The Lowlands' TREC Experiments 2005

Henning Rode[1] Georgina Ramírez[2] Thijs Westerveld[2]
Djoerd Hiemstra[1] Arjen P. de Vries[2]

[1]University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
{h.rode,d.hiemstra}@utwente.nl

[2]CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
{georgina,thijs,arjen}@cwi.nl

1 Introduction

This paper describes our participation to the TREC HARD track (High Accuracy Retrieval of Documents) and the TREC Enterprise track. The main goal of our HARD participation is the development and evaluation of so-called query profiles: Short summaries of the retrieved results that enable the user to perform more focused search, for instance by zooming in on a particular time period. The main goal of our Enterprise track participation is to investigate the potential of the structural information for this type of retrieval task. In particular, we study the use of the *thread* information and the *subject* and *header* fields of the email documents. As a secondary and long standing research goal, we aim at developing an information retrieval framework that supports many diverse retrieval applications by means of one simple yet powerful query language (similar to SQL or relational algebra) that hides the implementation details of retrieval approaches from the application developer, while still giving the application developer control over the ranking process. Both the HARD system and the Enterprise system (as well as our TRECVID video retrieval system [14]) are based on MonetDB, an open source database system developed at CWI [1].

The paper is organised as follows. First, we discusses our participation in the HARD track. We define query profiles and discuss the way we generate them in Section 2. Section 3 describes the clarification forms used and Section 4 explains how we refine the queries and rank the results. We end this part by analysing our experimental results in Section 5 and giving some conclusions for this track in Section 6. The second part of the paper discusses our participation in the enterprise track. We start by describing the system and experimental setup in Section 7. Section 8 discusses the approaches taken for each of the subtasks and Section 9 analyses the results. We end by giving some conclusions and future work for this track in Section 10. The final part of the paper describes our future plans for building a so-called *parameterised search engine* within the Dutch National project MultimediaN.

HARD TRACK EXPERIMENTS

The following part reports about our HARD track experiments. It shows how query profiles can be employed for clarification and describes the evaluation of the system within the HARD track setting.

In an interactive information retrieval session, which is the underlying concept of the HARD track, the role of the system is to detect query ambiguity and to ask the user for clarification in a way, she/he feels able to answer. Following an idea of Diaz and Jones [5] to predict the precision of queries by using their temporal profiles, we analyzed temporal as well as topical profiles with respect to their application in clarification forms. We hope to enable the user to give better feedback to the system by showing him/her this summarized information about the expected query outcome.

2 Query Profiles

Query profiles play a central role within our HARD track experiments. Therefore, we start this paper by a short general definition of the term and show later in more detail how special types of query profiles can be computed and analyzed.

Definition A query profile shows the distribution of documents, the system regards as relevant to the given query, with respect to a certain document property. E.g., a temporal query profile shows the distribution of relevant documents along the time dimension, a topical profile along the dimension of predefined topics the answers belong to.

The set of documents, the system regards as relevant to a given query, is determined by the top X documents from the ranked answer list. Notice, that this set can differ by far from the set of documents relevant to the user. By summarizing and visualizing information about the expected query results in profiles the user should be able to realize this difference him-/herself and to refine the query.

Expectations on the work with profiles:

- Query profiles give an helpful and easy to understand preview over the expected results.
- The user can detect ambiguity in the query and resolve it.
- Combining profiles with interactive means to express preferences or dislikes can be a powerful tool for query refinement.

Whereas the general ideas stay the same for all kinds of query profiles, there are several domain specific issues to consider. We will thus take a closer look on generating temporal and topical profiles, the two types used in our HARD track experiments.

2.1 Generating Temporal Profiles

Having a date-tagged corpus, a basic temporal profile for a given query is simple to compute. We treat the 100 top ranked documents from the baseline run as the set of relevant answers R and aggregate a histogram with monthly time steps:

$$frq(R_i) = \{|D_j| \; |month(D_j) = i\}.$$

The decision for the granularity of one month is based on the overall time span of the corpus and the timeliness of news events. Other granularities, however, would be considerable as well.

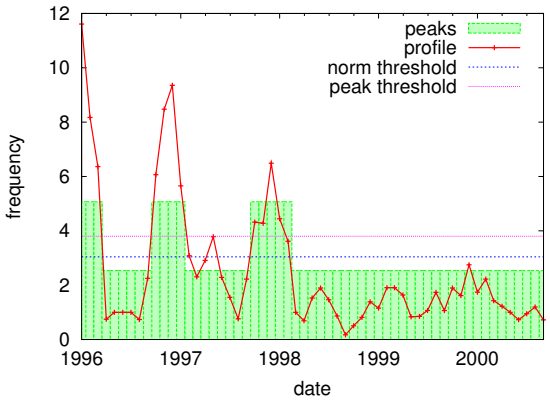

Figure 1: Temporal Profile of Topic 363: *Transportation Tunnel Disasters*

As a next step, we performed a *time normalization* on the profile. Knowing that the corpus articles are not evenly distributed over the total time span, the time profile should display the relative monthly frequency of articles relevant to the given topic rather than absolute numbers. Therefore, the frequency of each monthly partition R_i is divided by the total number of corpus articles C_i originating from month i. The average frequency $\overline{frq(R)}$ is used here as a constant factor to avoid exceptional small numbers:

