Highlighting Relevant Passages for Users of the Interactive SPIDER Retrieval System

Daniel Knaus, Elke Mittendorf, Peter Schäuble, Páraic Sheridan
(knaus|mittendorf|schauble|sheridan)@inf.ethz.ch
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland

Abstract

We report here on our participation in the 1995 Text Retrieval (TREC 4) conference. This was limited to participation in the interactive searching task due to a re-implementation of our SPIDER retrieval system. We report on two aspects of the SPIDER system that are particularly useful for interactive searching, namely the use of fast query evaluation, and the use of passage highlighting in presenting retrieved documents to users. We also report on the lessons learned from observing people interact with our system and note the challenge faced by researchers wishing to evaluate the interactive use of information retrieval systems.

1 Introduction

Due to the evolution of the SPIDER system over the years, in part because of lessons learned from participation in past TREC conferences, and to allow a more seamless integration of SPIDER's different facets, we have this year undertaken a major redesign and re-implementation of the SPIDER retrieval system. A major objective of this redesign was to modularise the system based on client-server architecture so as to increase its flexibility and allow it to be easily adapted to the different types of tasks for which we use it—running our TREC experiments on the one hand, while also using it as a vehicle for our continuing research into multilingual and multi-media information retrieval. Because of our participation this year in the interactive track of TREC, we also took the opportunity to provide new search and document inspection functionality, like identifying and highlighting relevant passages of documents chosen for viewing.

Although we are now reaping the fruits of our extended redesign effort, we did not have it completed with sufficient time to load the TREC collection and perform the adhoc and routing tasks. This led us to concentrate on the interactive track for which we successfully completed the primary task. We therefore concentrate here on reporting the aspects of the SPIDER system that are specifically directed toward interactive searching, namely efficient query evaluation in section 2, and passage identification in section 3. The system functionality is summarized in section 4. We also report in section 5 on our experiences in performing the interactive task.

2 Fast Query Evaluation

For interactive searches we want to have efficient and effective retrieval, i.e.: *Find as many documents as possible relevant to the topic, without too much rubbish, in about 30 minutes.* We decided to use the best weighting scheme of last year's TREC as our basic method, namely the BM25(2.0, 0, ∞, 0.75) weighting of the Okapi team [Robertson et al., 1994].

$$RSV(q, d_j) = \sum_{\varphi_i \in q \cap d_j} a_{ij} b_i,$$

where

$$a_{ij} = \frac{\mathrm{ff}(\varphi_i, d_j)}{2(1/4 + 3/4 \cdot l_j/\Delta) + \mathrm{ff}(\varphi_i, d_j)},$$

$$b_i = \log\left(\frac{1/2 + n - \mathrm{df}(\varphi_i)}{1/2 + \mathrm{df}(\varphi_i)}\right) \cdot \mathrm{ff}(\varphi_i, q),$$

$\mathrm{df}(\varphi_i)$ = document frequency,

$\mathrm{ff}(\varphi_i, d_j)$ = feature (term) frequency,

l_j = number of tokens in the document d_j,

Δ = average of l_j over all documents,

n = number of documents in the collection.

Fast query evaluation is achieved by using an inverted file with the posting lists sorted by decreasing feature frequencies:

$$\varphi_i \mapsto \mathrm{df}(\varphi_i), \{(\mathrm{ff}(\varphi_i, d_j), d_j) | \mathrm{ff}(\varphi_i, d_j) > 0\}$$

Only the important parts of the posting lists (high feature frequency postings) are processed, less important parts are skipped. Furthermore, the document lengths are cached in main memory for faster access. We originally wanted to use the pruning algorithm of Persin [Persin, 1994] but like other speedup approaches

[Buckley & Lewit, 1985] this approach does not take into account the number of documents to be ranked, i.e. no matter how many documents are to be retrieved, these algorithms try to optimize the overall retrieval effectiveness.

