Question Answering with QED at TREC-2005

Kisuh Ahn, Johan Bos, James R. Curran*
Dave Kor, Malvina Nissim & Bonnie Webber

School of Informatics, University of Edinburgh
*School of Information Technologies, University of Sydney
trec-qa@inf.ed.ac.uk

Abstract

This report describes the system developed by the University of Edinburgh and the University of Sydney for the TREC-2005 question answering evaluation exercise. The backbone of our question-answering platform is QED, a linguistically-principled QA system. We experimented with external sources of knowledge, such as Google and Wikipedia, to enhance the performance of QED, especially for reranking and off-line processing of the corpus. For factoid and list questions we performed significantly above the median accuracy score of all participating systems at TREC 2005.

1 Introduction

The QA evaluation exercise at TREC consists in automatically finding answers for a collection of questions arranged by different topics, or, to use the TREC terminology, *targets*. Questions can be either *factoids*, asking for a unique short answer, or *list*-questions, asking for a set of answers. Each series of questions ends with an *other*-question, which is a request of providing all relevant information about the target which was not already asked in the other questions. Here is an example:

TARGET: Russian Submarine Kursk sinks

66.1 (factoid) When did the submarine sink?
66.2 (factoid) Who was the on-board commander of the submarine?
66.3 (factoid) The submarine was part of which Russian fleet?
66.4 (factoid) How many crewmen were lost in the disaster?
66.5 (list) Which countries expressed regret about the loss?
66.6 (factoid) In what sea did the submarine sink?
66.7 (list) Which U.S. submarines were reportedly in the area?
66.8 other

The answers must be found in the Aquaint corpus, a collection of over a million newspaper articles from three American newspapers dating from 1998–2000. A response is evaluated as correct if it exactly answers the question (in an exhaustive but not overinformative way) and if it is accompanied by an appropriate document from the Aquaint corpus supporting the answer.

In this paper we describe the TREC-2005 entry of the Universities of Edinburgh and Sydney for the question-answering evaluation exercise. This is the third time we participate in the TREC-QA campaign. Compared to the previous years (Leidner et al., 2003; Ahn et al., 2004), the performance of our system improved considerably.

The most interesting aspect of the QED system is that it is linguistically principled, combining symbolic with statistical approaches. QED uses one and the same theory and implementation for the analysis of both the question and the documents, employing detailed semantic representations and inference techniques to match possible answer-sentences with the question.

With respect to its architecture, QED is a fairly traditional QA system, which is composed of a standard sequence of modules: Question Analysis, Document Retrieval, Passage Analysis, Answer Extraction, Answer Reranking. Section 2 describes QED in more detail.

Although the main focus of research is on factoid questions, this year we also put additional effort into developing techniques dedicated to process list questions (resulting in a subcomponent called LiQED (Kor, 2005), questions asking for titles of published works and other-questions (see Section 3).

We also experimented by integrating an additional processing pipeline that views question answering in a radically different way, namely TOQA, a QA system developed by Kisuh Ahn (see Section 4).

Finally, in Section 5, we present the results obtained at TREC 2005 and an evaluation of the individual components in QED.

2 The QED System

2.1 Question Analysis

Like most traditional QA systems, the first stage of processing in QED is analysing the question. Each question is interpreted in the context of the TREC target, and processed along the following steps: tokenisation; syntactic analysis; semantic interpretation; and question typing.

Tokenisation is performed with NLProcessor[1]. Syntactic analysis is based on Combinatory Categorial Grammar (CCG), using a robust wide-coverage parser (Clark and Curran, 2004). The output of syntactic analysis is a CCG-derivation, as shown in Figure 1 (Underneath each word is the CCG category used in its analysis. Horizontal lines show the result of category combination, each labelled underneath with the resulting category. At the right end of the horizontal lines a symbol denotes the combinatorial rules used: $>$ for forward application, $<$ for backward application, $> B$ for forward composition, and $> P$ for the punctuation rule.).