$$frq^*(R_i) = frq(R_i)/frq(C_i) * \overline{frq(R)}.$$

Furthermore, we performed moving average smoothing on the histogram, a technique used for trend analysis on time series data. It replaces the monthly frequencies of the profile by the average frequencies of a small time window around the particular month. We used here a window size of 3 months:

$$frq^{**}(R_i) = \frac{frq^*(R_{i-1}) + frq^*(R_i) + frq^*(R_{i+1})}{3}.$$

There are two reasons for using such a smoothing technique. First, the time-line the search topic is discussed in the news will often overlap with our casual monthly partitioning. Second, although we want to spot peaks in the profile, we are not interested in identifying a high number of splintered bursts. If two smaller peaks are lying in a near timely neighborhood they should be recognized as one. The graph in Fig. 1 shows an example of a resulting temporal profile.

Finally, we want to determine the number, bounds, and the importance of peaks in the temporal profile. Diaz and Jones [5] tried several techniques for this purpose and decided to employ the

so called burst model [7]. However, starting with the described preprocessing of the temporal profile a simple threshold driven approach seemed sufficient in our case. In order to explain the method one should imagine two horizontal lines dividing the temporal profiles into 3 areas (see Fig. 1). We will call the upper line peak threshold and the lower one norm threshold. It remains a matter of parameterization where to set those lines exactly. For our experiments, we set the thresholds depending on the average frequency of the profile $\overline{frq^{**}(R)}$:

$$\begin{aligned} norm\ threshold &= 1.2 * \overline{frq^{**}(R)}, \\ peak\ threshold &= 1.5 * \overline{frq^{**}(R)}. \end{aligned}$$

We define then a peak as any continuous time span with frequencies above the norm threshold, if there is at least one frequency value inside above the peak threshold. On the other hand, between two different peaks lies always at least one frequency value below the norm threshold, otherwise the two or more bursts are counted as one. Following this definition, the bounds of existing peaks in the profile can easily be determined.

When we also want to compute a measure for the importance of the found peaks P, the corresponding frequency values of the temporal profile can simply be summed up. A further division by the average of such frequency sums $\overline{frq(P)}$ leads to a value for peak intensity better comparable among different temporal profiles:

$$\begin{aligned} frq(P_j) &= \sum_{i \in P_j} frq^{**}(R_i), \\ intensity(P_j) &= frq(P_j)/\overline{frq(P)}. \end{aligned}$$

2.2 Generating Topical Profiles

For topic classification we need to build abstract (language-) models for all different concepts, the classification should take into account. In order to save work and to keep the experiments comparable, we used the models built for last year's HARD experiments. The queries then distinguished 12 different concepts similar to the main sections of common newspapers (like politics, business, sports...). A further advantage using the old models is that they are based on a different corpus and thus can't be illegally over-fitted. A more detailed description about the construction of these language models can be found in our last year's TREC paper [10].

The required text classification for computing a topical profile differs slightly from the typical categorization task (described in [11]). We do not need to assign binary labels whether a document belongs to a certain category or not. A similarity measure showing to which extend an article belongs to a given category is already sufficient. Hence, the task falls back to the known domain of ranking a set of documents given a query. In fact, an abstract language model describing a topical concept is nothing but an exceptional long query. We used in the experiments the NLLR measure (described in a later section) which is also applied to compute a score for the initial query. Only the smoothing factor λ is set smaller in this case. Firstly, because the exceptional query length makes smoothing less important, and secondly, to increase differences between the models.

In order to speed up the computation of topical profiles as well as the later ranking procedure the score computation is performed offline. For each classifier in the set of topical concepts a score vector is maintained, holding the individual scores for all documents within the collection.

After the classification task is done, topical profiles can be computed in the following way. Similar to temporal profiles explained previously, the set R of the 100 top ranked documents given the query is determined. The score for a specific topic category T_i is then defined by the sum of all document scores in R for this category. The intensity value, as introduced in the last section, is computed accordingly:

$$\begin{aligned} score(T_i) &= \sum_{D \in R} NLLR(T_i|D), \\ intensity(T_i) &= score(T_i)/\overline{score(T)} \end{aligned}$$

An example topical profile is displayed in Fig. 2.

3 Clarification Forms

Compared to the profiles shown in the last section (Fig. 1 and Fig. 2) a user does not need to see the whole spectrum of the profile. Instead it seems sufficient to cut out the most relevant part of it, which means the highest temporal or topical peaks. For the experiments, we just displayed the 5 top ranked topics, but all identified temporal peaks. In practice their number never exceeds 4.

