The University of Amsterdam at TREC 2003

Valentin Jijkoun Jaap Kamps Gilad Mishne Christof Monz
Maarten de Rijke Stefan Schlobach Oren Tsur
Language & Inference Technology Group
ILLC, University of Amsterdam
`http://lit.science.uva.nl/`

Abstract: We describe our participation in the TREC 2003 Question Answering, Robust, and Web 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 2003 we took part in the Question Answering, Robust and Web tracks. Our aim for the Question Answering track was to experiment with a new multi-stream architecture, in which we implemented 6 separate subsystems that each try answer questions in different ways. Withing the Question Answering track we also wanted to experiment with a dedicated biography question module that is currently in development. Our aim for the Robust track was to investigate the impact of blind feedback and stemming on poorly performing topics. Our aim for the Web track was to experiment with different document representations and retrieval models for the home/named page finding and topic distillation tasks.

For all three tracks, our experiments exploited the home-grown FlexIR document retrieval system [10]. The main goal underlying FlexIR's design is to facilitate flexible experimentation with a wide variety of retrieval components and techniques. FlexIR is implemented in Perl, and supports many types of pre-processing, scoring, indexing, and term-weighting methods. In particular, we used FlexIR's implementations of the `Lnu.ltc` weighting scheme, various language models, as well as the Okapi scheme; see the detailed descriptions of our efforts for each of the tracks below for the exact settings.

The rest of this paper is organized as follows. In three (largely self-contained) sections we describe our work for the Question Answering, Robust, and Web tracks. Finally, we summarize our findings in a concluding section.

2 Question Answering Track

Current Question Answering (QA) systems, as reflected by the ones taking part in the TREC QA track, can roughly be divided into two categories: *knowledge-intensive* systems, that make use of various linguistic tools for the question answering process, and *redundancy-based* systems, that rely on very high volumes of data (in many cases, the Web) and take a more shallow approach to text analysis.

Until last year, the University of Amsterdam was focused on the first approach, concentrating its QA efforts exclusively on *Tequesta* [11, 12], a linguistically informed QA system. This approach may be successful for some types of questions, but for others more shallow approaches seem more beneficial, and therefore this year we have expanded our QA work and implemented a *multi-stream* approach. While maintaining Tequesta as one possible QA method, we have developed other parallel systems that compete which each other to find the correct answer. These systems, or "streams," employ a range of redundancy-based and knowledge-intensive techniques.

This year we took part in the main QA task and in the passage QA task. For our participation in the main task we employed our new multi-stream architecture; for the passage task we relied on the Tequesta stream only.

2.1 The Main Question Answering Task

We now describe the approach we adopted for the main QA task; we devote separate subsections to factoid questions on the one hand, and list questions and definition questions on the other.

System Description

A general overview of our system is given in Figure 1. The system consists of 6 separate QA streams and a final answer selection module that combines the results of all streams and produces the final answers. An important practical benefit of this architecture is easy modification, maintenance, and testing of the different subsystems as well as easy integration of multiple source of information. Evaluation of the contribution of each approach to the entire QA process becomes a relatively simple task too. We now give a brief description of the different streams.

Table Lookup. This stream uses specialized knowledge bases constructed by preprocessing the collection. The stream exploits the fact that certain types of information (such as country capitals, abbreviations, and names of political leaders) tend to occur in a small number of fixed patterns. When a question type indicates that the question might potentially have an answer in these tables, a lookup is performed in the appropriate knowledge base and answers found there are assigned high confidence.

We hand-crafted a small number of regular expressions for extracting information about the categories listed in Table 1. For instance, the "Location" category concerns geographic information of the following type "Amu Darya, river, Turkmenistan, XIE19990811.0277," where the first field indicates a location, the second its type, the third a country or region in which it is located, and the fourth the identifier for the document from which it was extracted. "Geography" contains similar information, but without the type; "Leaders" has information of the following kind "Dutch, Foreign Minister, Jozias van Aartsen, XIE19991027.0270", and "Roles" generalizes this to also include other roles besides government-related ones. We used a number of external resources such as WordNet at various stages of the extraction process, for instance, to find professions or different manners of death.

Table 1: Facts extracted from the AQUAINT corpus.

Category	# Facts	Category	# Facts
Abbreviations	31737	Birthdates	9156
Capitals	1273	Currencies	231
Dates	9331	Deathdates	1510
Geography	70363	Height	15603
Inhabitants	2025	Languages	853
Leaders	18073	Locations	1348
Manners of death	857	Organizations	98758
Roles	396558		

When a question is classified as possibly having an answer in a table, we first identify the question keywords that will be used in the table search. Next, a line matching all of the words in the order they appeared in the question is searched; if no line matches, we look again for a line containing all words, this time in any word order. If there is still no match, we start removing words from the list of words to match; the order of removal is based on the frequency of words in the language (i.e., common words are removed first) and part-of-speech tags (e.g., superlatives like *fastest*, *largest* are removed last). We do this until some threshold is reached (percentage of lookup words out of total keywords in the question). When a matching line is found, we return the text in the column that is declared to contain the information required as the answer.

Pattern Matching. This stream exploits the fact that in some cases, the contextual format of an answer to a question can be back-generated from the question itself. For example, an answer to a question such as *2257. What is the richest country in the world?* will possibly match the pattern `<Capitalized-Words>(,| is) the richest country in the world`. In these cases, the position of the answer within the context is also known when generating the context pattern; in the given example, it would be the capitalized word or words (and indeed, in document XIE19980302.0146, this pattern matches against "... Although the United States is the richest country in the world, 20 percent of its population ...").