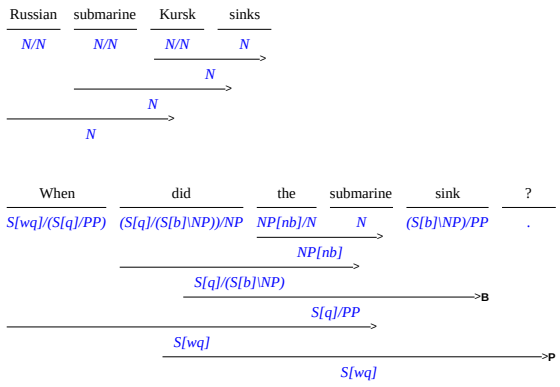

Figure 1: CCG derivation for question 66.1

The example in Figure 1 shows the target incorrectly analysed as a noun (N), due to "sinks" being incorrectly assigned the category N, and the question correctly analysed as a wh-question (category S[wq]).

Semantic interpretation is based on Discourse Representation Theory, DRT (Kamp and Reyle, 1993), and semantic representations are built on the basis of the CCG-derivation output by the parser (Bos et al., 2004; Bos, 2005). DRSs are defined as ordered pairs of a set of discourse referents and a set of DRS-conditions. See Figure 2 for an example DRS for question 66.1 and its target.

[1] A product from Infogistics, see `http://www.infogistics.com/textanalysis.html`

x0 x1 x2 x3 e4
nn(x1,x0)
nn(x2,x0)
russian(x0)
submarine(x1)
kursk(x2)
sink(x0)
answer(x3,date,date)
temp_rel(e4,x3)
arg1(e4,x1)
sink(e4)

Figure 2: Discourse Representation Structure for question 66.1

The top of Figure 2 shows the set of discourse referents (four of semantic type "individual" and one of semantic type "event"), while the bottom shows the set of DRS-conditions. The first six conditions come from the target, where "sink" has been incorrectly analysed as a noun, yielding "sink(x0)", and the other four from the question, where "sink" has been correctly analysed as an event "sink(x4)" whose subject is correctly resolved to the same russian submarine as in the target "arg1(e4,x1)".

The question type is determined on the basis of the semantic content of the question. In QED we distinguish a hierarchy of 12 main question types: `reason`, `manner`, `definition`, `color`, `count`, `measure`, `date`, `location`, `name`, `abbreviation`, `publication`, and `general`. Each of these main types is further divided into various subtypes. Questions introduce a special DRS-condition of the form `answer(x,T,S)` for a question type `T` and subtype `S`. We call this the *answer slot*; answer slots play an important role in answer extraction.

2.2 Document Prefetching and Passage Selection

The Aquaint document collection, which forms the basis for TREC-2005, was pre-processed and tokenised offline with NLProcessor. The result was indexed with the Lemur search engine (Ogilvie and Callan, 2002). Using ranked document retrieval, we obtained the best 1,000 documents from Lemur, supplying the target phrase as a query. Hence for the entire series of questions related to a single target, the same set of documents was used.

Since our approach involves full parsing to obtain detailed semantic representations in later stages, we need to reduce the amount of text to be processed to a fraction of each document. Passages spanning two sentences were selected from the returned documents based on the presence of at least one of the words from the target phrase, which were weighted (based on the number of

target words found) to get an overall ranking of the passages. Again, the passages are the same for all the questions associated with the same target.

Document retrieval as such is relatively unimportant for the success of QED. At this stage of processing we aim for high recall and ignore precision, by selecting a high number of documents and passages, and narrowing down this pool of potential answers as late as possible in the processing pipeline.

2.3 Passage Analysis

As with the questions, we used the CCG-parser to parse the passages and then build DRSs on the basis of the derivations output by the parser. The CCG-parser also performs POS-tagging (Curran and Clark, 2003a) and named entity recognition (Curran and Clark, 2003b), identifying named entities from the standard MUC-7 data set (locations, organisations, persons, dates, times and monetary amounts).

Each passage is translated into a single DRS; hence a DRS can span several sentences. A set of DRS normalisation rules are applied in a post-processing step, thereby dealing with active-passive alternations, inferred semantic information, normalisation of date expressions, and the disambiguation of noun-noun compounds. The resulting DRS is enriched with information about the original surface word-forms and POS-tags.

2.4 Answer Extraction

The answer extraction component takes as input a DRS for the question, and the set of DRSs for selected passages. It extracts answer candidates from the passages by matching the question-DRS and a passage-DRS, using a relaxed unification method and a scoring mechanism indicating how well the DRSs match each other.

