How Frogs Built the Berlin Wall

A Detailed Error Analysis of a Question Answering System for Dutch

Valentin Jijkoun, Gilad Mishne, and Maarten de Rijke

Language & Inference Technology Group, University of Amsterdam
Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands
E-mail: {jijkoun, gilad, mdr}@science.uva.nl

Abstract. The paper describes the University of Amsterdam's participation in the Question Answering track at CLEF 2003, our system and the results produced by it. A thorough analysis of the wrong answers given by our system is provided, including a discussion of each type of error and possible strategies for handling them. We outline our current efforts for improvement of the system, and propose additional research directions and procedures to reduce errors of the presented types.

1 Introduction

In this year's CLEF evaluation exercise we participated in the *Dutch Question Answering* task, new on the CLEF agenda, building on and extending our earlier work on question answering at TREC [1]. We experimented with a multi-stream architecture for question answering, in which the different independent streams, each a complete Question Answering (QA) system in its own right, compete with each other to provide the system's final answer.

In this paper, we chose to focus on the errors made by our system. We give a detailed breakdown of the types of wrong answers we encountered and discuss their causes; additionally, we propose possible solutions for these errors, some of which are currently being implemented by us in our ongoing QA work. We hope that the paper may benefit both QA system researchers and QA engineers: we suggest areas on which to focus research, possible caveats, and directions to explore.

The paper is organized as follows. In Section 2 we give a general overview of our system and briefly present our results in CLEF 2003. Section 3 includes a classification of the errors, a diagnosis of their causes as well as a discussion of possible strategies to address them. We summarize and conclude in section 4.

2 System Description

The general architecture of a QA system, shared by many systems, can be summed up as follows. A question is first associated with a *question type*, out of

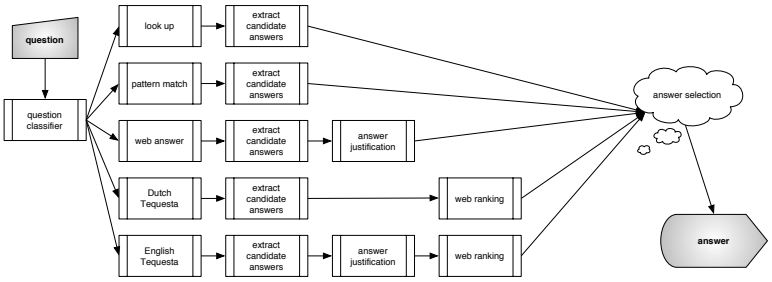

Fig. 1. The University of Amsterdam's Dutch Question Answering System.

a predefined set such as DATE-OF-BIRTH or CURRENCY. Then a query is formulated based on the question, and a retrieval engine is used to identify documents that are likely to contain the answer. Those documents are sent to an *answer extraction* module, which identifies candidate answers, ranks them, and selects the final answer. On top of this basic architecture, numerous add-ons have been devised, ranging from logic-based methods [2] to ones that rely heavily on the redundancy of information available on the World Wide Web [3].

2.1 Multi-Stream Architecture

During the design of our QA system, it became evident that there are a number of distinct approaches for the task; some are beneficial for all question types, and others only for a subset. For instance, abbreviations are often found enclosed in brackets, following the multi-word string they abbreviate, as in "*Verenigde Naties (VN)*." This suggests that for abbreviation questions the text corpus can be mined to extract multi-word strings with leading capitals followed by capitalized strings in brackets; the results can then be stored in a table to be consulted when an abbreviation (or an expansion of an abbreviation) is being asked for. Similar table-creation strategies are applicable for questions that ask for capitals, dates-of-birth, etc., whereas the approach seems less appropriate for definition questions, why-questions, or how-to questions. It was therefore decided to implement a *multi-stream* system: a system that includes a number of separate and independent subsystems, each of which is a complete standalone QA system that produces ranked answers, but not necessarily for all types of questions; the system's answer is then taken from the combined pool of candidates.

Scientifically, it is interesting to understand the performance of each stream on specific question types and in general. On the practical side, our multi-stream architecture allows us to modify and test a stream without affecting the rest of the system. A general overview of our system is given in Figure 1. The system consists of 5 separate QA streams and a final answer selection module that combines the results of all streams and produces the final answers.