Because there is in fact a tradeoff between the speed of query evaluation and the effectiveness achieved, we wanted to be able to tune our system to balance between fast and effective query evaluation. Although the system should guarantee a certain response time, we assume that users will accept slightly longer response times if they submit longer queries or ask for more documents to be returned in the ranked list. It was with this in mind that we formed our approach to efficient query evaluation.

If a query q were to be evaluated completely then $k_{tot} = \sum_{\varphi_i \in q} \mathrm{df}(\varphi_i)$ postings would have to be processed. We want to process only k postings ($k \leq k_{tot}$) such that the query is evaluated within a certain time (restricted number of disk accesses) while still ensuring that the list of the r top ranked documents is not significantly different from the beginning of the list produced by processing all k_{tot} postings. The response time is proportional to the number of postings we have to process, and the quality of the ranking produced is then dependant on the ratio k/k_{tot} and the number of requested documents, r. In our implementation we used the following simple formula to estimate k:

$$k = c \cdot |\vec{q}\,| \cdot r \qquad (1)$$

The constant c is used to tune the system: a small c allows fast but less effective retrieval and with a large c the query evaluation takes more time but the ranked list is better.

The k postings are selected from the k_{tot} postings such that the ranked list is most effective. The algorithm simultaneously works on the posting lists of all query features and processes as the next posting that which adds the highest estimated partial similarity $S_{\mathrm{nxt}}(\varphi_i)$ to a score. $S_{\mathrm{nxt}}(\varphi_i)$ is estimated from the last posting of φ_i that has been processed. Because the postings are ordered by decreasing $\mathrm{ff}(\varphi_i, d_j)$ we know that the feature frequency of the next posting must be equal or less than the current one. The length of the document of the next posting is estimated by the average document length (Δ).

The following pseudo-code illustrates our algorithm:

1. estimate k by (1)
2. initialization

```
for each φ_i ∈ q with df(φ_i) > 0 do
  open cursor on the posting list of φ_i
  compute b_i
  set S_nxt(φ_i) to b_i · a_i^max
endfor
create a structure of accumulators
set φ_h =argmax{S_nxt(φ_i)|φ_i ∈ q, df(φ_i) > 0}
```

3. accumulation

```
while (k > 0) and (S_nxt(φ_h) > 0) do
  get next posting (d_j, ff(φ_h, d_j)) of φ_h
  if found then
    increment accumulator A_j by b_h · a_hj
    set S_nxt(φ_h) to b_h · a_hj^nxt
  else
    set S_nxt(φ_h) to 0
  endif
  set φ_h =argmax{S_nxt(φ_i)|φ_i ∈ q, df(φ_i) > 0}
  decrement k
endwhile
```

4. identify r accumulators with highest values

Using the the current posting $(d_j, \mathrm{ff}(\varphi_i, d_j))$ we estimate the document weight a_{ik} of the next posting $(d_k, \mathrm{ff}(\varphi_i, d_k))$ by

$$a_{ij}^{\mathrm{nxt}} = \frac{\mathrm{ff}(\varphi_i, d_j)}{2 + \mathrm{ff}(\varphi_i, d_j)}$$

assuming that the document length of the next document d_k is Δ. For estimating a_{ik} before having processed any postings we use the upper bound

$$a_i^{\max} = 1.$$

For features φ_i with $\mathrm{df}(\varphi_i) = 0$ the estimated partial similarity $S_{\mathrm{nxt}}(\varphi_i)$ is initialized by 0.

Accesses to the inverted list of postings of a feature are cached of course, i.e. it is not necessary to access the disk each time another posting is to be read.

With the parameter c it is possible to choose between fast but less effective retrieval or effective but slower retrieval, whichever is preferred. It is even possible to change c between two query evaluations. This can be very helpful to quickly get an idea of the data and then to set up the query more carefully and evaluate it more effectively.

With respect to r, the estimation formula (1) is not yet realistic because the constant c does not have the same effect for low and high values of r. The influence of c on the retrieval effectiveness is subject to further investigations.