The Pattern Matching stream consists of three stages: *Generation*, *Document Prefetch* and *Matching*. In the Generation stage, the question is analyzed and possible answer patterns are generated. For questions like *2347. Where is Mount Olympus?* the question type and focus (both provided by the question classifier) are

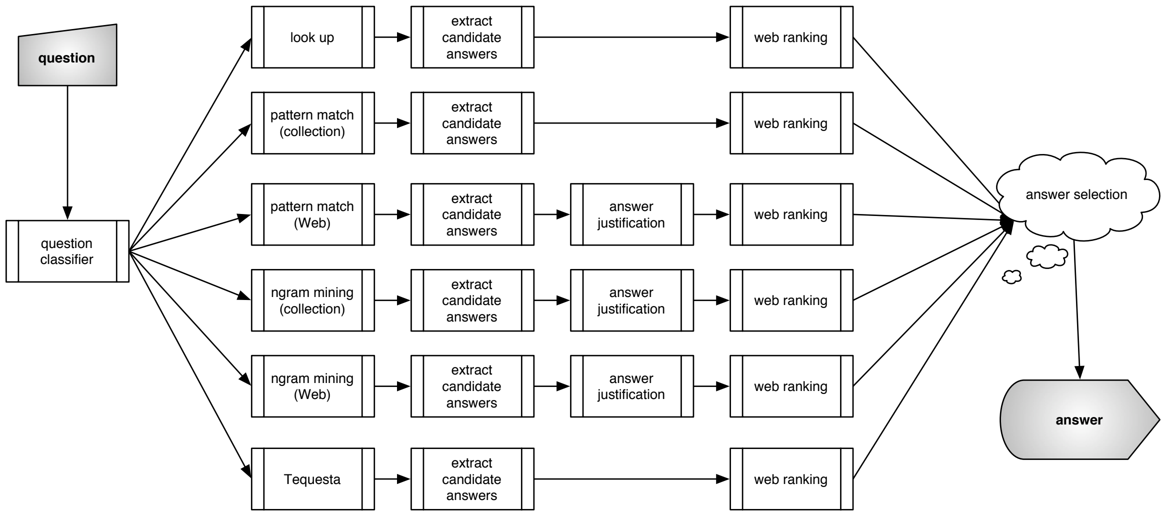

Figure 1: Quartz-e System Overview.

sufficient for generating a number of answer patterns. For other questions (e.g., *2375. What date did Thomas Jefferson die?*) we also use a set of manually created rules based on part-of-speech tags of the question words and a dictionary of word forms, in order to rewrite the question into declarative forms (e.g., `Thomas Jefferson (died|dies) (on|in) <answer>`). In the Prefetch stage, for each generated pattern a query containing words from it is formed, and documents are retrieved from the collection using the query. In the final stage, the patterns are matched against the retrieved documents, and answers are extracted from the matches.

Two variations of this stream were implemented, *Web Pattern Matching* and *Collection Pattern Matching*. For the first variation the text collection was the Web, and for the second, the local AQUAINT corpus. For the prefetch stage we used the top-ranking documents from Google (for the Web variation) and all matching documents retrieved using a boolean query to our document retrieval engine FlexIR [10] against from the AQUAINT corpus.

Ngram Mining. This stream, similar in spirit to [3], constructs a weighted list of queries for each question using a shallow reformulation process, similar to the Pattern Match stream. The queries are then sent to a large document collection; we implemented two variations for this stream, *Web Ngram Mining* and *Collection Ngram Mining*, using the Web and the local AQUAINT corpus, respectively. For Web searches, we used Google, and for local searches FlexIR, with the `Lnu.ltc` weighting scheme. Then, we looked at word ngrams in the relevant retrieved document paragraphs (for the Web we used the snippets provided by Google, and for the collection we used a window of 200 bytes around the query). The ngrams were ranked according to the weight of the query that generated them, their frequency in the paragraphs, their NE type, the proximity to the query keywords and more parameters, and the top-ranking ngrams were considered answer candidates. To find justification for the answer in the local corpus, we constructed a query with keywords from the question and the answer, and considered the top-ranking document for this query to be the justification, this time using an Okapi model as this tends to do well on early high precision in our experience.

Tequesta. As mentioned before, this is a stream that implements a linguistically informed approach to QA. We defer a discussion of this stream to Subsection 2.2 where we describe our strategy for the passage task.

Many components are shared by all streams, including a locally developed named entity tagger and the following:

Question Classifier. An incoming question is first analysed for its type (e.g., `date-of-birth`), expected answer type (e.g., `location`) and focus (the *core* of the question, used e.g., for answer pattern generation). Currently our system recognizes 37 question types. The question analysis is based on manually created surface and part-of-speech patterns. We also use the hierarchical relations in WordNet to identify semantic classes of question focus words (e.g., this allows us to assign the type `person-ident` to the question *1943. What is the name of Ling Ling's mate?*).

Web Ranking. The different candidates produced by the streams have different confidence levels, according to stream-specific internal parameters and different measuring methods. To compare these different levels, a uniform way of ranking the candidates was required. For this reranking we implemented a search engine hit count module, similar to [8].

Answer Selection. Each of the six streams produces a pool of answer candidates, with confidence scores normalized using hit counts. After filtering the candidates to remove obvious noise, we create a joint pool of answers, adjusting each candidate's score by a factor that reflects the past performance of its stream on questions of the same type. We tried different ways of assigning these stream/question-type weights: manually (i.e., based on human intuition about how good different streams perform on different questions) and automatic (using Machine Learning to find weights that optimize the performance of the system on a training set of questions). See below for a discussion of our findings. Then, in the joint pool of answer candidates we identify identical or similar (small edit distance) answers, merge and add their confidence scores. Finally, a candidate with the highest score is returned.

