because with that number we obtained the best results for the TREC-2004 questions.

The basic strategy for answering the "other" questions has not changed significantly from last year. The method uses IR and NLP techniques to locate documents containing information about the topic, and extract nuggets from the retrieved documents. The nuggets are assigned an initial score, i.e., the retrieval score of the document from which the nugget was extracted. Furthermore, the method makes use of a reference corpus, an encyclopedia, in order to locate "good" facts for the given topic and use them to rerank the nuggets extracted from the target corpus in case the topic is found in the encyclopedia.

This year we introduced a centroid-based summarization technique for ranking nuggets extracted from the retrieved documents. It involves computing centroids, a set of statistically significant words which describe the list of nuggets extracted from the documents. The nuggets are then ranked based on their distance from the centroid [20].

4.1.4 Runs

We submitted three runs for TREC-QA 2005. We were interested in two research questions. First, would the system perform better with all six[3] streams or with a subset of these streams? This question was important since our work this year has focused on two streams (table and XQuesta) while other streams were not changed. In order to determine the best combination, we evaluated different stream combinations on the TREC 2004 questions. We found that the combination of table, XQuesta, Wikipedia and web ngrams was the best for factoid questions while XQuesta, ngrams from the web and ngrams from the collection performed best for list questions.

Using ngrams from the web for generating factoid answers has as a disadvantage that answers will be generated for almost all questions. This means that few NIL answers will be produced. We considered the presence of NIL answers as an interesting difference between the run with all streams (`uams05all`) and the run with a subset (`uams05be3`) and therefore we excluded the web ngrams stream from the factoid questions run. This means that the stream subset run used combinations of three streams: table, XQuesta and Wikipedia for factoid questions and XQuesta, ngrams from the web and ngrams from the collection for list questions. All runs contained the same answers for other questions.

The second question in which we were interested was: Will answer reranking based on web frequencies improve the quality of the top answers? In order to test this we created a run (`uams05rnk`) in which the answers of the question have been reranked based on their frequency. We replicated the *search engine corroboration* method of [4], in which answers are ranked according to their frequency of occurrence in the summaries of the top 1000 hits returned by a search engine for a query based on the question. We depart from the Bangor method in two respects: first, we use Yahoo rather than Google, because of the more convenient API, and second, we use a different method to construct queries on the basis of questions. Instead of extracting all NPs and VPs from a question to use as a single query, we submit two queries for each question and use the top 500 hits from each. One query is simply the question itself as a set of keywords, i.e., not constrained to be a phrase. The other query consists of *question selectors*, words from the question that are highly likely to occur in a correct answer snippet [21]. Question selectors are extracted from a question using a C4.5 decision tree trained on pairs of questions and correct answer snippets from previous editions of the TREC QA track; we followed [21] in our training procedure.

As the results in Table 11 show, however, this re-ranking method does not improve performance—in fact, re-ranking produces substantially fewer correct answers. The primary reason for this decline is that re-ranking tends to prefer answers that are shorter and more common on the web (irrespective of the question). An analysis of the 287 factoid questions for which the uams05rnk run yielded a different answer than the uams05all run reveals that in 241 cases, the answer chosen by re-ranking is more common than the answer chosen without re-ranking (according to Yahoo).[4] This analysis suggests that the first step to improving re-ranking is to use a more sophisticated scoring mechanism that normalizes with respect to the overall frequency of candidate answers; see [18, 22] for discussion of using web statistics in QA.

4.1.5 Results

Table 11 gives the combined results for the 3 QA tasks (accuracy for factoids, F score for list and other questions) and the overall scores of our three runs `uams05all`, `uams05be3` and `uams05rnk`. The column factoid accuracy contains three numbers exact answers, unsupported answers and inexact answers.

run	factoid accuracy (exact,unsup.,inex.)	list F	other F	overall
be3	0.119 , 0.052 , 0.050	0.064	0.201	0.127
all	0.105 , 0.058 , 0.086	0.050	0.200	0.113
rnk	0.066 , 0.025 , 0.039	0.029	0.201	0.090

Table 11: Results for the main QA task.