Question Answering Streams. We now provide a brief description of the five streams of our QA system: Table Lookup, Pattern Match, English Tequesta, Dutch Tequesta, and Web Answer.

The *Table Lookup* stream uses specialized knowledge bases constructed by preprocessing the collection, exploiting the fact that certain information types (such as country capitals, abbreviations, and names of political leaders) tend to occur in the document collection 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 which are found there are assigned high confidence. For example, to collect abbreviation-expansion pairs we searched the document collection for strings of capitals in brackets; upon finding one, we extracted sequences of capitalized non-stopwords preceding it, and stored it in the "abbreviation knowledge base." This approach answered question such as:

Q084. Waar staat GATT voor?	
(English	What does GATT stand for?)
Knowledge Base	Abbreviations
Table Entry	GATT: Overeenkomst over Tarieven en Handel
Extracted Answer	**Overeenkomst over Tarieven en Handel**

For a detailed overview of this stream, see [4].

In the *Pattern Match* stream, zero or more regular expressions are generated for each question according to its type and structure. These patterns indicate strings which contain the answer with high probability, and are then matched against the entire document collection. Here's a brief example:

Q002. In welke stad is het Europese Parlement?	
(English	In which city is the European Parliament located?)
Generated pattern	`Europese Parlement\s+in\s+(\S+)`
Match	... voor het **Europese Parlement in Straatsburg**, dat ...
Extracted Answer	**Straatsburg**

The *English Tequesta* stream translates the Dutch questions into English using Worldlingo's translation service available at `http://www.worldlingo.com/`. The auto-translated questions are then fed to *Tequesta*, an existing QA system for English developed at the University of Amsterdam [1]. The system uses the English CLEF corpus, and is extended with an Answer Justification module to anchor the answer in the Dutch collection.

The *Dutch Tequesta* is an adaptation of English Tequesta to Dutch and used as an independent stream, provided with the original Dutch newspaper corpus. The modifications to the original system included replacing (English) language specific components by Dutch counterparts; for instance, we trained TNT [5] to provide us with Part-of-Speech tags using the *Corpus Gesproken Nederlands* [6]; a named entity tagger for Dutch was also developed.

The *Web Answer* stream looks for an answer to a question on the World Wide Web, and then attempts to find justification for this answer in the collection. First, the question is converted to a web query, by leaving only meaningful

keywords and (optionally) using lexical information from EuroWordNet. The query is sent to a web search engine (for the experiments reported here we used Google); if no relevant Web documents are found, the query is translated to English and sent again. Next, if the query yields some results, words and phrases appearing in the snippets of the top results are considered as possible answers, and ranked according to their relative frequency over all snippets. The Dutch named entity tagger and some heuristics were used to enhance the simple counts for the terms (e.g., terms that matched a TIME named entity were given a higher score if the expected answer type was a date). Finally, justifications for the answer candidates are found in the local Dutch corpus.

While each of the above streams is a "small" QA system in itself, many components are shared between the streams, including, for instance, an *Answer Justification* module that tries to ground externally found facts in the Dutch CLEF corpus, and a *Web Ranking* module that uses search engine hit counts to rank the candidate answers from our streams in a uniform way, similar to [7].

2.2 Results

Table 1 shows the evaluation results of our CLEF 2003 submissions and two post-submission runs, which resulted from very minor bug fixes. Besides the standard *Strict* and *Lenient* measures, we also evaluated our runs using a more "generous" *Lenient, Non-exact* measure that accepts non-exact answers as correct. For more details about the runs and a discussion of the results, see [8].

Table 1. Results on the CLEF 2003 test set.

	Strict		Lenient		Lenient, Non-exact	
Run	# correct	MRR	# correct	MRR	# correct	MRR
uamsex031md	78 (39%)	0.298	82 (41%)	0.317	96 (48%)	0.377
uamsex032md	82 (41%)	0.305	89 (44.5%)	0.335	102 (51%)	0.393
uamsex031md.fixed	84 (42%)	0.335	87 (43.5%)	0.352	100 (50%)	0.407
uamsex032md.fixed	88 (44%)	0.349	95 (47.5%)	0.375	107 (53.5%)	0.428

3 Error Analysis

We now turn to a discussion of the incorrect answers given by our system, give examples of each, and suggest strategies for reducing the amount of wrong answers of these types. Some of these strategies are being implemented or tested as part of our ongoing QA work; others are offered as possible research areas in the QA domain.