List and Definition Questions

Because of time constraints, we were unable to implement a proper module for handling list questions. All list questions were automatically rewritten into factoids using rule-based tranformations (e.g., *2097. Which countries were visited by first lady Hillary Clinton?* was transformed to *Which country was visited by first lady Hillary Clinton?*) and fed to our multi-stream QA system. The top *N* candidate answers to this factoid question were submitted as answers to the original list question. We experimented with different values of *N* (10 and 20 in our official runs) and with different numbers of retrieved documents used during answer selection (both for collection- and web-based QA streams).

In contrast to list questions, we did invest a serious effort in developing a component for handling definition questions. More precisely, we piggybacked on ongoing inhouse activities aimed at developing a QA system for handling "biography oriented" definitions on the web. The main steps in our handling of definition questions are Question Analysis (very similar to the analysis carried out for factoids), Answer Retrieval (always from external resources), Answer Filtering, and Answer Justification (very similar to the justification performed for externally found answers to factoid questions).

For concept definition questions we followed a WordNet-based strategy suggested in the literature by teams handling concept definition questions in earlier editions of TREC [13]. Given a question that asks for a definition of a concept, we simply consult WordNet. As our primary strategy for handling person definition questions, we also consulted an external resource. The main resource used is `biography.com`. However, in many cases no biography could be found in this resource. In such cases we backed off to using Google, with queries obtained by combining the name of the person in question with varying subsets of a predefined set of hand-crafted features (including "born", "graduated", "suffered", etc.) For questions asking for definitions of organizations the latter was the strategy used (with a set of "organization features").

As a final fallback option for each type of definition question, if the use of the strategies mentioned earlier returned no satisfactory results, we simply submitted `<question term> is a` to Google and mined the snippets returned. This method worked surprisingly well for questions like *2385. What is the Kama Sutra?*.

Given a set of candidate answer snippets, we performed two more steps before carrying out the final answer justification step: we separated junk snippets from valuable snippets and we identified snippets whose content is very similar. We addressed the first step by analyzing the distances between query terms submitted to the search engine and the sets of features, and by means of shallow syntactic aspects of the different features such as sentence boundaries. To address the second step we developed a snippet similarity metric based on edit distance, stemming, stopword removal, and keyword overlap.

Runs

We submitted 3 runs. These runs used the exact same strategies and settings for definition questions. They did differ in their settings for factoids and list questions. Here's a brief description:

UAmsT03M1 For factoids, the answer selection module used automatically learned stream/question weights; answers coming only from external sources (streams based on Web) were justified against the AQUAINT collection using the Okapi model. For each list question the top 10 answers to its factoid counterpart were submitted.

UAmsT03M2 For factoids, the weights for answer selection were learned automatically; external answers were discarded. For list questions the number of collection and web documents used for answer mining was increased, and the top 20 answers were submitted for each question.

UAmsT03M3 Manually assigned weights were used for answer selection; external answers were discarded. The number of documents for answering list questions was as in UAmsT03M2, but only top 10 answers were submitted.

Our three runs allowed us to compare the impact of justification, and the impact of using manually assigned versus learned weights for our answer selection. For the list questions we wanted to evaluate the effect of *using more data* and of *giving more answers* on the final performance.

Results

Table 2 gives the detailed results of our system for the 413 factoid questions: accuracy and the number of correct (R), unsupported (U), inexact (X) and wrong (W) answers.

Table 2: Results for the QA track (factoid questions).

Run identifier	Accuracy	R	U	X	W
UAmsT03M1	0.136	56	22	32	303
UAmsT03M2	0.145	60	20	26	307
UAmsT03M3	0.128	53	24	30	306

The results for the factoids are disappointing. Our preliminary error analysis reveals that out of 413 factoids only 192 (46%) had a correct answer among answer candidates extracted by the system, which means that either at retrieval or at candidate extraction stage we already miss too many answers. Moreover, only for half of the questions where the right answer was among the candidates (98 out of 192) it was actually selected as the final answer, indicating serious problems with candidate scoring. Yet more, for our best run out of 110 "not wrong" (i.e., correct + unsupported + inexact) answers only 60 (55%) were judged "correct."

A few more remarks are worth making. First, although the run with the automatically learned weights for answer selection from multiple streams (UAmsT03M2) outperformed the run with manually assigned weights (UAmsT03M3), our subsequent experiments revealed that whereas a small difference exists, it is not statistically significant. On the other hand, both run improve significantly over a baseline system with equal weights to all streams.

We also evaluated the contribution of different streams to the performance of the system on the factoids (using unofficial answer patterns). Table 3 gives the results (the number of "correct" answers, i.e. those that match the patterns) for the whole system, for separate streams and for the system with one of the streams turned off.

As expected, each of the six streams answered some questions correctly and more interestingly, each stream contributed to the overall performance of the system. The two "worst" performing streams (predictably, collection-based pattern matching and ngram mining) brought one more answer each either at the top rank or in the top 5. Suprisingly, the "winner" among the streams is equivocal: while *Table Lookup* allows the system to answer 15 questions more, *Web Ngrams* accounts for more (35 vs. 19) unique correct answer candidates in the top 5.

Table 4 gives the combined results for the 3 QA tasks (accuracy for factoids, F score for list and definition ques-

Table 3: Contribution of different streams.

Configuration	# correct	# correct in top 5
All streams	98	165
Collection ngrams	39	42
Without collection ngrams	98	164
Web ngrams	65	115
Without Web ngrams	89	130
Collection patterns	39	39
Without collection patterns	97	165
Web patterns	51	59
Without Web patterns	94	163
Table lookup	71	77
Without table lookup	83	146
Tequesta	63	102
Without Tequesta	91	140

tions) and the final scores of our runs.

Table 4: Results for the QA track.