3 Passage Retrieval

In previous experiments [Knaus et al., 1994, Mittendorf & Schäuble, 1994] we have shown that Hidden Markov Models (HMM) provide a natural and effective method for retrieving relevant passages from documents. In this section we summarize the ideas of

HMM-based passage retrieval and we describe the refinements and changes we have made in the passage retrieval method in contrast to previous papers. Relevant passages of a document are identified and highlighted when the text of documents are presented to the user in the interactive search environment to help speed up decisions about the relevance or non-relevance of the document. Users can also mark relevant passages for relevance feedback, rather than using the features of the whole document. We use Hidden Markov Models to identify relevant passages as follows:

We assume that the document was produced by a HMM. In other words, a document is considered as a sequence of basic units produced by a finite state automaton, e.g. a sequence of tokens (occurrences of indexing features such as Porter-reduced non-stop-words [Porter, 1980]) or a sequence of sentences. The production of basic units in each state and the transitions outgoing from each state can be described with probability distributions. These probability distributions can be optimized on training material by applying the so-called Baum–Welch [Baum, 1972] training algorithm. As in other description oriented approaches for probabilistic retrieval [Fuhr, 1992], to get probability distributions which are tractable given a query, we map each basic unit to a value which describes the similarity between the basic unit and the query; for example if a single indexing feature is the basic unit then the similarity is described by $\mathrm{ff}(\varphi_i, q) \cdot \mathrm{nidf}^2(\varphi_i)$ [Knaus et al., 1994], where the normalized inverse document frequency is defined by $\mathrm{nidf}(\varphi_i) := 1 - \frac{\log(\mathrm{df}(\varphi_i)+1)}{\log(n+1)}$. We assume that there are certain states of a HMM which model the production of the passages that are relevant to the query and there are other states that are responsible for producing non-relevant passages. Given a document and a query and the corresponding sequence of similarity values, the Viterbi-algorithm [Viterbi, 1967] then enables us to find the most probable state sequence in the Hidden Markov Model which has produced this document. We identify as interesting passages those passages which are produced in the states of the HMM which are responsible for producing relevant passages.

We have recently adapted this approach of relevant passage identification to be more suitable to the interactive environment in which it is now used:

From Token to Sentence Descriptions:

Experiments have shown that it is difficult to provide meaningful similarity descriptions between a single token and a query. We therefore sought to find larger units to provide a more meaningful unit for measuring similarity with the query, but which retain the property that a sequence of such units still contains structure which can be modeled by a HMM. Sentences fulfill these requirements and so have been chosen as our basic units. Very rudimentary heuristics have been used to decide sentence boundaries in our texts. A sentence is simply considered to be a string s consisting of at least 15 characters such that the terminal character $s[N]$ must be a typical end-of-sentence delimiter like '?', '!' or '.'. To cater for certain types of abbreviation, we specify that if the sentence is terminated by a period ($s[N] =$'.') then $s[N-2]$ must be an alphanumeric character (e.g. to avoid false recognition of an end-of-sentence in "John F. Kennedy"). To compute a similarity value, the sentence s_k and the query q are preprocessed as usual by removing stop-words and performing Porter-reduction on the remaining words. A similarity value between s_k and q is then computed with ff · nidf-weighting and the scalar product.

$$\mathrm{sim}(q, s_k) := \sum_{\varphi_i \in q, s_k} \mathrm{ff}(\varphi_i, q) \cdot \mathrm{nidf}^2(\varphi_i) \cdot \mathrm{ff}(\varphi_i, s_k).$$

We have not yet made thorough evaluations to investigate whether sentential units lead to better effectiveness than token units. In a few experiments that we have conducted, it performed as well as token based retrieval. However, the use of sentential units does lead to better *efficiency*, so even for very long documents the passage retrieval is performed in a short time.

Continuous Hidden Markov Models:

Changing from discrete Hidden Markov Models to continuous Hidden Markov Models was mainly due to implementational reasons. We wanted to use the very flexible Hidden Markov Toolkit (HTK) [Young et al., 1993], which, at that time, supported only continuous Hidden Markov Models.