Out of 200 questions, we answered 88 correctly (in this context we refer to the "fixed" strict runs, i.e., inexact and unsupported answers are regarded as incorrect ones). For the remaining 112 questions, we consider 2 wrong answers

Table 2. Breakdown of errors.

Error type	Retrieval	Table	Justification	Tile	Confidence	Patterns	Lookup	Unit	Quest. classif.	Named entity	Answer selection
Absolute number of errors	2	2	5	6	8	9	12	12	20	51	97
Fraction of errors	1%	1%	2%	3%	4%	4%	5%	5%	9%	23%	43%

per question — the two top answers given by our system. Our third answer for all questions was NIL, as a simple strategy for answering the 10% questions with no known answer in the document collection. In total, we look at 224 wrong answers (also referred to as "errors"). Table 2 provides a breakdown of the errors according to type.

3.1 Answer Selection Errors

This large group of errors — 97 in total (43%) — is rather loosely-defined and revolves around the answer extraction process. Answer extraction, one of the critical stages of the Web Answer and the Tequesta streams, includes identifying possible answers from documents which were retrieved as relevant for the question. The extraction is composed of labeling terms which are likely to be answers (using named entity, part-of-speech tags and other techniques), and ranking these terms according to various measures, mostly based on proximity to query words. Some of these errors also originate from non-optimal top-ranking documents returned by our retrieval engine. Examples of such errors, where the answers given by the system are of the correct answer type and appear frequently in the relevant articles, are listed in Table 3.

Most of the errors stem from our simple ranking approach for the candidate answers, which is almost exclusively based on the proximity of the terms to the query words in the documents and their frequencies, with shallow NE tagging techniques. For example, for question 60 shown above, *Frogs* is a repeating entity in documents discussing a construction of a wall in Berlin for protecting frogs from car accidents; the construction was in debate at the time of the CLEF experiments, resulting in many web pages containing relevant terms, and referring to frogs as the reason for a construction of a Berlin wall.

This issue of *linking* entities of the right type to input questions is one of the most critical ones for QA systems. A number of partial solutions for this error class were proposed, ranging from usage of theorem provers for justification of the answer [2], rule based approaches [9] to the usage of parse tree similarity [10] and paraphrase dictionaries [11]. Recently, a noisy-channel approach was succesfully applied to address this issue [12]. Our research in this area is aimed at exploiting a range of light-weight reasoning methods, including some of the ones mentioned

Table 3. Examples of errors.

Q033. Welke Russische president gaf opdracht tot de interventie in Tsjetsjenie?
(English: Which Russian president ordered the intervention in Chechnia?)
Answer: Gratsjov
Q037. Noem een Japanse stad die door een aardbeving is getroffen.
(English: Name a Japanese city hit by an earth quake.)
Answer: Los Angeles
Answer: Tokio
Q177. Wie stelde een embargo in tegen Irak?
(English: Who imposed an embargo against Iraq?)
Answer: Saddam Hussein
Q060. Wie heeft de Berlijnse Muur gebouwd?
(English: Who built the Berlin Wall?)
Answer: frogs

above; the lack of lexical resources for Dutch and the relative poverty of existing ones is an important bottleneck in this respect.

3.2 Table Lookup Errors

As noted earlier, our knowledge bases were constructed by preprocessing the document collection, searching for facts which tend to appear in fixed, repeating patterns. For this extraction process we used a small number of hand-crafted regular expression patterns; although these patterns also pick up noise from the text, we assumed that the amount of "real" facts extracted will be much higher, so we used frequency counts of the facts to filter out the noisy ones. This approach was very successful, and indeed the number of wrong answers given by the Table stream is low. They can be grouped into two classes: *Construction Errors* and *Lookup Errors*.

Construction Errors occurred when the selected answer was one of the "noisy facts" picked up during the preprocessing; A small number of inexact answers are derived similarly. For example:

Q022. Wie is de voorzitter van de Europese Commissie?
(English: Who is the president of the European Committee?)
Answer: **Jacques Santer. Volgens**

Out of our 224 errors, only 2 (1%) were Construction Errors.