Run identifier	A (Fact)	F (List)	F (Def)	Overall
`UAmsT03M1`	0.136	0.054	0.315	0.160
`UAmsT03M2`	0.145	0.042	0.308	0.160
`UAmsT03M3`	0.128	0.035	0.292	0.146

The results for the list questions suggest that using more retrieved documents for answer extraction and submitting more answer candidates hurts performance: the increase in recall does not compensate for the drop in precision.

Turning to definition questions now, recall that there is *no* difference between the three runs listed in Table 4 as far as definition questions are concerned, despite the different scores in the table. The differences are due to inconsistencies in the judgments provided by NIST. Table 5 provides a breakdown of the scores for the different types of definition questions; the highest scores are obtained for person definitions, which reflects the fact that those are the type of definition questions in which we put most work.

Table 5: Breakdown of F scores for definition questions.

Run identifier	Concept	Person	Org.	Overall
`UAmsT03M1`	0.150	0.392	0.268	0.315

As an aside, in our submission we found *no* answer for 19 of the 50 definition questions. If we compute the F score not over all 50 question but only over questions with a positive F score, we obtain an average of 0.527. In post-submission experiments we changed the subsets of features we use in the queries sent to Google as well as the number of queries/subsets we use. This resulted in a reduction of unanswered definition questions to 6 instead of 19. Using our own (unofficial) assessment, this yielded an F score of 0.688.

Conclusions

Our general conclusion on answering factoid questions is that our new multi-stream approach helped answer considerably more questions than our “old” single-stream Tequesta system. This year’s questions seem *much* harder than those of previous years. A preliminary error analysis shows that retrieval, named entity recognition, and answer selection all require further attention. Our main conclusion on answering definition questions is that external dictionary-like resources are crucial for this type of questions, but a feature-based approach offers an effective strategy in case such resources are absent or too sparse.

2.2 Passage Task

The aim of the passage task was to return an excerpt from a document rather than an exact answer. Excerpts had to be unmodified snippets from a document in the AQUAINT collection, and were not allowed to be longer than 250 characters. For the passage task only the factoid questions from the main task were used, i.e., list and definition questions were not included.

System

For the passage task, we used a modification of the Tequesta question answering system. Tequesta, both as it was used this year as one of the streams of the Quartz-e question answering system as it has been used at previous TREC QA tracks, see [11, 12], returns an exact answer, but for the passage task, we dropped this constraint, and included some of the context surrounding the answer identified by Tequesta.

The Tequesta system itself has remained largely unchanged since last year’s TREC; see [12] for a more detailed description. This year, we added the use of minimal span weighting for identifying documents that are likely to contain an answer to a given question. We used minimal matching spans as the textual units in which the exact answer is to be found.

Minimal span weighting takes the positions of matching terms into account, but does so in a more flexible way than passage-based retrieval; see [9] for a mored detailed discussion and evaluation of minimal span weighting. Intuitively, a minimal matching span is the smallest text excerpt from a document that contains all terms which occur in the query and the document. More formally:

Definition 1 (Matching span) Given a query q and a document d, where the function $\text{term\_at\_pos}_d(p)$ returns the term occurring at position p in d. A *matching span* (ms) is a set of positions that contains at least one position of each matching term, i.e. $\bigcup_{p \in \text{ms}} \text{term\_at\_pos}_d(p) = q \cap d$.

Definition 2 (Minimal matching span) Given a matching span ms, let b_d (the beginning of the excerpt) be the minimal value in ms, i.e., $b_d = \min(\text{ms})$, and e_d (the end of the excerpt) be the maximal value in ms, i.e., $e_d = \max(\text{ms})$. A matching span ms is a *minimal matching span* (mms) if there is no other matching span ms$'$ with $b'_d = \min(\text{ms}')$, $e'_d = \max(\text{ms}')$, such that $b_d \neq b'_d$ or $e_d \neq e'_d$, and $b_d \leq b'_d \leq e'_d \leq e_d$.

The next step is to use minimal matching spans to compute the similarity between a query and a document. Minimal span weighting depends on three factors.

1. *document similarity*: The document similarity is computed using the Lnu.ltc weighting scheme, see Buckley et al. [1], for the whole document; i.e., positional information is not taken into account. Similarity scores are normalized with respect to the maximal similarity score for a query.

2. *span size ratio*: The span size ratio is the number of unique matching terms in the span over the total number of tokens in the span.

3. *matching term ratio*: The matching term ratio is the number of unique matching terms over the number of unique terms in the query, after stop word removal.

The msw score is the sum of two weighted components: the normalized original retrieval status value (RSV), which measures *global similarity* and the spanning factor which measures *local similarity*. Given a query q, the original retrieval status values are normalized with respect to the highest retrieval status value for that query:

$$\text{RSV}_n(q,d) = \frac{\text{RSV}(q,d)}{\max_d \text{RSV}(q,d)}$$

The spanning factor itself is the product of two components: the span size ratio, which is weighted by α, and the matching term ratio, which is weighted by β. Global and local similarity are weighted by λ. The optimal values of the three variables λ, α, and β were determined empirically, leading to the following instantiations: $\lambda = 0.4$, $\alpha = 1/8$, and $\beta = 1$. Parameter estimation was done using the TREC-9 data collection only, but it turned out to be the best parameter setting for all collections.

The final retrieval status value (RSV') based on minimal span weighting is defined as follows, where $|\cdot|$ is the number of elements in a set:

Definition 3 (Minimal span weighting) If $|q \cap d| > 1$ (that is, if the document and the query have more than one term in common), then

$$\text{RSV'}(q,d) = \lambda \cdot \text{RSV}_n(q,d) + (1-\lambda) \cdot \left(\frac{|q \cap d|}{1 + \max(mms) - \min(mms)}\right)^{\alpha} \cdot \left(\frac{|q \cap d|}{|q|}\right)^{\beta}.$$

If $|q \cap d| = 1$ then $\text{RSV'}(q,d) = \text{RSV}_n(q,d)$.