Matching takes advantage of Prolog unification, using Prolog variables for all discourse referents in the question-DRSs, and Prolog atoms in passage-DRSs. It attempts to unify all terms of the question-DRSs with terms in a passage-DRS, using an A^* search algorithm. Each potential answer is associated with a score. High scores are obtained for perfect matches (i.e., standard unification) between terms of the question and passage, low scores for less perfect matches (i.e., obtained by "relaxed" unification). Less perfect matches are granted for different semantic types, predicates with different argument order, or terms with symbols that are semantically related (hypernymy) according to WordNet (Fellbaum, 1998).

After a successful match, the answer slot is identified with a particular discourse referent in the passage-DRS. This is made possible by the fact that DRS-conditions and discourse referents are co-indexed with the surface word-forms of the source passage text. This information is used to generate an answer string, simply by collecting the words that belong to DRS-conditions with discourse referents denoting the answer. Finally, all answer candidates are output in an ordered list. Duplicate answers are eliminated, but answer frequency information (F) is retained and added to each answer in this final list. Figure 3 shows an example output file.

2.5 Reranking

The basic ranking algorithm for potential answers is fairly straightforward. Only two features are used: the matching score, and the frequency of similar answers. If there are different answers with the same matching score, frequency will be used to order them.

This simple method yields a reasonable ranking. However, in many cases the correct answer is ranked high but not highest, as for 99.3:

TARGET: Woody Guthrie

99.3 Where was Guthrie born?'

Initially, QED produced the following ranked answer candidates for 99.3 (answer 2 and 4 are correct):

1. Britain
2. Okemah, Okla.
3. Newport
4. Oklahoma
5. New York

To deal with this problem, we experimented with external additional knowledge, namely the World Wide Web accessed via the Google API, to generate a possibly more accurate ranking. This technique is also known as "answer validation", and can be seen as a tie-breaker between the top-N answers.

In more detail, this method works as follows. For each of the N-best answer candidates, we take the semantic representation of the question and fill the answer slot with the answer candidate. From this we generate a set of declarative sentences (covering all morphological variations). The generated sentences are submitted as strict (within quotes) queries for Google. Any information from the target which is not included in the generated sentence (for this example "Woody") is added as a query term in order to constrain the search space. The queries and number of hits returned for each of the queries (in brackets) are shown below.

1. Woody "Guthrie born in Britain" (0)
   Woody "Guthrie are OR is OR was OR were born in Britain" (0)
2. Woody "Guthrie born in Okemah, Okla." (1)
   Woody "Guthrie are OR is OR was OR were born in Okemah, Okla." (10)
3. Woody "Guthrie born in Newport" (0)
   Woody "Guthrie are OR is OR was OR were born in Newport" (0)

Document	Score	F	Answer
XIE20000822.0059	0.755556	2	Russian President Vladimir Putin on Tuesday declared Wednesday a day of mourning for the crew of the sunken nuclear submarine Kursk. " The Russian nuclear submarine Kursk sank in the Barents Sea on **August 12, 2000**, its crew perished.
NYT20000814.0435	0.733333	4	In **1989** a Soviet nuclear submarine sank off north Norway, killing 42 of the 69-man crew. The Komsomolets submarine, the prototype of the Mike-class sub, now lies rusting, along with its nuclear torpedoes, on the sea bed of the Barents.
NYT20000828.0399	0.688889	7	Given that secrecy, and the likelihood that the Russians will not fully share what they learn even if they recover the wreckage, it will be difficult to learn with any certainty what happened to the Kursk. In **1968**, an American submarine, the Scorpion, sank in the Atlantic near the Azores.

Figure 3: QED extracted answers for question 66.1

4. Woody "Guthrie born in Oklahoma" (7)
   Woody "Guthrie are OR is OR was OR were born in Oklahoma" (42)

5. Woody "Guthrie born in New York" (0)
   Woody "Guthrie are OR is OR was OR were born in New York" (2)

The returned Google-counts are used as the deciding factors to rerank the N-best answers. Note that we generate several queries for each answer candidate and we sum the returned hits. In this example, the answers would be reranked as follows:

1. Oklahoma ($7+42$)
2. Okemah, Okla. ($1+10$)
3. New York ($0+2$)
4. Britain ($0+0$)
5. Newport ($0+0$)

For this example, reranking correctly promoted "Oklahama" to best answer.