Different Model Layouts:

We experimented with different model-layouts for the Hidden Markov Models. A relevance model R and an irrelevance model I consist of only one state. Probability distributions were trained with a random selection of relevant or irrelevant query-document pairs from queries 101–200 and documents from the AP1 and AP2 collections.

We tried various different combinations of the relevance HMM R and the irrelevance HMM I in the Hidden Markov Model that models the document production:

- If documents are assumed to have exactly one relevant passage we take the model:

  $$[I]R[I]$$

  We adopt the notation for regular expressions from the HTK notation [Young et al., 1993]. The expression $[X]$ denotes an optional traversal of the model X. (Note: This model—though very restrictive—yielded the best results when using

passage level evidence in the routing or ad-hoc task for data from last year's TREC. We do not report the experiments here.)

- If we allow the document to have an arbitrary order of relevant and irrelevant documents we can choose the model

$$[I]\{R[I]\}$$

where $\{X\}$ denotes an arbitrary number of traversals, but at least one. It is distracting for some users that this model sometimes yields two or more consecutive relevant passages.

- We thus chose the following model for interactive searches

$$[I]\{RI\}[R]|IR|R$$

where $(X|Y)$ denotes the alternative of the two models X and Y.

4 System Functionality

The SPIDER System [Schäuble, 1993] is based on a client-server architecture. The retrieval server's tasks are kept to a minimum (inserting and deleting document descriptions, evaluating indexed queries), and a retrieval client (in the case of TREC-4) is used to provide the following search and inspection functionality:

- Evaluating queries: the queries are indexed by the client and then sent to the server. The result returned by the server is a ranked list consisting of document identifiers, retrieval status values and titles of the ranked documents.

- Inspecting documents: the documents are loaded from disk and passed to the passage retrieval component which is part of the client. The document is then displayed to the user with relevant passages highlighted for attention.

- Relevance feedback: The user can specify passages or whole documents as relevant. Features of relevant passages are stored and are added to the query when relevance feedback is performed. To limit the size of feedback queries, a maximum of the 50 highest weighted terms from the feedback query are considered.

Users presently interact with the retrieval client through a text-based user interface. This allows them to specify queries and view the ranked list of resulting documents, view the full texts of documents with passages judged relevant by the system appropriately marked, specify full documents as either relevant or non-relevant, or explicitly record marked passages as relevant, and perform relevance feedback so that the query is expanded with features from relevant passages (not documents) and resubmitted for evaluation. The full set of functions available to the users of our system are included in Appendix A.

5 Interactive Searching

For our participation in the interactive track we had 11 different people search for documents relevant to the TREC topics set aside for the interactive task. For 13 of the 25 interactive topics there was only one person who searchered for documents on that topic, ten of the topics were tried by two people, and in two cases three different searchers tried to find documents for a given topic. Of the 11 people who performed the interactive task for us, only three would consider English to be their native language, so the task of formulating queries for the TREC topics and skimming retrieved documents to judge relevance was difficult for many of the participants. (For example, many of our participants did not know what *"affirmative action"* was). The process of interactive searching that these searchers engaged in has essentially three parts, coinciding with the three classes of functionality of the system outlined in the previous section:

Query Specification: Searchers started out with the text of the TREC topic and were encouraged during training to specify a query that only included the important words or phrases from this description. They were also encouraged to add other related words to the query specification - to "express in as many ways as possible what you're looking for". Query specification also encompasses those points during an interactive session where a user feels that a dead-end has been reached on a given query formulation so a completely new query is given. The result of query evaluation was a ranked list of documents (doc-ids and titles) with their associated rank positions and RSV values. This list contained a default of 20 documents, but the user was free to adjust the number of documents displayed.

Viewing Retrieved Documents: Once a query had been evaluated the users would generally spend some time browsing the list of titles and would then choose to view the full text of documents that were of interest. The full text of a chosen document would be presented to the user, but passages of the text that were judged to be interesting by our system were marked and users were encouraged during training to direct their attention to these marked sections.