At this point it may be helpful to illustrate the formal definitions by considering question *1395. Who is Tom Cruise married to?* After stop word removal and applying morphological normalization, the query q=\{*cruise*, *marri*, *tom*\}. Assume that there is a document d with terms matching at the following positions: $\text{pos}_d(\textit{cruise})$ = \{20, 35, 70\}, $\text{pos}_d(\textit{marri})$ = \{38, 80\}, and $\text{pos}_d(\textit{tom})$ = $\emptyset$. Then, the minimal matching span (mms) = \{35, 38\}, the span size ratio is $2/(1+38-35) = 0.5$, and the matching term ratio is $2/3$. Taking the latter two and the proper instantiations of α and β, the spanning factor is $0.5^{1/8} \cdot 2/3 = 0.611$. If the global (normalized) similarity between q and d is n ($0 < n \leq 1$), for instance $n = 0.8$, and $\lambda = 0.4$, the final msw-score for q and d ($\text{RSV'}(q,d)$) is $0.4 \cdot 0.8 + 0.6 \cdot 0.611 = 0.6866$.

Given a minimal matching span, the document analysis component of Tequesta tries to identify a phrase which is of the appropriate type. All phrases that are of the appropriate type are considered candidate answers. Within Tequesta, answer selection is accomplished by considering the frequency of a candidate answer. Most of the procedures that identify candidate answers rely on linking a candidate to the question by proximity. Hence, all candidate answers are weighted equally. But there is one exceptions. If the question is of type `what-np`, candidate answers that are in a WordNet hypernym relationship with the question focus receive a higher weight than candidate answers that are identified by means of the fallback strategy. In the second case, the weight of the candidate answer is actually not based on the confidence with which it is linked to the question, but on the confidence that this phrase is indeed an instance of the question focus.

Once an answer has been selected, the corresponding minimal matching span from which the answer has been extracted is returned as the answer passage. If the passage is longer than 250 characters, it is trimmed down to the appropriate length.

Results

We submitted one run to the passage task, run id `UAmsT03P1`. The results are shown in Table 6.

Table 6: Results for the QA passage track

Run identifier	Accuracy	R	U	W
`UAmsT03P1`	0.111	46	6	361

(R) stands for passages that contained a correct and exact answer, (U) for passages that contained the correct answer, but were not supported by the corresponding document, and (W) stands for wrong answers. The passage track does not make a distinction between exact and inexact (X) answers, as in the main task. Here, an inexact answer is simply judged wrong (W).

Conclusions

The results were quite disappointing. At this point we are not sure what caused this rather bad performance. Before submitting this year's run to the passage track, we conducted some experiments on the question sets from previous TRECs, and these results were substantially better. Therefore, one explanation could be that this year's question set was much harder than the previous ones, but a more detailed error analysis remains to be done.

3 Robust Track

In this section, we will discuss our official runs for the Robust Track. After describing the experimental setup for this track, we discuss our runs investigating the impact of blind feedback and stemming on the poorly performing topics.

System Description

All robust track runs use the home-grown FlexIR information retrieval system. We employ a number of techniques:

Tokenization We remove punctuation marks, apply case-folding, and map marked characters into the unmarked tokens. We either index the words or stems of the words. We use the Snowball stemming algorithm [16]. Snowball is a small string processing language designed for creating stemming algorithms for use in information retrieval

Retrieval model We use a language model [5]. For all the robust track runs, we use a uniform query term importance weight of 0.15.

Blind feedback Term weights are recomputed by using the standard Rocchio method [15], where we consider the top 10 documents to be relevant and documents ranked 501–1000 to be non-relevant. We allow at most 20 terms to be added to the original query.

We use the Title and Description fields of the topics, and investigate whether

1. Snowball stemming; or

2. Rocchio blind feedback; or

3. both stemming and blind feedback,

help retrieval for the lowest performing topics.

Runs

We submitted the following five official runs:

`UAmsT03RDesc` Language model run on a word-based index, using only the description-field of the topics. This is our mandatory description-only run.

`UAmsT03R` Language Model run on a word-based index. This runs serves as the baseline for our stemming and feedback experiments.

`UAmsT03RFb` Language model run on a word-based index, using Rocchio blind feedback.

`UAmsT03RSt` Language model run on the Snowball stemmed index.

`UAmsT03RStFb` Language model run on the Snowball stemmed index, using Rocchio blind feedback.

Results

Table 7 gives the results of the five official runs over all 100 robust topics (best scores in boldface). The second column shows the mean average precision, the third column the precision at 10 documents, the fourth column the percentage of topics with no relevant document in the top 10; the fifth column shows the area underneath the MAP(X) versus X curve for the worst 25 topics.

Table 7: Results for the robust track.

Run identifier	MAP	Prec.10	no Top10	MAP(X)
`UAmsT03RDesc`	0.2065	0.3530	15.0%	0.0076
`UAmsT03R`	0.2324	0.4050	9.0%	0.0216
`UAmsT03RFb`	**0.2452**	0.4110	13.0%	0.0210
`UAmsT03RSt`	0.2450	**0.4150**	**6.0**%	0.0256
`UAmsT03RStFb`	0.2373	0.4040	14.0%	**0.0273**

The results of blind feedback are mixed. On the one hand feedback helps precision at 10 and gives the best score for mean average precision. On the other hand feedback hurts the performance on the worst scoring topics.

The results for Snowball stemming are positive overall. Stemming helps both the overall performance, with a best score for precision at 10, as well as the performance of the worst scoring topics, with a best score for the percentage of topics with a top 10 relevant document. The use of both stemming and feedback gives the best score for the area under the MAP(X) curve, but does not promote performance on the other measures.