3 Dedicated Processing

3.1 Processing Publication-Questions

For questions of type `publication` (titles of creative works, such as books, records, plays, etc.), one of the way answer candidates are recognised by the answer extraction module is by quotation marks. However not all titles in the corpus are decorated with quotes.

To overcome this problem we adopted a strategy that exploits information encoded in `http://www.amazon.com`, a commercial website equipped with a large database containing information on several types of publications, especially useful for printed material and music. This database is accessible via an API. The basic strategy is to collect a list of titles relevant to the question target and to mark them with quotes in the corpus. QED also assigns a subtype to questions of type publication, e.g. book, so we used predetermined rules to map such subtypes to Amazon categories. The selected category (e.g. music) together with the question target (e.g. Nirvana), was then searched over the whole of Amazon's database. All resulting hits were collected and matched back into the portions of the Aquaint corpus retrieved for a given question. Quotes were then added around each successful match in the text.

For all questions that underwent such procedure, we parsed the quotes-enriched passages a second time, since the parser has further clues to determine whether a given phrase is an instance of a publication. After this, the standard procedure followed.

3.2 Processing List Questions

This year, besides using QED to generate answers for list questions, we also introduced a new approach to answering list questions. List questions are interesting because they offer an opportunity for a Question Answering system to directly examine the relationship between a question and its answers. The new approach takes a question and an existing set of answers, generated by QED in our case, and uses these answers as examples to identify more answers of similar nature. In essence, this is a bootstrapping method for expanding our initial set of answers. This approach is built into our new list question anwering module, LiQED (Kor, 2005).

The general approach takes a question and an initial set of potential answers, identifies some context that is shared between the question and two or more answers, then extrapolates from this shared context to expand on the initial set of answers with new and distinct answers. This shared context can be expressed in several forms, such as text patterns, logical forms or branches in parse trees. The initial set of answers can contain a mix of both correct and wrong answers. Typically, only shared contexts from correct answers are identified. The reason, using parlance from physics, is that typically wrong answers create "interference" while correct answers mutually "reinforce" each other. In general, this approach requires a minimum of two correct answers in the answer set before any shared context can be identified. Although we did not implement this, the identified shared context can conceivably be used in reverse to verify correct answers in the initial answer set.

Specifically for LiQED, the shared context we use comprises surface text patterns. In essence the new module takes answers generated by QED as examples, automatically identifies common shared surface text patterns and finally uses these patterns to look for more answers. This is achieved for each question in two phases: a pattern generation phase, and an answer extraction phase.

In the pattern generation phase, we identify sentences in the Aquaint corpus that contain the question target and one answer. We noticed that a significant number of sentences contain some form of long distance dependency separating the question target and answer. In order to capture these dependencies, our text patterns are able to match either one contiguous fragment or two separate fragments within a single sentence. This is achieved by only considering relevant terms that fall within a three-chunk window around either a question target or an answer. The function $R(x)$ measures the relevance of each term by determining if the term frequently occurs near the question target and/or answer:

$$R(x) = \frac{1}{N}\sum_{i=1}^{N} w_q \left(1 - \frac{D(x_i, T_q^i)}{S(T_q^i)}\right) + w_a \left(1 - \frac{D(x_i, T_a^i)}{S(T_a^i)}\right)$$

In the formula above, N is the number of occurrences of term x in the Aquaint corpus. T_q and T_a are respectively the set of terms found in the question target and the potential answer. $D(x,T)$ measures the distance between term x and the nearest term in T while $S(T)$ is the longest span within a sentence between two terms in T or the start or end of the sentence. w_q and w_a are weights that give importance to either the question target or answer. Using a threshold, we retain the most relevant terms in each sentence. This leaves N sentence fragments that are compared in a pairwise manner using the Smith-Waterman (Smith and Waterman, 1981) and Gotoh (Gotoh, 1982) word alignment algorithm, to identify common patterns. We filter away irrelevant patterns, leaving only patterns that contain both the question target and answer as our set of automatically generated patterns.

Figure 4 shows some of the patterns generated for the question "What movies was Bing Crosby in?". Square brackets in the pattern indicate there is some long distance dependency gap between question target and answer, asterisk characters indicate a small gap of 1–3 terms. Using these generated surface text patterns, we search the Aquaint corpus again for matching sentences. The text patterns narrow down to a three-chunk window in the sentence that potentially contains an answer.