Judging Relevance: Based on either the title of a document in the ranked list, or on the full text of a

viewed document, a user could categorise the document as either relevant or non-relevant. Users were aware that it was much more important to find and mark relevant documents rather than non-relevant ones. If the full text of a document had been viewed, then the user could also explicitly mark highlighted passages as relevant. Only these marked passages were used in relevance feedback, not whole documents. Users were encouraged during training to perform relevance feedback as soon as they had one or two relevant passages.

The users therefore proceeded by iterating through these actions, using either relevance feedback or specifying a new query once they had browsed the full list of documents retrieved from an initial query. From observing the interactive sessions, we have concluded that the most effective search strategy seemed to involve specifying as precise a query as possible initially, with the objective of finding at least one or two documents with good relevant passages so that these relevant passages could be used through relevance feedback to retrieve more relevant material. In general, relevance feedback on passages appeared to work very well. Initial query formulation was therefore critical, as the most difficult task seemed to be finding that first good relevant document with which to launch feedback. Our starting advice on formulating long initial queries may therefore have gone against us, long initial queries seemed to retrieve documents that, although relevant to the topic in general, or rather to parts of the topic, were not directly relevant to the information need expressed. For example, when searching for information on *how affirmative action has affected the construction industry*, it was quite easy to find documents relating to affirmative action, and also easy to find documents reporting on the construction industry, but it was very difficult to find documents tying the two together.

In general, users were happy with the performance of the system if they got the chance to use relevance feedback, as this proved to be a very effective way of finding relevant documents once one or two relevant passages had been discovered. On the other hand, users found it extremely frustrating if they were unable to find any good relevant documents (i.e. with passages that they could mark relevant) with which to launch relevance feedback, as they generally found that query specification was the most difficult part of their task and wanted to avoid having to re-formulate queries after unsuccessful searches.

6 Conclusions

This was our first year of participation in the interactive track of TREC, and we feel it was a very valuable and rewarding experience - to watch different people use our system in trying to find documents relevant to given query topics, especially when the wording of the query topic and the SPIDER system together seemed to conspire to really test the searcher's patience. Our experiences with observing the interactive sessions left us very satisfied with the effectiveness of our fast query evaluation and passage retrieval reported on here. The fast query evaluation resulted in user queries being evaluated in a matter of two or three seconds, though the longer queries submitted through relevance feedback took longer than this. We also found that passage identification and highlighting was an efficient and very effective tool, especially for concentrating users' attention to the most important parts of documents. Given that passage identification was performed when a user chose to view the full text of a document, documents could still be presented to the user with highlighted passages within a matter of seconds. We were particularly pleased with the effectiveness of our decision to allow users to distinguish between documents that were relevant in general and specific passages that were relevant, and to use only user-marked relevant passages for feedback.

We note however, the lack of a definitive means of evaluating our system, for example in order to compare it to other systems that have been used in the interactive track of TREC 4. This lack of systematic evaluation is a result of both our time constraints due to our re-implementation efforts, and also a lack of a coherent evaluation plan for the interactive track of TREC. This is an evolving track and one which must address some very complex evaluation issues. We look forward to playing our role in this evolution.

If we were to choose a direction for further improvement in our interactive system it would certainly be to address the formulation and evaluation of initial queries. We hope to investigate in particular the use of query expansion techniques (e.g. [Qiu & Frei, 1993]), but this will go hand in hand with a detailed examination of the tradeoffs between efficiency and effectiveness resulting from our work described in section 2, so that we can be confident that our expanded queries will be evaluated efficiently while retaining an acceptable level of effectiveness.

Acknowledgements: The authors would like to thank Roger Aeppli and Philipp Häfliger for their help with implementation. We also want to thank all participants of the interactive track for taking part, namely Jean-Paul Ballerini, Sushil Bhattacharjee, Tore Bratvold, Dana Davis, Ulrich Gerloff, Oliver Lorenz, Arlette Piquet, Yonggang Qiu, Marc Roth, David Tonhofer, and Martin Wechsler.