A potential cause for the low numbers could be a ranking problem: correct answers might not be ranked as number one. In order to check this, we estimated the accuracy of the system on factoid questions while looking at the top-*n* answers rather than only examining the top answer. This evaluation was performed automatically and therefore inexact and unsupported answers have also been counted as correct unlike in the official TREC-QA evaluation where only supported exact answers are correct. As Table 12 shows, our system potentially could have answered close to 60% of the factoid questions correctly (corresponding to an estimated 32% exact score) with a perfect ranking scheme.

A close look at the top 1 factoid answers generated for the first five targets by our best run (`uams05be3`) revealed that the errors made by the system had causes in different modules. Of the 27 factoid questions in this group, 22 were answered incorrectly. Incorrect handling of the target topic or

[3] The web pattern match stream which was employed in the last two editions was not included this year because of technical difficulties.

[4] Of the 46 times in which the answer chosen by re-ranking is less common, that answer is correct (or inexact) 9 times, while the answer chosen without re-ranking is correct (or inexact) 10 times.

n-answers	uams05all	uams05be3	uams05rnk
top 1	102 (28.2%)	85 (23.5%)	47 (18.0%)
top 2	126 (34.8%)	102 (28.2%)	73 (20.2%)
top 3	139 (38.4%)	109 (30.1%)	90 (24.9%)
top 5	160 (44.2%)	126 (34.8%)	109 (30.1%)
top 10	177 (48.9%)	149 (41.2%)	151 (41.7%)
top 20	190 (52.5%)	163 (45.0%)	188 (51.9%)
any rank	216 (59.7%)	188 (51.9%)	215 (59.4%)
MRR	35.1%	28.8%	21.9%

Table 12: Potential improvements of QA factoid scores.

the questions caused errors in twelve of the latter questions. We use Wikipedia for finding the most common version of names in the topic but unfortunately this process mapped the topic *France wins World Cup in soccer* to *FIFA Beach Soccer World Cup* which made finding correct answers for the related six questions hard. In other cases it was just difficult to determine the question topic or to find the right focus words. Question analysis is over-represented in the error cause list but to its defense it should be noted that where input processing failed, problems in other modules usually did not have a chance to surface.

The most important other problem was the justification of Wikipedia answers. In five cases the presented justification document was irrelevant for the question topic. Incorrect named entity labeling also caused problems for five questions although in two of these a solution would require annotation at a micro level which is beyond our current automatic annotation efforts (`stadium` and `placeInItaly` rather than `location`). Another system task which we need to review carefully is answer tiling (i.e., the combination of several partial answers to produce the final answer delivered as output). Four questions displayed tiling problems, often because correct answers were lost after they were combined with incorrect ones.

The three streams involved in this evaluation caused fewer errors than the previously mentioned parts. The Wikipedia and the XQuesta stream each produced three incorrect answers while the table stream generated one of these. The internal ranking of the Wikipedia stream should be improved as should answer filtering within XQuesta. An extra correct answer was missed because the word *competitor* was not linked to its synonym *contestant*. A more careful future use of WordNet could be helpful.

4.1.6 Conclusions

In our TREC-QA efforts for 2005, we built on our existing QA system and focused on on question analysis, named entity recognition, offline information extraction, encoding the document collection, and its annotation in XML and processing other questions. However, the performance of our system (our best run scores just over the median overall score and well under the median factoid score), leaves much room for improvement. The errors seem to be caused by different modules of the system. In our initial error analysis we have uncovered some of the problems to which we need to devote attention in the near future.

4.2 QA: Document ranking task

Our multi-stream QA architecture does not rely on an ordered set of documents returned from a preprocessing phase. In order to create the obligatory entry for the document ranking task, we returned the justification documents for each answer set in the same order as the final ranking of the answers. Note that our system associates exactly one justification document with each answer.