Besides providing a preview on the query results, our clarification forms allow to refine the query. In

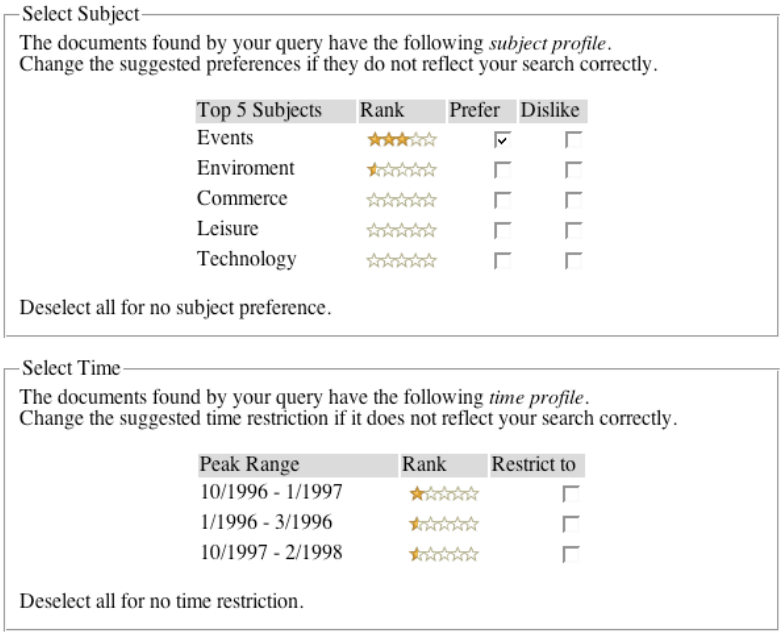

Figure 3: Clarification Form of Topic 363: *Transportation Tunnel Disasters*

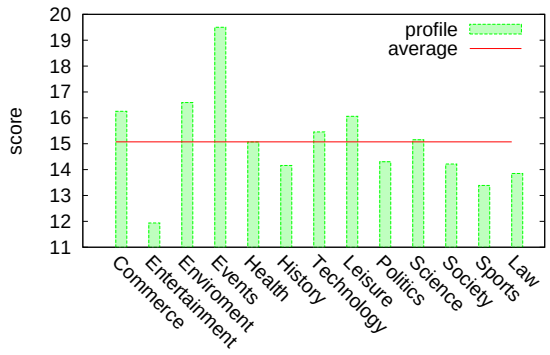

Figure 2: Subject Profile of Topic 363: *Transportation Tunnel Disasters*

the topical profile the user can state preferences and dislikes, which will influence the final result list accordingly. Similarly, the time profiles allow to express temporal restrictions for the search to any combination of peak ranges.

3.1 Automatic Preselection

We also looked, whether it is possible to make an automatic suggestion of an appropriate selection in the profiles. Obviously, the most high ranked topics or temporal peaks are good candidates, especially if they distinctively stand off from the lower ranked ones. The intensity measure defined in the last section explicitly addresses these characteristics. Using an intensity threshold, we can preselect all topics and temporal peaks above. In the experiments an intensity threshold of 1.2 was used for the topical profiles, respectively 1.5 for the temporal profiles. These values have been shown high enough to assure the selection of only distinctive peaks of the profile. An example clarification form with preselected items is shown in Fig. 3.

4 Query Refinement and Ranking

4.1 Translation to NEXI Queries

Having a parameterized search system as a high level objective in the Lowlands team, that allows to be adapted to all kind of search scenarios, we need to express our queries in a common language. We found the NEXI query language used in the field of structured retrieval a suitable and already well known candidate. Hence, the first task here will be to express the complex user refined HARD track queries in the NEXI syntax (A definition of NEXI can be found in [12]). It provides along the way a comprehensible overview of the ranking procedure.

Baseline queries searching for documents (with

tag-name `DOC`) that are relevant to the query terms `t1...tn` are expressed in NEXI in the following way: `//DOC[about(., t1...tn)]`.

The `about` usually triggers a scoring process according to a set of query terms, but the basic expression can be extended in a syntax-conform way to initiate a scoring with respect to a predefined language model: `about(., model(sports))`. Similarly, the preceding minus sign for terms can be translated accordingly to language models in order to express the dislike of such a model: `about(., -model(business))`. Furthermore, we can combine scores with the `and` operator. A user refined query for topic 363 might look now in the following way:

```
//DOC[about(., Transport Tunnel Disasters)
and about(.,model(events) -model(sports))]
```

The last needed query language construct is a time restriction. In contrast to the soft preference expressed by score combination above, the time filter should be restrictive, either letting a document pass or not. Such a filter is simply set by further predicates on the document nodes exploiting the XML-structure of the collection. Here written in a simplified syntax:

```
//DOC[about(., ...)][//date > min
and //date < max]
```

4.2 Retrieval Model and Score Combination

Now all HARD track queries can be expressed in the NEXI syntax, however, we have neither specified the concrete employed retrieval model of the `about` operator nor the used score combination technique. The abstraction of the NEXI syntax still allows to change those parts.