References

[Baum, 1972] Baum, L. E. (1972). An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. *Inequalities*, 3,1–8.

[Buckley & Lewit, 1985] Buckley, C., & Lewit, A. F. (1985). Optimization of Inverted Vector Searches. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 97–110.

[Fuhr, 1992] Fuhr, N. (1992). Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255.

[Knaus et al., 1994] Knaus, D., Mittendorf, E., & Schäuble, P. (1994). Improving a Basic Retrieval Method by Links and Passage Level Evidence. In *TREC-3 Proceedings*, pp. 241–246.

[Mittendorf & Schäuble, 1994] Mittendorf, E., & Schäuble, P. (1994). Document and Passage Retrieval Based on Hidden Markov Models. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 318–327.

[Persin, 1994] Persin, M. (1994). Document Filtering for Fast Ranking. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 339–348.

[Porter, 1980] Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3),130–137.

[Qiu & Frei, 1993] Qiu, Y., & Frei, H. (1993). Concept Based Query Expansion. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 160–169.

[Robertson et al., 1994] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1994). Okapi at TREC-3. In *TREC-3 Proceedings*.

[Schäuble, 1993] Schäuble, P. (1993). SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data. In *ACM SIGIR Conference on R&D in Information Retrieval*, pp. 318–327.

[Viterbi, 1967] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13(2),260–269.

[Young et al., 1993] Young, S., Woodland, P., & Byrne, W. (1993). *HTK Version 1.5: User, Reference & Programmer Manual*. Entropic Cambridge Research Laboratory, Sheraton House, Castle Park, Cambridge CB3 OAX, England.

A Addendum to System Description for Interactive Experiments

A.1 System Description

Usable Features

- *New-Topic:* Begin a session searching for documents for a given TREC topic. The topic number is entered and the text of the topic is displayed on the screen.
- *Query:* Enter a query to the system. The current (previous) query (initially blank) is displayed to remind the user, but a completely new query must be entered. Any relevance judgements made in the current session are *retained* however; to begin with a completey clean slate, the command *New-Topic* must be given.
- *Display-Last-Query:* The text of the current (previous) query is displayed.
- *Feedback:* Perform relevance feedback using features from passages that have been judged relevant by the user. The list of marked relevant passages is maintained from the *New-Topic* command. The user can remove passages previously marked relevant by marking the document non-relevant.
- *Ranked-List:* Display the current ranked list of documents returned from the current (previous) query. The list includes the rank position, RSV value, Doc-id and document title. Documents are also marked as follows:
  - **V** The full text of the document has been viewed.
  - **R** The whole document has been marked as relevant.
  - **RP** A passage has been marked relevant.
  - **N** The whole document has been marked non-relevant.
- *More-Titles:* Display X documents in the ranked list. The default value is 20 documents.
- *Show-Document:* Present the full text of the chosen document. The document is first passed through the passage identification module so that interesting passages can be highlighted when the text is presented.
- *Mark-Relevant-Passage:* Mark a highlighted passage as relevant. The marked passage is noted for future relevance feedback and the whole document is marked relevant for the purpose of the interactive task.
- *Relevant-Doc:* Mark a document as relevant. This does not contribute to future feedback commands, but the document is saved for the interactive results.
- *Non-Relevant-Doc:* Mark a document as not relevant. Unless this is being used to reverse early relevance judgements (where a document or passage was earlier marked relevant), this has no real effect, other than to provide more information for analysis by us.
- *Exit:* End the session and leave the SPIDER system.

Style of Interface We used a text-based command interface for our experiments. It was sufficient to use the first two characters of the commands given above.