Table 13 lists the results obtained by the three runs submitted to the document ranking task with the scores for average precision, precision at 10 answers and precision at R answers, where R is the number of correct answers found by the human assessors. Since our main interest in the QA task lies with QA and not with IR, we have not taken any separate actions to optimize these scores. However, this change in combination with adaptations of the named entity annotation, question analysis and the table stream have not lead to an improvement of our 2004 score. We have identified a number of potential causes on which we will work in the coming year.

run	avg. prec.	prec. at 10	R-prec.
uams05all	0.108	0.170	0.129
uams05rnk	0.094	0.156	0.106
uams05be3	0.071	0.132	0.082

Table 13: Results for the document ranking task

4.3 QA: Relationship finding task

In the Relationship Finding task, systems were given topics, i.e., relatively verbose descriptions of user information needs, and had to return collection nuggets answering these needs. In most cases, a topic set a context and asked an explicit question about relationship between two or more entities. E.g.,

(12) *The analyst is interested in information regarding the Nobel Prize winners from previous years. Records indicate that David Trimble and John Hume shared the Nobel Peace Prize in 1998.*
Who are Trimble and Hume, and what was their relationship?

This year we took part in this task with a system based on passage retrieval, word similarity, and named entity matching. The system first retrieved passages relevant to a topic, thenextracted sentences from the retrieved passages, and reranked the sentences based on similarity to the topic. We describe the process in some detail.

Passage retrieval. The collection documents were split into passages of 400 characters (extended to the end of a paragraph). As in the main QA task, we used Lucene [17] with the standard Lnu.ltc model for passage retrieval. We used original full topics as retrieval queries, after (automatically) removing phrases and words likely to be irrelevant for user information needs, such as "*The analyst is interested in information regarding*" in example 12. The top 10 retrieved passages were split into sentences and processed further.

Topic processing. For a topic T, our system took its last sentence t (most often, the question expressing the user's information need) for subsequent processing. We extracted named entities from t using our NE tagger; in case t contained fewer than two named entities, we expanded t with preceding sentences, until it contained at least two NEs. For the example 12, t is "*Who are Trimble and Hume, and what was their relationship?*" and two NEs "*Trimble*" and "*Hume*" are extracted. The text t and the list of extracted named entities were used to rerank sentences obtained in the passage retrieval step.

Word-based sentence score. Each retrieved sentence s was assigned a score based on directed word similarity between s and t. In essence, we summed similarities between each word in t and its most similar word in s, according to a specific word similarity measure [15]. More details on the word-based calculation of similarity can be found in [16].

NE-based sentence score. We combined the word similarity-based score with the score based on the number of shared named entities between s and t. To detect whether two sentences contain common named entities (persons, organizations, locations, miscellaneous entities), we used a dictionary of NE variants created from lists of location-adjective correspondences (e.g., *Europe* and *European*) and redirecting links in Wikipedia (e.g., *William Jefferson Blythe IV* is also known as *Bill Clinton*, and *Burma* is an alternative name for *Myanmar*).

Collection sentences were ranked using the sum of word-based and NE-based scores, duplicates and near duplicates were removed using a simple string distance measure, and the best 5 sentences for each topic were returned as answer nuggets.

4.3.1 Runs and results

We submitted two fully automatic runs for the 25 official test topics. The run *uams05l* was created as described above and the run *uams05s* was identical, expect for the fact that the nuggets were shortened by removing all definite and indefinite articles, adjectives and adverbs (other than *first*, *last*, etc.). With the second run, we tried to create answers that were as short as possible (evaluation included a length penalty) without removing important material.

Both runs obtained the F-score of 0.12, with the median over all submitted fully automatic runs being 0.12, the best 0.228 and the worst 0.06.

5 Conclusions

In this paper we have described our participation in the TREC 2005 Enterprise, Terabyte, and Question Answering tracks.

For the 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.

For the Terabyte track, our main findings can be summarized as follows. First, our retrieval system scales up to 25 million documents, although in terms of indexing time we are approaching the limits of non-distributed retrieval system. Second, we found that larger collections require far less smoothing: especially for the adhoc task, using very little smoothing leads to substantial gains in retrieval effectiveness.

This year, our work for the Question Answering track was largely motivated by the wish to port one of the streams to a "pure" QA-as-XML-retrieval setting, where the target collection is automatically annotated with linguistic information at indexing time, incoming questions are converted to semistructured queries, and evaluation of these queries gives a ranked list of candidate answers.

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, and 640.001.501.