We also break down the score over the 50 old topics (in Table 8) and the 50 new topics (in Table 9). Note that the area underneath MAP(X) versus X curve (in the last column) is now calculated for the worst 12 topics. For both the old and new topics, the effectivenes of feedback and stemming is comparable to the effectiveness on all topics. There is, however, a striking difference in the performance between the two types of topics: the new topics give a much higher mean average precision score. This is an obvious consequence of the way the old topics were selected for inclusion in this year's robust track. As a result, the worst topic measures are dominated by the old topics.

Table 8: Results for the old topics.

Run identifier	MAP	Prec.10	no Top10	MAP(X)
`UAmsT03RDesc`	0.1066	0.2640	14.0%	0.0064
`UAmsT03R`	0.1349	0.3180	12.0%	0.0142
`UAmsT03RFb`	**0.1377**	0.3200	16.0%	0.0143
`UAmsT03RSt`	0.1327	**0.3300**	**6.0**%	0.0185
`UAmsT03RStFb`	0.1361	**0.3300**	16.0%	**0.0204**

Table 9: Results for the new topics.

Run identifier	MAP	Prec.10	no Top10	MAP(X)
`UAmsT03RDesc`	0.3064	0.4420	16.0%	0.0142
`UAmsT03R`	0.3300	0.4920	**6.0**%	0.0433
`UAmsT03RFb`	0.3528	**0.5020**	10.0%	0.0368
`UAmsT03RSt`	**0.3572**	0.5000	**6.0**%	**0.0551**
`UAmsT03RStFb`	0.3386	0.4780	12.0%	0.0478

Conclusions

Our general conclusion is twofold: although feedback helps overall performance, it does not help improve the score of the lowest scoring topics, but stemming turns out to be an effective strategy for improving the worst scoring topics.

4 Web Track

In this section, we discuss our official runs for the Web Track. We investigate the impact of various document representations and retrieval models for web retrieval. After describing our experimental setup for this track, we discuss our runs for the home/named page finding task (known-item search), followed by the runs for the topic distillation task (key resource search).

System Description

All web track runs use the home-grown FlexIR information retrieval system. We employ a number of techniques:

Document representation We create indexes for (1) the full documents, (2) the text in the title tags, (3) the anchor texts pointing towards the document. For the anchor texts index, we unfold relative links and normalize URLs, and do not index repeated occurrences of the same anchor text [12].

Tokenization We remove HTML-tags, punctuation marks, apply case-folding, and map marked characters into the unmarked tokens. We either index the free-text without further processing, or use the Snowball stemming algorithm [16].

Retrieval model We use three retrieval models. First, a statistical language model [5] with a uniform query term importance weight of either 0.35 or 0.70. Second, the Okapi weighting scheme [14] with tuning parameters $k = 1.5$ and $b = 0.8$. Third, the `Lnu.ltc` weighting scheme [1] with *slope* at 0.1 or 0.2; the pivot was set to the average number of unique words per document.

Combination We use the standard combination methods such as CombSUM and CombMAX [4], or weighted fusion [17]. We combine either full length runs, or limit the combination to the top n results. Unless indicated otherwise, we normalize the scores before combining them.

Minimal span weighting We calculate a minimally matching span for each document, as detailed in Section 2.2; see also [2].

In two separate sections, we will now address our runs and results for the home/named page finding task, and the topic distillation task.

4.1 Home/Named Page Finding Task

Runs

We submitted the following five official runs for the home/named page finding task:

`UAmsT03WnOWS` CombSUM of top 1000 of Okapi on word-based and stemmed full document indexes.

`UAmsT03WnLM` Language model run ($\lambda = 0.70$) on word-based full document index.

`UAmsT03WnLn3` CombMAX on the top 25 of `Lnu.ltc` runs ($slope = 0.2$) on the three stemmed indexes: full documents, titles, and anchor texts.

`UAmsT03WnLM3` Weighted fusion of language model runs ($\lambda = 0.70$) on the three word-based indexes: 0.7 full documents, 0.2 titles, and 0.1 anchor texts.

`UAmsT03WnMSW` Minimal span weighting based on the `Lnu.ltc` run ($slope = 0.1$) on the stemmed full document index.

Results

The results of the official runs for the home/named page finding task are shown in Table 10 (best scores in boldface). The second column gives the mean reciprocal rank,

Table 10: Results for home/named page finding.

Run identifier	MRR	Top 10	not found
`UAmsT03WnOWS`	0.3833	178 (59.3%)	70 (23.3%)
`UAmsT03WnLM`	0.3592	170 (56.7%)	81 (27.0%)
`UAmsT03WnLn3`	0.4982	**218** (72.7%)	**38** (12.7%)
`UAmsT03WnLM3`	**0.5185**	214 (71.3%)	46 (15.3%)
`UAmsT03WnMSW`	0.4073	189 (63.0%)	64 (21.3%)

the third column the number and percentage of topics with a relevant document in the top 10, the fourth column the number and percentage of topics for which no relevant document is found (in the top 50). The language model run combining the non-stemmed documents, titles, and anchors scores best with an average reciprocal rank of 0.5185. The `Lnu.ltc` weighted combination of the three stemmed indexes scores second best.

Table 11 shows the mean average precision of the base runs used in combinations for our official runs. All

Table 11: MRR for home/named page finding base runs.