A.2 Experimental Conditions

Searcher Characteristics

We did not record detailed biographical information for each searcher. Their main characteristics are as follows:

- *Number of Searchers* 13 in total performed searches, the results of 11 different searchers were used in our final submission.
- *Number per Topic* Ranged from 1 to 3.
- *Searcher Ages*
  - (25 to 29) 9
  - (30 to 34) 1
  - (35 to 39) 1

- *Search Experience* None of our participants could be described as "professional" searchers, but rather as casual users of information retrieval systems. Although we had one librarian take part, none of her sessions were included in the submitted results (we submitted for each topic the searcher who marked most documents relevant). Many of our participants have some experience with information retrieval from a research perspective. One was a complete novice searcher, but was chosen because she was American and so had an edge on background knowledge over the European searchers.
- *Educational Level* All hold at least a Masters Degree or equivalent (Diplom), 3 also hold PhD degrees.
- *Undergrad Major*
  - 7 Computer Science,
  - 1 Engineering,
  - 1 Mathematics,
  - 2 Business.
- *Topic Familiarity*
  - 9 Minimal familiarity.
  - 2 Some background knowledge.
- *Work Affiliation*
  - 8 CS Dept. ETH Zurich,
  - 2 Union Bank of Switzerland (also computer people),
  - 1 Institue of Industrial Engineering and Management, ETH.

Task Description

Find as many documents as possible relevant to the topic, without too much rubbish, in about 30 minutes

The document collection consists of about 760,000 documents. Documents vary greatly in size, from some very short documents, to some very long documents.

The SPIDER system marks individual passages in documents that are found to be relevant. When reading a retrieved document, it is only necessary to look at those passages marked as relevant. Relevant passages are marked with "*1" (the first relevant passage).

It is not useful to specify *negative* search information, as one might use in a boolean system. Instead, try to focus your query more toward the exact topic, using additional search terms that will favour relevant documents.

Try to find as many ways as possible to express the query concept, for example by specifying a list of synonyms and related terms, rather than using a small set of search terms.

The SPIDER system has a *feedback* option that uses *passages* that you have marked as relevant to find other similar relevant documents. Once you have found two or three documents with relevant passages, the "feedback" command is an option to find more relevant documents.

Searching Tips

Spend some time planning your initial query. Your first query should be designed to retrieve a few relevant documents so that you can use these to focus your query and find more relevant documents.

- Use only those terms from the topic description that are central to the meaning: so terms like *"Find documents that..."* will not help.
- Understand the concept/idea of the topic and try to find related terms to add to your query.

Once you have a ranked list of documents, identify the relevant documents and use these to find others...

- Note the terms used in relevant documents. See if any of these terms might be added to your query. Resubmit a new query including the terms that you have found in good documents.
- Mark relevant passages as relevant and use them to perform "feedback". Note, only documents with *passages* marked as relevant are used for feedback (these are displayed as "RP" in the ranked list).

Training

The training process consisted of allowing the searchers time to read through the task description and the searching tips presented above, after which any questions or clarifications were dealt with. The searchers were then given a practice topic with which to practice. This was one of the official TREC interactive topics, but one that would not be given to that searcher during the actual task. We observed the searchers during their practice session and provided suggestions and guidance where we thought necessary.

The time taken to train each searcher was not accurately recorded. I estimate that the average time is somewhere around 45 minutes.

A.3 Search Process

Clock Time

(This was recorded in Minutes - Seconds are given here)

Mean	Median	SD	Range
1862.4	1860	218.732	1440 - 2280

Documents Viewed

We consider *viewing* a document as seeing the text of a document. We compiled the numbers for this by counting the number of times the command *show-document* was issued during a session. (If the same document was viewed twice, then it counts twice)

Mean	Median	SD	Range
31.76	29	9.479	20 - 54

Iterations per Search

We consider as an *iteration* either the submission of a query by the searcher, or the use of feedback. We present here results based only on the total number of iterations per session. Individual results for the use of the *Query* and *Feedback* commands may be found later in this section.

Mean	Median	SD	Range
7.36	6	5.971	2 - 30

Terms per Query

We consider here only those queries entered by the searcher (we don't in fact have data about the queries submitted through relevance feedback). Since we did not perform the secondary interactive task, there was no real *final query* as such, so we present here only figures computed over all queries, and then over all first queries.