For the experiments, we continued following the language modeling approach from last year. In particular, we choose again the NLLR, the normalized logarithmic likelihood ratio [8], as a score function:

$$\sum_{t \in Q} P(t|Q) * \log \left(\frac{(1-\lambda)P(t|D) + \lambda P(t|C)}{\lambda P(t|C)} \right) = NLLR(Q|D).$$

The NLLR is able to compare query terms and documents as well as entire language models. Due to the normalization it produces comparable scores independent of the size of the query. The factor λ determines the degree of smoothing with the background collection model. Since smoothing plays an important role for short queries, whereas it dilutes the score differences for large-scale query language models, we decreased λ from 0.85 for queries to 0.5 for topic models.

If an `about` function includes several language models M_i, their logarithmic scores are simply added, respectively subtracted for disliked models with preceding minus. This allows to make efficiently use of the precomputed document scores for topic language models:

$$score(D) = \sum_{M_i} \pm NLLR(M_i|D).$$

Finally, the `and` operator takes on a special position. It is only used here for combining the initial baseline query with the scores from the topic models. In this special case we have to ensure that the scores on both sides deliver "compatible" values or even more to guarantee the superiority of the initial query in the final result. A minimum-maximum normalization solves such a task (among others described in [3]). It shifts in a first step the scores on both sides to set the minimum to zero and stretches the ranges of scores to an equal level by comparing the maxima on both sides. In our case, we even stretch the score range of the baseline query to double the size of the other part to stress its dominance in the final ranking.

5 Experimental Results

We submitted this year two baseline runs, `TWENbase1` and `TWENbase2`. Whereas the first uses the title keywords only, the second makes use of title and description of the topic. Both runs, thus, only differ with respect to query length. Furthermore, we submitted in total six final runs, three belonging to each of the two baseline runs. The `TWENuser` runs evaluate all information given by the user in the clarification form. In contrast, the `TWENblind` runs take in a blind-feedback manner only account of the system's preselections explained above. Finally, the `TWENdiff` runs analyze the difference between the two and only utilize the changes the user made over the preselections.

All original HARD submissions this year suffer unfortunately from a hard-to-spot system bug. Our system produced still useful output, however less accurately than possible. For this reason we rerun all experiments with the inevitable inconsistency between the new rankings and the user feedback based on the old clarification forms. In order to keep the results comparable, we also used the old system preferences for the `TWENblind` runs.

A closer look at the set of the 50 search topics revealed, that they haven't been distinctive with respect to their temporal profile. In fact, there was almost no case where the user wanted to restrict the query to a certain time span. We restricted our analysis, therefore, to the improvements by topical query refinement and ignored all temporal restrictions.

Table 1 presents an overview on the main evaluation measures computed for all 8 runs. At a first glance it is visible, that the topic refined queries show a considerable improvement over the baseline runs. The precision gain is most visible at the $P10$ measures. If we compare the results with respect to the initial query length, we see the title-and-description runs always ahead of the title-only, but the difference becomes smaller for the refined queries. Hence, the refinement strategy helps the most in the case of short queries, which are typical for common search engine users.

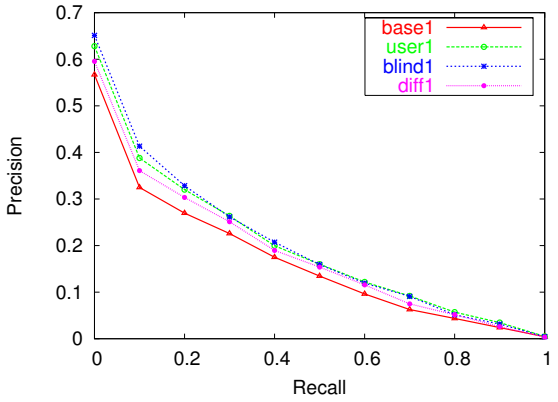

Figure 4: Precision-recall graph for all runs based on title-only queries

The precision recall graph in Fig. 4 confirms the observation made with the $P10$ values. The precision gain stays the highest at the top of the ranked list. The user refined query results soon converge to the baseline. Surprisingly the blind feedback variant stays even on top of all other runs.

A more detailed view on the results of all single search topics is presented in Fig. 5. The queries reaching high MAP values already in the baseline run, apparently gain the most from query refinement. The relation between the improvement and the quality of the initial query result seems almost proportional. This might give an explanation for the good results of the blind feedback run, which depends purely on the quality of the initial query.