Index type		Lnu.ltc	Okapi	LM
Documents	Words	0.3750	0.3795	0.3604
	Stems	0.3697	0.3833	0.3616
Titles	Words	0.2339	0.3421	0.3536
	Stems	0.3655	0.3334	0.3487
Anchors	Words	0.3068	0.3593	**0.4436**
	Stems	0.2934	0.3379	0.4278

`Lnu.ltc` runs use a slope of 0.2, and all language model runs use a uniform term weight of 0.70. Here, we retrieve up to 1,000 documents per topic, leading to slightly higher MRRs than the official runs using a maximum of 50 documents. We see an interesting difference between the three retrieval models: where the `Lnu.ltc` and Okapi models score best on the full document representation, the language model runs on the anchor text index score more than 20% better than the runs on the full document index. In fact, our best score on a single index is on the language model run on the non-stemmed anchor text index. There is no clear benefit of the use of a stemming algorithm on the mean reciprocal ranks: stemming improves the score for four out of the nine comparative runs.

The Okapi combination of document stems and words, `UAmsT03WnOWS`, does not improve over document stems run. The combination of the three stemmed `Lnu.ltc` runs, run `UAmsT03WnLn3`, does improve 34.8% over the best scoring stemmed runs. The combination of the three non-stemmed language model runs, `UAmsT03WnLM3`, improves 16.9% over the best scoring base runs. Finally, the run using the matching-span weighting uses a `Lnu.ltc` full document base run with a different slope of 0.1 scoring a MRR of 0.2742. The resulting run, `UAmsT03WnMSW`, improves no less than 48.5% over the underlying base run.

Table 12: Results for home page topics.

Run identifier	MRR	Top 10	not found
UAmsT03WnOWS	0.2567	67 (44.7%)	55 (36.7%)
UAmsT03WnLM	0.2462	64 (42.7%)	60 (40.0%)
UAmsT03WnLn3	0.4105	97 (64.7%)	**26** (17.3%)
UAmsT03WnLM3	**0.4402**	**101** (67.3%)	33 (22.0%)
UAmsT03WnMSW	0.2708	73 (48.7%)	53 (35.3%)

We also break down the score over the 150 home page topics (in Table 12) and the 150 named page topics (in Table 13). Here we see a much better performance on the named page topics. This is perhaps unexpected because named page finding is conceived to be a more difficult task than home page finding. The simple explanation is that we decided not to apply special home page finding strategies. Although techniques like slash-counts or URL priors are effective for home page finding [7], they seem to hurt the named page topics considerably. Even without a particular home page bias, home pages can be retrieved with reasonable effectiveness, as is witnessed by our results for the home page topics in Table 12.

Table 13: Results for named page topics.

Run identifier	MRR	Top 10	not found
UAmsT03WnOWS	0.5098	111 (74.0%)	15 (10.0%)
UAmsT03WnLM	0.4721	106 (70.7%)	21 (14.0%)
UAmsT03WnLn3	0.5859	**121** (80.7%)	12 (8.0%)
UAmsT03WnLM3	**0.5969**	113 (75.3%)	13 (8.7%)
UAmsT03WnMSW	0.5438	116 (77.3%)	**11** (7.3%)

Conclusions

Our general conclusion on the home/named page finding task is that the compact document representations such as the title and anchor text indexes can outperform the massive full document index for known-item searching.

4.2 Topic Distillation Task

Runs

We submitted the following five official runs for the topic distillation task:

`UAmsT03WtOk3` Weighted fusion of Okapi runs on the three stemmed indexes: 0.7 full documents, 0.2 titles; and 0.1 anchor texts.

`UAmsT03WtLM3` Weigthed fusion of language model runs on the three stemmed indexes: 0.7 full documents ($\lambda = 0.35$), 0.2 titles ($\lambda = 0.7$), and 0.1 anchor texts ($\lambda = 0.7$). We combine the probabilities without normalization.

`UAmsT03WtOkI` Weighted fusion of 0.9 Okapi run on the stemmed full document index with 0.1 of a link topology measure. We applied the realized indegree on the top 10 documents [12]. This is similar to HITS [6], where we consider the fraction of inlinks that is in the local set—roughly a `tf.idf` measure for link topology.

UAmsT03WtLMI Weighted fusion of 0.9 language model run ($\lambda = 0.35$) on the stemmed full document index with 0.1 of the realized indegree of the top 10 documents.

UAmsT03WtOkC Weighted fusion of 0.8 Okapi run on the stemmed full document indexe with 0.2 of a URL-based reranking. The reranking was done by clustering the found pages by their base URLs, and to only return the page with the lowest slash-count per cluster.

Results

The results of the official runs for the topic distillation task are shown in Table 14 (best scores in boldface). The second column shows the mean average precision,

Table 14: Results for topic distillation.

Run identifier	MAP	Prec. at 10, 20, 30		
UAmsT03WtOk3	**0.1344**	**0.0980**	**0.0810**	**0.0787**
UAmsT03WtLM3	0.1019	0.0840	0.0630	0.0533
UAmsT03WtOkI	0.0862	0.0760	0.0660	0.0567
UAmsT03WtLMI	0.0412	0.0280	0.0260	0.0267
UAmsT03WtOkC	0.1127	0.0860	0.0650	0.0540

the third to fifth columns show the precision at 10, 20, and 30 documents, respectively. The best score is obtained by `UAmsT03WtOk3`, the Okapi run on the three stemmed indexes. The second best score is obtained by `UAmsT03WtOkC`, a URL-based clustering of the Okapi full documents run. Before discussing the results of our experiments, we first have to evaluate the results of the runs used to create our official runs.