Lookup Errors occurred when the lookup process in the table produced a wrong fact. The tables we constructed are simple text files, containing one fact per line. During the lookup process we search for lines containing all keywords from the question in a certain column, and then consider the data in another column as a candidate answer. For example, for question 93, *Wie is de leider van Sinn Fein?* (English: Who is the leader of Sinn Fein?), we search the Leaders table for a line containing the words (leider, Sinn, Fein) in the "description"

column, and then consider the data in the "name" column (Gary Adams) to be an answer. If we do not find such a line, we start to omit some of the words we are looking for, based on heuristics such as capitalization and frequency of the word in the language (omitting high frequency words first). In most cases this process produces good results; however, in 12 cases (5%) incorrect answers were found using this lookup, either because the lookup words were found but had other semantics than that we intended, or because we omitted too many keywords and received irrelevant results. An example (of the first type of error):

Q118. Wie is de president van Joegoslavie?	
(English:	Who is the president of Yugoslavia?)
Answer:	**Vitaly Tsjoerkin**
Table Entry	Vitaly Tsjoerkin, president Jeltsins speciale afgezant voor het voormalige Joegoslavie

Our ongoing work for addressing these types of errors includes involving more linguistic resources in the table construction phase. For example, we now build our table using an NE-tagged version of the corpus, eliminating many noisy patterns; we also use dependency parsing for locating facts in the text, rather than just simple patterns. In future work, we intend to further enhance the construction process by using machine learning techniques to learn the patterns that store useful data in the document collection.

To address the lookup errors, we enhanced the lookup mechanism to support separate lists of stopwords and "keepwords" (words which should not be omitted during the lookup) per table. These lists help to avoid lookup errors such as supplying a former or vice president instead of a current one etc. Additionally, we plan to convert the simple text tables to real databases, allowing much more flexibility and accuracy with SQL querying (as well as increased efficiency).

3.3 Pattern Match Errors

In the Pattern Match stream, regular expression patterns were formulated using the question keywords and the question, in such a way that text which matches them contains the answer in a known position; the patterns were then matched against the document collection. As with the patterns used for constructing the tables, these patterns matched noisy elements, usually as a result of a pattern which is not strict enough; another problem was generation of non-grammatical patterns, but these did not yield any wrong answers since no text was matched at all. The following is an example of a mismatch for such a pattern:

Q021. Waar is Chiapas?	
(English:	Where is Chiapas?)
Generated Pattern	`Chiapas\s+is\s+([\^.]+)`
Match	de indiaanse boeren in Chiapas is zelfs verslechterd
Extracted Answer	**zelfs verslechterd**

In total, we encountered 9 (4%) pattern match errors; to handle them, we are experimenting with the use of part-of-speech tags in the pattern formulation

process, to generate patterns which are more strict and thus less likely to match irrelevant text. A different approach to creation of the patterns involves learning them from data collected on the Web [13].

3.4 Question Classification Errors

Our question classifier, based on a set of manually constructed pattern-based rules, achieved 86% accuracy for about 20 question types. Of the questions not classified correctly, most were not classified at all and only a small number of questions was classified incorrectly. In some cases, incorrect or no classification still produces reasonable answers (using the Web, for example); for other cases, a wrong question type implies various other failures along the system pipeline that result in wrong answers. In total, about 20 (9%) of our errors are directly attributed to mis-classifications, but deeper analysis may reveal that other errors are also derived from an incorrect question type. For example, question *138. Onder welke naam is het EFA-project ook wel bekend?* (English: The EFA project is also known under which name?) was not classified at all; had it been classified as EXPAND-ABBREVIATION or even ALSO-KNOWN-AS, it would possibly contain much better candidate answers than the ones which were actually selected.

Since our participation at CLEF, we have improved our question classifier to use part-of-speech tags and WordNet for for classifying questions not classified by the rule-based approach. We are currently reformulating the classifier, moving from the rule-based approach to a machine learning approach, using features such as ngrams of words from the question, subtrees of the question parse tree, and part-of-speech tags, in a manner similar to [14].