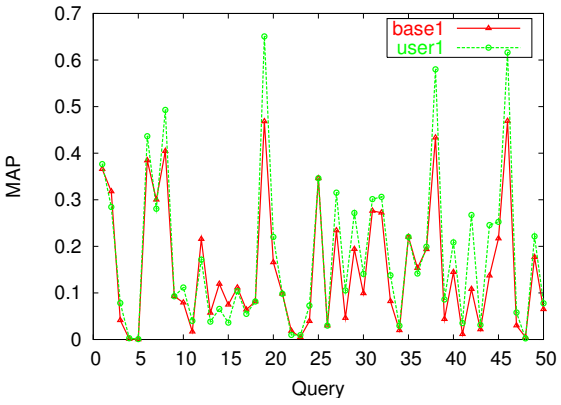

Figure 5: MAP improvements on single topics

6 Conclusions and Outlook

The results show promising improvements for all runs that make use of query profiles. However, our so called blind feedback run performed at least as good as the one with user interaction. Therefore, we must conclude that the main information the query profile shows currently can already be extracted and used by the machine itself. The clarification forms have been less useful so far to dissolve query ambiguity by user interaction. However, we think it is too early for a general judgment on the usability of query profiles for clarification. Firstly, our clarification forms suffered to an unknown extend from the reported system bug. Secondly, we would expect better results by a finer grained "topical resolution" combined maybe with a technically more elaborated classification process.

Furthermore, we need to examine search improvement by query profiles on other dimensions. The temporal profiles remained untested by the current HARD track query set, but also geographical or genre profiles - in order to name just two possible other parameters - might enable similar improvements as the topical query refinement.

ENTERPRISE TRACK EXPERIMENTS

In this part, we report about the experiments carried out for the Enterprise track. We explain the approaches taken and the results obtained for each of the subtasks of the track: known-item search, discussion search, and expert search. The main goal of our experiments is to investigate the po-

	base1	user1	blind1	diff1	base2	user2	blind2	diff2
MAP	0.1511	0.1808	0.1844	0.1686	0.1809	0.1985	0.2049	0.1927
R-prec	0.2142	0.2387	0.2479	0.2295	0.2447	0.2557	0.2638	0.2512
P10	0.2860	0.3560	0.3600	0.3100	0.3760	0.4260	0.4220	0.3880

Table 1: Result overview: title-only queries (base1) vs. title-and-description (base2)

tential of the structural information for these type of retrieval tasks. In particular, we study the use of the *thread* information and the *subject* and *header* fields of the email documents.

7 System and Experimental Setup

As mentioned in the introduction, our long standing research goal is to develop an information retrieval framework that supports diverse retrieval scenarios. An example of the need of such a system can be seen already within the enterprise track. In our case, the approaches taken for each of the sub-tasks required a different document manipulation and retrieval approach. Instead of implementing the specific approach for each of the sub-tasks, a *parameterised search engine* would provide an automatic implementation and processing of these approaches. Unfortunately, for this year tasks, we still had to make use of two independent systems. *MonetDB/XQuery* [2], an XQuery implementation build on top of the MonetDB database system [1], was used for structure manipulation when preparing the collection. *Tijah XML-IR* [9], a content oriented XML retrieval system build on top of the same relational database system, was used for ranked retrieval. Tijah provides implementations of different retrieval models and supports structural queries expressed in the NEXI query language [13]. A tighter integration of the two XML processing systems is planned for future work.

7.1 Collection Pre-processing

For all our experiments we used only the *lists* part of the W3C corpus. However, to be able to study the structural aspects of the collection with the Tijah system, we manipulated the collection in the following way:

- Explicitly include the thread information by creating a *thread* field and clustering the mails by thread.
- Convert the attribute fields into text fields to be able to use the ranking facilities provided by the Tijah system.

7.1.1 Thread Detection

A *thread* is the sequence of messages posted to a mailing list or newsgroup dealing with the same topic. Messages in such a thread typically refer to a previous email and sometimes this information is recorded in some of the email fields. However, this is not always the case and that is when thread detection becomes a difficult problem.

To cluster the emails from the collection by thread, we used the thread information contained in the *all-in-reply-to* list provided by William Webber[1]. He uses different fields from the marked-up HTML and from the mail headers to identify child-parent relationships. In this case, child is a mail that refers to another mail (parent).

We created an algorithm to cluster all the mails that belong to the same thread. The algorithm starts by creating a thread for each mail that does not reply to any other (named root) and recursively, adds to that thread the mails that refer to this one or to any of the mails in that thread, a so-called top-down processing approach.

However, the child-parent information is not exact nor complete. Due to the existence of *cycles* in the threads (such as mail A refers to B, mail B refers to C and mail C refers to A), not all the mails could be classified using this algorithm. The remaining 63 mails, contained in 7 different threads, were classified manually.

Once all the mails were clustered, we created an XML document with the thread index using the structure shown in Figure 6. The mails from the collection not belonging to any of the identified threads were added to the index as single-mail threads.