To extract the answers, we use a simplified form of ensemble learning by combining evidence from our text patterns, a named entity tagger and other external information sources. For example, with questions expecting organization names as answers, we can extract answers by choosing all distinct phrases that fall within the three-chunk window and are labeled as organizations by a named entity tagger. External sources of information include book and film titles extracted from Amazon as well as hyponyms extracted using Google queries (Hearst, 1992).

3.3 Processing Other-Questions

Other-questions were and still are the poor relation of QED. Not much effort is put into processing these: answers to other-questions are sentences identified in the corpus that contain the target as subject or object. No sophisticated techniques were used to filter out duplicated or repeated answers. All nuggets that the system had already provided as answers to other questions for the same target were considered as redundant and removed. We considered this setting as a baseline for answering other-questions.

To improve on the baseline we collected a set of "important" words, and used this set to select interesting answers from the set of answers generated by the baseline. The important words were extracted by the correctly judged answers generated for the other-questions of TREC-2004, that is cue words that might make a sentence worth considering. Example of such "important" words are 'first', 'best', 'award', 'invented', and similar. This list was collected by semi-automatically producing a word-frequency list and removing stop words.

As an additional improvement, we exploited the online encyclopedia, Wikipedia (`http://en.wikipedia.org/wiki/Main_Page`). For a given question, the system sent the TREC target as a query to the Wikipedia's built-in search engine. If there was a direct match between the target and the Wikipedia topic retrieved by the search engine, then the system fetched the found Wikipedia article. Otherwise, the article of the closest Wikipedia topic, as judged by the built-in Wikipedia search engine was used. Finally, if there was no match at all, the system did not attempt to answer the question.

The Wikipedia article is an XML file whose meta-data includes information about the categories the article belongs to. Wikipedia categories are vast and somewhat unconstrained, since the individual who authors the article can specify any number of pre-existing or even new categories. This makes such information a possibly very rich and valuable description of the target entity.

Finally, in order to extract an appropriate supporting document for the information provided, we used boolean search with the category descriptions as the queries to the orginial Aquaint document collection. This was done by first finding interesting facts about a target in Wikipedia, and then aiming to localise these nuggets of information in the TREC corpus.

```
<BING_CROSBY> 's '' <ANSWER> ''
[ '' <ANSWER> to ] [ bob hope and <BING_CROSBY> ]
bob hope and <BING_CROSBY> * '' <ANSWER>
bob hope and <BING_CROSBY> * <ANSWER>
the '' <ANSWER> * '' <BING_CROSBY>
```

Figure 4: *Answer-finding text patterns for "What movies was Bing Crosby in?"*

4 Topic-based QA

TOQA (Topic Oriented Question Answering) is a self-contained question answering system and was used in TREC 2005 to complement QED. The philosophy of TOQA is to rely heavily on off-line processing of the Aquaint corpus, to yield speedy answer retrieval in real-time. For the present exercise, TOQA was ran in an independent pipeline to produce answer candidates that were combined with QED answer candidates to form the best answer-candidate pool. TOQA is still a system in development, and only factoid questions pertaining to named-entity answers were processed with TOQA.

4.1 Off-Line Processing

TOQA does most of its work off-line. This preprocessing stage is aimed at identifying all types of expressions in the Aquaint corpus that could serve as potential answers in a question-answering exercise. We call these *topics*. TOQA extracts topics together with their contexts creating a new document for each distinct topic. The document for an identified topic is the set of all the sentences found in the corpus that say something about this topic. Each of these documents is then indexed to enable efficient retrieval. To constrain the search space, separate indices are created depending on the different topic types. For the present version of the system, only expressions that have been identified as relating to some named entity type, such as *person*, *location* and *organization*, have been targeted as topics.

4.2 Answer Retrieval

Answering questions in TOQA boils down to retrieving the most relevant topic document created in the aforementioned process. The title of the retrieved document (the topic itself), serves as the succinct answer to the question. Note that the answer can be retrieved using an ordinary information retrieval method. First, the index to be used is decided by the answer-type of the question identified. Then, the base query is formulated by concatenating the target words to the question, from which the stop-words had been removed. Finally, this query is fed into the off-the-shelf IR component Lemur Version 2.2 (Ogilvie and Callan, 2002) to retrieve the relevant answer candidates.