3.5 Justification Errors

Two of our streams obtained candidate answers from external resources rather than the Dutch document collection: the Web Answer stream used (Dutch and English) documents on the Web, and the English Tequesta stream used the English CLEF document collection, which contained English newspaper articles from the same dates as the Dutch one. Once an answer was found in one of these resources, justification of the answer — a document from the Dutch collection supporting the answer — was needed for a complete answer. For this justification process (sometimes referred to as "answer projection" [3]) we used an IR system to select the top ranking document from the Dutch collection, where the query was composed of keywords from the question and the answer, similarly to [15]. This approach sometimes failed for various reasons: English-Dutch translation problems and spelling variations, different formulations of the answer in the external resource and the document collection (synonymous words) and so on. In total, justification errors account for 5 (2%) of our errors.

To address this problem, we experimented with different IR models for retrieval of the answer justification; in the future we intend to also incorporate synonyms from WordNet for enriching our justification queries. Other answer

justification methodologies, that we have not experimented with, include sliding windows on retrieved passages techniques [16].

3.6 Named Entity Errors

For our named entity classifier, we used TnT [5] trained on the *Corpus Gesproken Nederlands* [6], and some hand-made rules for fine-tuning. Although generally this approach provided good results, 51 (22%) of the wrong answers are attributed to incorrect NE classifications. For example, *Sensibiliseringscampagne* was classified as a location and given as an answer for *185. In welk land ligt het gebied van de Grote Meren?* (English: In which country is the area of the Great Lakes located?). The Web Answer stream had many named entity errors, mainly because web snippets tend to be ungrammatical phrases, which made the NE tagging task harder.

Constructing a reliable, robust NE tagger for Dutch is an ongoing effort [17]; as part of our revision of the question classification phase, we are currently looking into usage of a state-of-the-art NE tagger for reducing this type of errors.

3.7 Wrong Unit Errors

Questions for which the answer is a number or a quantity are sometimes answered with a number that common sense would rule out. For example, for question *32. Hoeveel landen nemen deel aan de Internationale Conferentie over Bevolking en Ontwikkeling?* (English: How many countries take part in the International Conference on Population and Development?), the system produced the answer *miljard* (English: a billion). There are 12 such errors (5%), some of which may be handled by various sanity-checks and world-knowledge filters of the candidate answers, such as the usage of Cyc in [18]. We are currently building an ontology based type-checker for answer (currently for LOCATION questions only) that will address some of these problems.

3.8 Voting Errors

In the final stages of the QA pipeline, final answers are selected from a pool of candidates provided by the different streams. Our selection process between the candidates was very naive, giving preference to the Table and Pattern streams which we considered highly reliable, and using the confidence level of the other streams to compare their candidates. However, these confidence levels were not always comparable, since they originated in different sources; 8 wrong answers (3% of errors) originated from mismatches in the confidence levels of the streams and the simplified answer selection process.

Since our CLEF experiments we have changed the voting mechanism thoroughly. We now use Web hit counts for all streams, to normalize the confidence scores given by them; moreover, the voting process now uses weights based on the question type and the stream for deciding between the candidates in the

answer pool. The weights are learned from performance of the streams on a training set of questions, and initial experiments show significant improvements using this voting scheme.

3.9 Tiling Errors

Another step which is carried out at the final stages of the system is answer tiling. In this step, candidate answers which are similar (according to string similarity measures), or contain other candidates, or overlap with them, are joined to boost the confidence of the answer and to generate an answer which is more precise. For example, *Bill Clinton*, *president Clinton* and *former president Clinton* will be tiled to a single answer, *former president Bill Clinton*, which has higher confidence than any of the partial answers. In many cases the tiling process improves the answers given by the system, but 6 (3%) of the errors, mostly inexact ones, originate from this process:

Q031. Wat is de voornaam van Milosevic?	
(English:	What is Milosevic' first name?)
Candidate 1	Slobodan
Candidate 2	Slobodan Milosevic
Tiled Answer	Slobodan Milosevic

Addressing the tiling problem is tricky, since even humans will not necessarily agree whether *George Bush* and *George W. Bush* should be tiled to the same entity. We have done some refinement of our tiling process, but it is impossible to completely eliminate errors generated by it.