7.1.2 Structure Manipulation

Using the information from the created thread index and from the cleaned collection made available

[1]Available from `http://www.cs.mu.oz.au/~wew/w3c-lists-threads-wew.gz`

```
<THREADS>
  <THREAD THREADID="1">
      <DOCID>lists-000-0065591</DOCID>
      <DOCID>lists-000-0329499</DOCID>
  </THREAD>
  <THREAD THREADID="2">
    ....
</THREADS>
```

Figure 6: Structure of the threads index document: threads.xml

by Daqing He[2], we created a new XML file with the following modifications:

- Emails are clustered by thread and the threads are identified.
- Emails not belonging to any thread are considered as a thread themselves.
- Attribute values from the header field considered important for retrieval are converted into text fields. The reason for that is that, so far, the Tijah system can only use attribute values as strict predicate constraints. To make use of them for similarity ranking, they have to be contained in a text field.
- Fields considered not important for retrieval are removed from the collection. We decided to remove the *isoreceived* and *isosent* information.

Manipulation of XML structure is simplified by using XQuery. For example, the creation of our purpose-specific collection with the above four requirements, can be done using the single XQuery query shown in Figure 7.

The structure of the resulting modified collection is shown in Figure 8.

8 Approaches and Retrieval Model

Tijah is based on a scored region algebra approach. Each element in an XML tree naturally represents a region in the document. The scored region algebra provides functionality for scoring these regions (based on a specified retrieval model), and for combining scored regions in a principled manner. Tijah takes NEXI queries and produces a ranked list

[2]Available from `http://www.sis.pitt.edu/~daqing/w3c-cleaned.html`; this cleaned corpus is in turn based on Yejun Wu's parsed collection, which is available at `http://tides.umiacs.umd.edu/webtrec/trecent/parsed_w3c_corpus.html`.

```
for $t in doc("threads.xml")//THREAD
return (
 <THREAD><ID>{$t/@THREADID}</ID>
 {for $m in $t/DOCID
  for $d in doc("trecent.xml")/DOCS/DOC
  where $d/DOCNO/@VALUE=$m
  return(
    <DOC>
    {$d/DOCNO}
    <HEADER>
      <RECEIVED>{data($d/RECEIVED/@VALUE)}</RECEIVED>
      <SENT>{data($d/SENT/@VALUE)}</SENT>
      <NAME>{data($d/NAME/@VALUE)}</NAME>
      <EMAIL>{data($d/EMAIL/@VALUE)}</EMAIL>
      <SUBJECT>{data($d/SUBJECT/@VALUE)}</SUBJECT>
      <TO>{data($d/TO/@VALUE)}</TO>
      <CC>{data($d/CC/@VALUE)}</CC>
    </HEADER>
    {$d/TEXT}
    </DOC>)}
 </THREAD>)
```

Figure 7: XQuery query for processing the header and thread information

of XML elements. Consider the following example query:

```
//THREAD[about(.,X)]//DOC[about(.//SUBJECT,Y)
or about(.//TEXT,Y)]
```

This is expected to return `<DOC>` elements enclosed in `<THREAD>` elements about X, where the `<DOC>` element should either have a `<SUBJECT>` or a `<TEXT>` about Y. The actual implementation of the assigning and combining scores is flexible in Tijah and can be defined at a relatively high level. Scores are assigned to regions for each `about()` clause encountered in a query. Score combination occurs when results from multiple `about()` clauses have to be merged. For example, when an `AND` or `OR` operator is used, or when constraints are specified at different levels of the document hierarchy (like the combination of `<THREAD>` and `<DOC>` elements in the given example). For assigning scores an application developer can choose from a variety of information retrieval models (e.g., Vector space model, language models), for combining scores there is a choice of different aggregation functions.

For the Enterprise track we study the effect of structural information and fix our choices for retrieval model and score combination functions. We use a standard language modelling approach. This means the `about()` clause is implemented as a simple interpolation of foreground (element) and background (collection) probabilities:

$$\sum_{t \in Q} P(t|Q) * \log\left((1-\lambda)P(t|D) + \lambda P(t|C)\right), \quad (1)$$

```
<COLLECTION>
  <THREAD>
    <ID THREADID="1"/>
    <DOC>
       <DOCNO VALUE="lists-000-12345"/>
       <HEADER>
            <RECEIVED> </RECEIVED>
            <SENT> </SENT>
            <NAME> </NAME>
            <EMAIL> </EMAIL>
            <SUBJECT> </SUBJECT>
            <TO> </TO>
            <CC> </CC>
       </HEADER>
       <TEXT></TEXT>
    </DOC>
    <DOC>
    ...
  </THREAD>
  <THREAD>
  ...
</COLLECTION>
```

Figure 8: Structure of the email collection after preprocessing.

where D, the *document*, stands for the element we want to score. For `AND` and `OR` operators we used product and sum functions respectively and for combining scores from multiple levels in the XML tree, we used the product of the scores at the different levels. With all these parameters fixed, we can study the effects of using structural information. Structural constraints can easily be expressed in NEXI queries.