4.3 Document Retrieval

As explained, TOQA does not rely on the original passage or document for the answer extraction, but rather on a newly created set of documents that contains information originally spread across several documents of the Aquaint collection. For this reason, the document supporting the answer that TOQA provides can only be found via a post-processing phase. This consists in forming a new query by combining the base query for which the answer was retrieved with the retrieved answer candidate. This new query is then used with Lemur to retrieve the most relevant Aquaint document.

5 Evaluation

5.1 Experimental Setup

Three runs were submitted, all with different parameters with respect to the treatment of factoids, list, and other-questions. Let's first consider the factoid questions:

FACTOID-questions (run descriptions)	
Run A (Edin2005A)	QED
Run B (Edin2005B)	QED + Google reranking
Run C (Edin2005C)	QED + Google reranking + TOQA + paraphrases

We expected Run B to outperform Run A by gaining around 10% in accuracy, based on results on the training data of TREC 2004. Run C was partly an experimental run using newly developed techniques that we hadn't had tested thoroughly, such as the use of paraphrases in background knowledge, and TOQA.

For list questions, our first run comprises the top twelve answers generated by QED and serves as a baseline for comparison. Our second run is a combination of the top ten answers from QED and LiQED while our final run takes the top seven answers from QED, LiQED and TOQA. The number of answers were mainly chosen to balance precision and recall of the combined answer set.

LIST questions (run descriptions)	
Run A (Edin2005A)	QED's top twelve
Run B (Edin2005B)	QED's top ten + LiQED's top ten
Run C (Edin2005C)	QED's top seven + LiQED's top seven + TOQA's top seven

For list questions we expected Run B to have better results than Run A (a prediction based on training data of TREC 2004). Run C was again an experimental run.

OTHER-questions (run descriptions)	
Run A (Edin2005A)	QED
Run B (Edin2005B)	QED + important-word-filter
Run C (Edin2005C)	QED + important-word-filter + WIKIPEDIA

Finally, for the other-questions, we expected Run B to perform considerably better than Run A, and Run C better than Run B.

5.2 Results at TREC 2005

Factoid Questions Factoid questions formed the majority of the questions at the TREC 2005 QA evaluation exercise. Our results over 362 factoid questions are listed in the table below, where U is the number of unsupported (correct but without a supporting document), X the number of inexact and R the number of correct answers. The last two columns show the accuracy (calculated on the basis of R) and lenient accuracy (calculated on the basis of U+X+R).

FACTOID-questions (results)					
Run	U	X	R	Accuracy	Lenient Acc.
A	5	25	76	0.210	0.293
B	9	27	78	0.215	0.315
C	12	26	77	0.213	0.318

The reranking procedure using Google improved the results but only slightly. On the positive side, it transformed 10 wrong or inexact answers in Run A into correct answers, but it also transformed 8 correct answers into wrong, inexact, or unsupported answers. This seems a promising way of doing reranking, but there is clearly space for improvement.

Overall, our results were roughly as we had anticipated and we were satisfied with the performance of QED: the accuracy scores were significantly higher than the median accuracy score of all participating systems (0.152). However, we were slightly disappointed with the high number of inexact answers. Closer inspection of these cases revealed that some of the QED answers would have been judged as exact answers in previous TREC campaigns. A case in point are compound expressions such as "Atlanta-based" (88.1), "Dominican-born" (100.1), "Milwaukee-based" (119.2) and "Vienna-based" (128.4), where the expected correct answer was Atlanta, Dominican Republic, etc. Also for abbreviation-questions our system generated a relatively high number of inexact answers. These shortcomings relative to inexact answers seem easy to overcome.

Just for the record, we mention the results for dealing with factoid questions that have no known answer (in the corpus). There were 17 of those. A correct response to these questions would be NIL. For the three runs, our system generated a total of 43, 44 and 40 NIL answers, with only 3, 3 and 2 being correct.

List Questions There were 93 list questions in total. QED achieved an average F-score of 0.081 for Run A, of 0.075 for Run B, and of 0.074 for Run C. Although we were hoping to see Run B perform better than Run A, this was not the case. A post-submission analysis revealed that the majority of answers returned by LiQED were evaluated as unsupported or inexact. Still all obtained F-scores were higher than the median average F-score of all runs (0.053).