References

[1] D. Ahn, V. Jijkoun, G. Mishne, K. Müller, M. de Rijke, and S. Schlobach. Using Wikipedia in the TREC QA Track. In E. Voorhees and L. Buckland, editors, *The Thirteenth Text Retrieval Conference (TREC 2004)*, 2005.

[2] D. Ahn, V. Jijkoun, K. Müller, M. de Rijke, S. Schlobach, and G. Mishne. Making stone soup: Evaluating a recall-oriented multi-stream question answering stream for Dutch. In C. Peters, P. Clough,

G. Jones, J. Gonzalo, M. Kluck, and B. Magnini, editors, *Multilingual Information Access for Text, Speech and Images: Results of the Fifth CLEF Evaluation Campaign*, LNCS 3491, pages 423–434. Springer, 2005.

[3] E. Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL-00*, 2000.

[4] T. Clifton, A. Colquhoun, and W. Teahan. Bangor at TREC 2003: Q&a and genomics tracks. In *The Twelfth Text Retrieval Conference (TREC 2003)*. National Institute for Standards and Technology, 2003.

[5] 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.

[6] D. Hawking and S. Robertson. On collection size and retrieval effectiveness. *Information Retrieval*, 6:99–150, 2003.

[7] ILPS. The ILPS extension of the Lucene search engine, 2005. http://ilps.science.uva.nl/Resources/.

[8] V. Jijkoun and M. de Rijke. Answer selection in a multi-stream open domain question answering system. In S. McDonald and J. Tait, editors, *Proceedings 26th European Conference on Information Retrieval (ECIR'04)*, volume 2997 of *LNCS*, pages 99–111. Springer, 2004.

[9] V. Jijkoun, G. Mishne, and M. de Rijke. Preprocessing documents to answer dutch questions. In *Proceedings of BNAIC'03*. Nijmegen, The Netherlands, 2003.

[10] V. Jijkoun, G. Mishne, and M. de Rijke. How frogs built the Berlin Wall. In *Proceedings CLEF2003*, volume LNCS. Springer, 2004.

[11] V. Jijkoun, G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. The University of Amsterdam at the TREC 2003 Question Answering Track. In *Proceedings TREC 2003*, pages 586–593, 2004.

[12] J. Kamps. Web-centric language models. In *Proceedings of the Fourteenth ACM Conference on Information and Knowledge Management (CIKM 2005)*. ACM Press, New York NY, USA, 2005.

[13] J. Kamps, G. Mishne, and M. de Rijke. Language models for searching in Web corpora. In E. M. Voorhees and L. P. Buckland, editors, *The Thirteenth Text REtrieval Conference (TREC 2004)*. National Institute of Standards and Technology. NIST Special Publication 500-261, 2005.

[14] J. Kamps, C. Monz, M. de Rijke, and B. Sigurbjörnsson. Approaches to robust and web retrieval. In E. M. Voorhees and L. P. Buckland, editors, *The Twelfth Text REtrieval Conference (TREC 2003)*, pages 594–599. National Institute of Standards and Technology. NIST Special Publication 500-255, 2004.

[15] D. Lin. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning*, 1998.

[16] D. Lin. Recognizing textual entailment: Is word similarity enough? In *Lecture Notes in Artificial Intelligence*, 2005. to appear.

[17] Lucene. The Lucene search engine, 2005. http://jakarta.apache.org/lucene/.

[18] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of ACL 40th Anniversary Meeting (ACL-02)*, pages 425–432, University of Pennsylvania, Philadelphia, 2002.

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

[20] D. R. Radev, H. Jing, M. Stys, and D. Tam. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40:919–938, 2004.

[21] G. Ramakrishnan, S. Chakrabarti, D. Paranjpe, and P. Bhattacharya. Is question answering an acquired skill? In *Proceedings of the 13th international conference on World Wide Web*, 2004.

[22] S. Schlobach, D. Ahn, M. de Rijke, and V. Jijkoun. Data-driven type checking in open domain question answering. *Journal of Applied Logic*, 2006. To appear.

[23] Snowball. Stemming algorithms for use in information retrieval, 2005. http://www.snowball.tartarus.org/.

[24] 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.