8.1 Known-item search

The known-item search tasks simulate a user trying to find an already seen email. Systems have to use the terms provided by the user to locate the specific mail within the mail archive. The kind of information a user remembers about the mail might vary considerably. Typically, a user remembers some terms contained in the mail or the main topic of it, but sometimes he or she might also recall the person that sent the mail or the date it was sent. For this type of search task, we believe that the structural information of the mails, where information such as date, subject or sender is recorded, is an important factor to take into consideration when searching. This information, contained typically in the header field, can be used to, e.g., redefine the content based ranked list. This year at TREC, we study the effects of using the *subject* field and the complete *header* field information in combination with the text based scores.

The different runs we submitted for this task are shown in Table 2.

8.2 Discussion search

The discussion search task simulates a user trying to find different arguments about a topic. He or she wants to find pros and cons about some topic discussed in the mailing list. This type of information is very related to the thread information because a discussion about a topic will typically consist of mails referring to each other and just adding some new opinion on the matter. Therefore, for this task, we investigate what is the effect of using the information that a mail belongs to a certain thread. In particular, for a given query, we multiply the mail scores with the score of the thread they belong to.

We submitted for this task the two types of runs shown in Table 3.

8.3 Expert search

The expert search task simulates a user trying to find who to contact with questions about a given topic. The task is to return a ranked list of persons rather than a ranked list of documents or emails.

We only experimented with a very basic approach to expert finding, where we concentrated on the mailing lists part of the collection. For each of the persons in the list of candidates, we used a simple XQuery query to produce a person specific document consisting of all the emails for which the candidates email address was mentioned either in the *email* field of the pre-processed corpus, or in the *name* field. This roughly corresponds to constructing a corpus of all the mails sent by a given candidate.

From each of the person-documents, we estimated a language model and used this model as the person profile[3]. The personal profiles are easily extensible with information from other parts of the W3C corpus. For example, information from personal homepages in the *www* part could be added. Comparing the baseline, email-based profiles, to more extensive person profiles is planned as future work.

The only run we submitted for this task is based on simple queries against the constructed collection of person profiles:

```
//Person[about(.,X)]
```

[3]This is similar to the subject profiles used in the hard track, cf. Section 2.2.

Run	Query type	Combination
LMplaintext	`//DOC[about(.,X)]`	–
LMsubjectOR	`//DOC[about(.//SUBJECT, X) OR about(.//TEXT, X)]`	sum
LMsubjectAND	`//DOC[about(.//SUBJECT, X) AND about(.//TEXT, X)]`	product
LMheaderOR	`//DOC[about(.//HEADER, X) OR about(.//TEXT, X)]`	sum
LMheaderAND	`//DOC[about(.//HEADER, X) AND about(.//TEXT, X)]`	product

Table 2: Known-item runs

Run	Query type	λ
baseLMlam05	`//DOC[about(.,X)]`	0.5
baseLMlam08	`//DOC[about(., X)]`	0.8
LMMlam05Thr	`//THREAD[about(., X)]//DOC[about(., X)]`	0.5
LMlam08Thr	`//THREAD[about(., X)]//DOC[about(., X)]`	0.8

Table 3: Discussion search runs

9 Experimental Results

This section reports about the results from the three tasks in the Enterprise track. We used the same collection for all the experiments. That is, the original collection manipulated as described in Subsection 7.1 and a further tokenization using a stop word list and the Porter stemmer.[4]

9.1 Known-item search

Table 4 shows an overview of the evaluation scores for the five submitted runs described in Table 2.

If we look at the average reciprocal rank scores, we can see that there are not big differences between the runs. However, the combination of the subject and header information with the OR operator leads to better precision in the top 10, which might be useful for this type of task. Notice also that the use of the header information helps to find documents that other runs do not find. Further experiments are needed to quantise the importance of this type of information. The investigation of different implementations of the AND and OR operators is also an object of further study, as is the application of different weights to the structural fields.

9.2 Discussion search

Table 5 shows an overview of the evaluation scores for the four submitted runs described in Table 3.

[4]The numbers in this section differ from what is reported in the official result tables, since we re-ran our experiments after a bug-fix.

The use of thread information helps in the discussion search task. Initial precision may drop a bit, but for this task, that measure does not seem to be the most important one; when one is looking for pros and cons in a discussion, just a few arguments is usually not enough. In fact, the whole line of argument is important. Like in traditional text retrieval, the exact choice of lambda does not seem too important.

9.3 Expert search

For the expert search task we only tested a baseline run, based on the emails sent by the candidate experts. The MAP of this run, with $\lambda = 0.8$ is 0.1255 (lower values of λ give slightly worse results), this is around the median of all runs submitted for the expert task.

A study of the person-documents from which the profiles are created, already indicates that not all profiles can be very accurate, since for many persons few or no emails are found in the collection (either because those people did not contribute to the mailing lists, or because we failed to recognise their emails). The submitted email-only run is intended as a baseline for further research into using more extensive user profiles based on information from the whole collection. We have good hope that incorporating additional information in the candidate expert profiles will improve the results.