Other-Questions Since we didn't do anything sophisticated for dealing with other-questions, the results were rather meager compared to what our system achieves on factoid and list questions: Run A yielded an F-score of 0.070, Run B of 0.095, and Run C of 0.102. Our best score (Run C) is still below the median average F-score of all participating systems (0.156).

5.3 Evaluating Question Analysis

Despite the detailed linguistic analysis, our approach to question analysis is fairly robust and gives good results. Of the 455 questions at TREC-2005 (363 factoids, 93 list), we were able to analyse 426 (93.6%) "approximately" correctly. There were 9 questions for which the parser failed to find a derivation, and 20 questions that got a wrong parse. The example in Figure 2 illustrates an approximately correct output: it is an almost perfect analysis, with the definite description "the submarine" in the question resolved to the antecedent "Russian submarine Kursk" in the target. The only mistakes are in the target, where "sinks" is analysed as a noun and "Russian submarine Kursk" is analysed as simple noun-noun modification. Errors like these hardly affect the performance of the overall system.

5.4 Component Evaluation

For future work it is valuable to be able to identify which module in the system has initial responsibility for losing an answer. To do this thoroughly for each question is however rather time-consuming. In the context of this year's TREC results, we performed an analysis into how QED behaved with respect to one particular question-type, namely date-questions.

There where 67 date-questions in the TREC-2005 test set of questions (15% of the 455 total number of factoid and list questions). For Run A, we counted the number of correct answers available after each stage of processing (this was not just a syntactic check, also the exactness and relevant document were taken into account to mark an answer as correct). The table below shows the results,

together with the loss of correct answers caused by each component.

Component	Correct	Acc.	Loss
Question Analysis	64	0.96	4%
Document Retrieval	57	0.85	11%
Passage Selection	51	0.76	9%
Passage Analysis	46	0.69	7%
Extraction (top 10)	43	0.64	5%
Reranking	29	0.43	21%

These results suggest that substantial improvements can be made at the beginning of the processing pipeline (document retrieval plus passage selection) and at the end of it (reranking) in order to obtain the most significant improvement.

5.5 Future Work

Given the current performance of QED and our experience with the system, we can identify the following areas for future work.

- More fine-grained set of question types. The current set is not specific enough to cover all varieties of questions that occur in general QA tasks.
- Improving the Named Entity recognition module. A large proportion of the question types rely on the correct identification of named entities.
- Improving the document retrieval and passage selection modules.
- Given that in many cases the correct answer is ranked high but not highest, a more sophisticated reranking module should considerably improve the system.

References

Kisuh Ahn, Johan Bos, Stephen Clark, James R. Curran, Tiphaine Dalmas, Jochen L. Leidner, Matthew B. Smillie, and Bonnie Webber. 2004. Question answering with qed and wee at trec-2004. In E.M. Voorhees and Lori P. Buckland, editors, *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004)*, NIST Special Publication 500-261, Gaithersburg, MD.

J. Bos, S. Clark, M. Steedman, J.R. Curran, and Hockenmaier J. 2004. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*, pages 42–53.

S. Clark and J.R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain.

James R. Curran and Stephen Clark. 2003a. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 11th Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 91–98, Budapest, Hungary.

James R. Curran and Stephen Clark. 2003b. Language independent NER using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, pages 164–167, Edmonton, Canada.

Christiane Fellbaum, editor. 1998. *WordNet. An Electronic Lexical Database*. The MIT Press.

Gotoh. 1982. An improved algorithm for matching biological sequences. In *Journal of Molecular Biology*, number 162, pages 705–708.

Marti A. Hearst. 1992. Automatic aquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic. An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

K. W. Kor. 2005. Improving answer precision and recall of list questions. Master's thesis, School of Informatics, University of Edinburgh.

Jochen L. Leidner, Johan Bos, Tiphaine Dalmas, James R. Curran, Stephen Clark, Colin J. Bannard, Mark Steedman, and Bonnie Webber. 2003. The QED open-domain answer retrieval system for TREC 2003. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, NIST Special Publication 500-255, pages 595–599, Gaithersburg, MD.

P. Ogilvie and J. Callan. 2002. Experiments using the lemur toolkit. In *Proceeding of the 2001 Text Retrieval Conference (TREC 2001)*, pages 103–108.

T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.