4 Conclusions

We presented our multi-stream question answering system and the runs it produced for CLEF 2003. Question answering is a multi-faceted problem, requiring contributions from information retrieval, natural language processing, and artifical intelligence. While addressing the question answering task will always leave room for improvement in most of the many modules required, an in-depth analysis of the types of wrong answers given by our system has revealed two major types: answer selection and named entity recognition. Both are on the language processing side of the spectrum, and both require a mixture of sufficient data and novel insights. In our ongoing QA research we are working on both of these long-term aspects. In our short-term work we are addressing the errors discussed with various strategies, and extending the various streams to handle them.

Acknowledgments

All three authors were supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. In addition, Maarten de Rijke was supported by grants from NWO, under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 612.000.207, and 612.066.302.

References

1. Monz, C., de Rijke, M.: Tequesta: The University of Amsterdam's textual question answering system. In Voorhees, E., Harman, D., eds.: The Tenth Text REtrieval Conference (TREC 2001), National Institute for Standards and Technology. NIST Special Publication 500-250 (2002) 519–528
2. Moldovan, D., Harabagiu, S., Girju, R., Morarescu, P., Lacatusu, F., Novischi, A., Badulescu, A., Bolohan, O.: LCC Tools for Question Answering. [19]
3. et al., M.B.: AskMSR: Question answering using the Worldwide Web. In: Proc. EMNLP 2002. (2002)
4. Jijkoun, V., Mishne, G., de Rijke, M.: Preprocessing Documents to Answer Dutch Questions. In: Proc. 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03). (2003)
5. Brants, T.: TnT – a statistical part-of-speech tagger. In: Proc. 6th Applied NLP Conference, ANLP-2000. (2000)
6. Oostdijk, N.: The Spoken Dutch Corpus: Overview and first evaluation. In: Proc. LREC 2000. (2000) 887–894
7. Magnini, B., Negri, M., Prevete, R., Tanev, H.: Is it the right answer? exploiting web redundancy for answer validation. In: Proc. 40th Annual Meeting of the Association for Computational Linguistics (ACL). (2002) 425–432
8. Jijkoun, V., Mishne, G., de Rijke, M.: The University of Amsterdam at QA@CLEF 2003. In: Working Notes for the CLEF 2003 Workshop. (2003)
9. Xu, J., Licuanan, A., May, J., Miller, S., Weischedel, R.: TREC 2002 QA at BBN: Answer selection and confidence estimation. [19]
10. Ittycheriah, A., Roukos, S.: IBM's Statistical Question Answering System – TREC-11. In: Proc. 11th Text REtrieval Conference. (2002)
11. Duclaye, F., Yvon, F., Collin, O.: Learning paraphrases to improve a question-answering system. In: Proc. EACL 2003 Workshop on NLP for QA. (2003)
12. Echihabi, A., Marcu, D.: A Noisy-Channel Approach to Question Answering. In Hinrichs, E., Roth, D., eds.: Proc. 41st Annual Meeting of the Association for Computational Linguistics. (2003) 16–23
13. Ravichandran, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proc. 40th ACL conference. (2002)
14. Zhang, D., Lee, W.: Question Classification using Support Vector Machines. In: Proc. 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM Press (2003) 26–32
15. Brill, E., Lin, J., Banko, M., Dumais, S., Ng, A.: Data-Intensive Question Answering. In: Text REtrieval Conference. (2001)
16. Lin, J., Fernandes, A., Katz, B., Marton, G., Tellex, S.: Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques (2002)
17. Sang, E.T.K., Meulder, F.D.: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In W. Daelemans and M. Osborne, ed.: Proc. of CoNLL-2003, Edmonton, Canada (2003) 142–147
18. Chu-Carroll, J., Prager, J., Welty, C., Czuba, K., Ferrucci, D.: A multi-strategy and multi-source approach to question answering. In: Proc. Eleventh Text REtrieval Conference (TREC 2002). (2002)
19. Voorhees, E., Harman, D., eds.: The Tenth Text REtrieval Conference (TREC 2002). In Voorhees, E., Harman, D., eds.: The Tenth Text REtrieval Conference (TREC 2002), National Institute for Standards and Technology. NIST Special Publication 500-251 (2003)