10 Enterprise Track Conclusions

We reported our approaches and experiments for the enterprise track. We investigated the effects

	LMplaintext	LMsubjectOR	LMsubjectAND	LMheaderOR	LMheaderAND
Avg. Reciprocal Rank	0.513	0.530	0.517	0.518	0.532
Found at position 1	52	51	51	49	53
Found in top 10	88 (70.4%)	97 (77.6%)	91 (72.8%)	96 (76.8%)	91 (72.8%)
Not found	15 (12.0%)	17 (13.6%)	17 (13.6%)	13 (10.4%)	14 (11.2%)

Table 4: Result overview: known-item search runs

	baseLMlam05	baseLMlam08	LMMlam05Thr	LMlam08Thr
MAP	0.3055	0.3040	0.3230	0.3273
R-prec	0.3389	0.3415	0.3557	0.3654
P10	0.4746	0.4814	0.4525	0.4610

Table 5: Result overview: discussion search runs

of using structural information for the different retrieval tasks. We showed that the thread information is an important factor when searching for pros and cos in a mailing list and that the use of the header and subject information can improve the effectiveness of the systems in different ways.

A PARAMETERISED SEARCH ENGINE

In past TREC evaluations, we have used MonetDB [4] as well as standard information retrieval software like the TNO VSM engine [6] and Lemur [10] to develop new applications of information retrieval technology. For these TREC participations, it was often necessary to re-implement parts of the existing system, such as reimplementing APIs, introducing new APIs, and sometimes introducing new indexing and storage structures. Of course, the development of research prototypes is a tedious and time-consuming job, but in our experience, deploying information retrieval software is a time-consuming job in *any* non-standard environment or application.

When deploying an information retrieval system, it is the application developer's job to translate the user query (usually just some keywords), to operations on inverted files and ranking operations on the results. This is easy when the application envisaged can be handled by some standard software components. However, standard solution are often not good enough. General purpose retrieval components, such as general purpose web search engines, do not provide sufficient functionality in many scenarios. For some time now, companies like Google have started to develop special purpose search solutions like Froogle (searching products) and Google Scholar (searching scientific articles). Today, the development of such specialised applications is the job of information retrieval specialists, but in the near future however, any software developer should be able to develop applications like this in pretty much the same way as he/she would currently develop office automation applications using relational database management systems: design a database schema; come up with SQL queries; make a nice user interface; done! We call these search engines of the future *parameterised search engines*. As future work, we will explore the possibilities for developing a *parameterised search engine*: a search engine providing a high-level query language that supports many diverse search applications, as well as providing flexible ranking of search results. This years' TREC and TRECVID experiments are a first step towards that goal.

References

[1] P. A. Boncz. *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*. Ph.d. thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, 2002.

[2] P. A. Boncz, et al. Pathfinder: XQuery-The Relational Way. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Trondheim, Norway, 2005. (Demo). Accepted for publication.

[3] W. Croft. Combining Approaches to Information Retrieval. In W. Croft (ed.), *Advances in Information Retrieval : Recent Research From the Center for Intelligent Information Retrieval*, pages 1–36. Kluwer Academic Publishers, New York, 2002.

[4] A. P. de Vries. The Mirror DBMS at TREC-9. In *Proceedings of the Nineth Text Retrieval Conference (TREC-9)*, pages 171–177. Gaithersburg, MD, USA, 2000.

[5] F. Diaz, R. Jones. Using Temporal Profiles of Queries for Precision Prediction. In M. Sanderson, et al. (eds.), *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–24. ACM, Sheffield, UK, 2004.

[6] D. Hiemstra, W. Kraaij. Twenty-One at TREC-7: ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 227–238. Gaithersburg, MD, USA, 1999.

[7] J. Kleinberg. Bursty and Hierarchical Structure in Streams. In *Data Min. Knowl. Discov.*, 7(4):373–397, 2003.

[8] W. Kraaij. *Variations on language modeling for information retrieval*. Ph.D. thesis, University of Twente, Netherlands, 2004.

[9] J. List, et al. TIJAH: Embracing IR Methods in XML Database. In *Information Retrieval*, 8(4):547 – 570, 2005.

[10] H. Rode, D. Hiemstra. Conceptual Language Models for Context-Aware Text Retrieval. In *Proceedings of the 13th Text REtrieval Conference Proceedings (TREC)*. 2005.

[11] F. Sebastiani. Text Categorization. In A. Zanasi (ed.), *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*, pages 109–129. WIT Press, Southampton, UK, 2005.

[12] A. Trotman, R. O'Keefe. The Simplest Query Language That Could Possibly Work. In *Proceedings of the 2nd INEX Workshop*. ERCIM Publications, 2004.

[13] A. Trotman, B. Sigurbjo"rnsson. Narrowed Extended XPath I (NEXI). In N. Fuhr, et al. (eds.), *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*, volume 3493. Springer-Verlag GmbH, 2005. Http://www.springeronline.com/3-540-26166-4.

[14] T. Westerveld, et al. An Integrated Approach to Text and Image Retrieval The Lowlands Team at TRECVID 2005. In *Workshop proceedings TRECVID 2005*. 2005.