Mean	Median	SD	Range
5.375	4	3.353	1 - 18

Terms in First Query

Mean	Median	SD	Range
6.36	5	2.659	3 - 13

Use of Features

Note that with our text-based user interface, it was necessary to issue the *Ranked-List* command to view the current ranked list of documents each time the full text of a document had been viewed so the ranked list was lost from the screen. This accounts for the high usage of this command.

Command	Mean	Median	SD	Range
Query	5.68	5	5.821	1 - 28
Display-Last-Query	0.48	0	1.357	0 - 5
Feedback	1.68	2	0.802	0 - 3
Ranked-List	19.96	21	5.086	7 - 27
More-Titles	0.68	1	0.69	0 - 2
Show-Document	31.76	29	9.479	20 - 54
Mark-Relevant-Passage	8.16	8	5.572	0 - 20
Relevant-Doc	5.96	5	5.465	0 - 21
Non-Relevant-Doc	9.04	5	10.179	0 - 45

Narrative of Search for Topic 236

This seems to have been one of the more frustrating topics for our searcher. For brevity, we do not show the complete ranked lists here.

Are current laws of the sea uniform? If not, what are some of the areas of disagreement?

- **Query** sea marine maritime law legal regulation disagreement dispute
- View rank position 1 (FR88720-0111): Judge not relevant.
- View rank position 2 (FR881121-0087): Judge not relevant.
- View rank position 3 (AP880725-0215): Judge not relevant.
- View rank position 4 (WSJ910626-0071): Judge not relevant.
- View rank position 7 (FR881116-0126)
- **Query** sea law marine legal regulation MARAD
- View rank position 2 (FR88927-0018)
- View rank position 23 (FR881027-0027): Judge *relevant.*
- **Query** sea law regulation ship
- View rank position 1 (FR88120-0060): Judge not relevant.
- View rank position 2 (FR881201-0054): Judge not relevant.
- View rank position 25 (FR88916-0076): Judge not relevant.
- **Query** sea law regulation ship sailing marine permit
- View rank position 4 (FR88506-0038): Judge not relevant.
- **Query** maritime law regulation
- View rank position 1 (FR88212-0072)
- View rank position 2 (FR88726-0078)
- View rank position 3 (WSJ910619-0001)
- **Query** international shipping act regulation law
- View rank position 3 (FR88719-0023): Judge not relevant.
- View rank position 10 (FR881121-0030): Judge *relevant.*
- View rank position 20 (FR88712-0019): Judge not relevant.
- View rank position 25 (FR88712-0040): Judge not relevant.
- **Query** marine law regulation
- View rank position 24 (FR88909-0031)
- **Query** maritime shipping international law
- View rank position 4 (FR881102-0046)
- **Query** maritime shipping fishing internation sea law regulation permit
- View rank position 1 (FR88909-0065)

- View rank position 8 (FR88212-0072)
- View rank position 1 (FR88909-0065): Judge not relevant.
- View rank position 2 (FR88511-0163): Judge not relevant.
- View rank position 16 (FR88923-0151)
- **Query** maritime shipping law regulation international permit dispute disagreement
- View rank position 5 (FR881109-0064): Judge not relevant.
- **Query** sea law shipping act
- **Query** maritime law legal
- **Query** shipping law legal
- View rank position 23 (AP900828-0128): *Mark Relevant Passage.*
- **Feedback**
- View rank position 25 (AP900830-0086)
- **Query** maritime law ship coast board cargo naval Lloyds
- View rank position 1 (AP880809-0243): *Mark Relevant Passage.*
- View rank position 2 (AP900709-0216): *Mark Relevant Passage.*
- View rank position 3 (FR881107-0125)
- **Feedback**
- View rank position 2 (AP880824-0253): Judge not relevant.
- View rank position 3 (AP901015-0272): Judge not relevant.
- View rank position 4 (AP900924-0256): Judge not relevant.
- **Exit!**
- **Total Elapsed Time: 28 minutes**
- **Accumulated Relevant Documents: 5**
  Note: According to the qrel file, none of these were actually relevant.