Table 15 shows the results of the base runs used in combination for our official runs. All these runs use the Snowball stemming algorithm [16]. We see a remarkable divergence between the scoring for Okapi and the language model. The Okapi model performs comparable on all the three indexes, documents, titles, and anchors. The language model performs poorly on the document and title indexes, but excels for the anchor text index. The combination of the three Okapi runs, `UAmsT03WtOk3`, improves significantly over the best underlying run (MAP +38.4%, Precision at 10 +25.6%). The combination of language model runs, `UAmsT03WtLM3`, uses far from optimal relative weights and, as a result, does not improve over the anchor text run. The runs using the hyperlink graph topology do not result in significant improvement. The Okapi run `UAmsT03WtOkI` slightly improves its precision at 10 over the document run; whereas the language model run `UAmsT03WtLMI` slightly decreases its precision at 10 over the document run. Finally, the Okapi run clustering per base URL, `UAmsT03WtOkC`, does improve over the Okapi document run (MAP +25.1%, Precision at 10 +16.2%).

Table 15: Results for topic distillation stemmed base runs.

Run type	MAP	Prec. at 10, 20, 30		
Doc. Okapi	0.0901	0.0740	0.0580	**0.0527**
Title Okapi	0.0870	0.0780	**0.0590**	0.0453
Anchor Okapi	0.0971	0.0780	0.0560	0.0493
Doc. LM (0.35)	0.0386	0.0300	0.0320	0.0293
Title LM (0.70)	0.0434	0.0480	0.0360	0.0293
Anchor LM (0.70)	**0.1068**	**0.0860**	0.0560	0.0473

Conclusions

Our general conclusion for the topic distillation task is that methods using link topology do not help to improve retrieval effectiveness, whereas the document representation using anchor texts is particularly effective for the topic distillation task.

5 Conclusions

In this paper we have described our participation in the TREC 2003 Question Answering, Robust, and Web Tracks.

This year, our work for the question answering track was largely motivated by our move to a new, multi-stream architecture. Although a further and more detailed analysis of the performance of the system remains to be done, our preliminary results show that different approaches to the QA process do produce answers to different question types. Our combined use of external resources and handcrafted feature sets proved to be a successful approach for answering definition questions.

For the robust track, we experimented with the impact of stemming and feedback on the worst scoring topics. Our results suggest that blind feedback helps overall performance but does not increase the effectiveness on the lowest scoring topics. Our results also suggest that applying a stemming algorithm does benefit both the overall performance, as well as the performance of the worst scoring topics. This result sheds some new light on the role of morphological normalization in information retrieval.

For the web track, we saw very similar results for both the home/named page finding task and the topic distillation task. Using the hyperlinks in the collection for creating an anchor text index turns out to be very effective. Also, the use of HTML-structure in the documents to elicit their titles turns out to be effective. Combining these alternative document representations with a standard document index led to our best scores for both tasks.

A further general observation is the effectiveness of compact document representations, such as indexing only document titles, or only anchor texts pointing towards documents. These compact document representations result in performance that meets or exceeds the performance of a massive full document text index. This result suggests that it is feasible to create effective retrieval indexes for even larger web collections, provided that the appropriate document representation is chosen.

Acknowledgments

Thank you to Börkur Sigurbjörnsson for useful suggestions and discussion. We thank Raffaella Bernardi for help on the question classifier and table extraction components, and Karin Müller and Detlef Prescher for help with our named entity recognizer. Many thanks to Henry Chinaski for sanity checks and inspiration.

Valentin Jijkoun, Gilad Mishne, and Stefan Schlobach were supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Jaap Kamps was supported by NWO under project numbers 400-20-036 and 612.066.302. Christof Monz was supported by NWO under project numbers 612-13-001 and 220-80-001. Maarten de Rijke was supported by NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, and 612.066.302.

References

[1] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In D. Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. National Institute for Standards and Technology. NIST Special Publication 500-236, 1996.

[2] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel, editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–365. ACM Press, New York NY, USA, 2001.

[3] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In P. Bennett, S. Dumais, and E. Horvitz, editors, *Proceedings of SIGIR'02*, pages 291–298, 2002.

[4] E. Fox and J. Shaw. Combination of multiple searches. In D. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–252. National Institute for Standards and Technology. NIST Special Publication 500-215, 1994.

[5] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Center for Telematics and Information Technology, University of Twente, 2001.

[6] J. M. Kleinberg. Authoritative structures in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.

[7] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, editors, *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 27–34. ACM Press, New York NY, USA, 2002.

[8] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? exploiting web redundancy for

answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 425–432, 2002.

[9] C. Monz. *From Document Retrieval to Question Answering*. PhD thesis, University of Amsterdam, 2003.

[10] C. Monz and M. de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems, CLEF 2001*, volume 2406 of *Lecture Notes in Computer Science*, pages 262–277. Springer, 2002.

[11] C. Monz and M. de Rijke. Tequesta: The university of Amsterdam's textual question answering system. In E. M. Voorhees and D. K. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2001)*, pages 519–528. National Institute for Standards and Technology. NIST Special Publication 500-250, 2002.

[12] C. Monz, J. Kamps, and M. de Rijke. The University of Amsterdam at TREC 2002. In E. M. Voorhees and L. P. Buckland, editors, *The Eleventh Text REtrieval Conference (TREC 2002)*, pages 603–614. National Institute for Standards and Technology. NIST Special Publication 500-251, 2003.

[13] J. Prager, J. Chu-Carroll, and K. Czuba. Use of wordnet hypernyms for answering what-is questions. In E. M. Voorhees and D. K. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2001)*, pages 250–257. National Institute for Standards and Technology. NIST Special Publication 500-250, 2002.

[14] S. Robertson, S. Walker, and M. Beaulieu. Experimentation as a way of life: Okapi at TREC. *Information Processing & Management*, 36:95–108, 2000.

[15] J. Rocchio, Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.

[16] Snowball. Stemming algorithms for use in information retrieval, 2003. `http://www.snowball.tartarus.org/`.

[17] C. C. Vogt and G. W. Cottrell. Predicting the performance of linearly combined IR systems. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 190–196. ACM Press, New York NY, USA, 1998.