# AnswerFinder — Question Answering by Combining Lexical, Syntactic and Semantic Information

**Diego Mollá** and **Mary Gardiner**
Centre for Language Technology
Division of Information and Communication Sciences
Macquarie University
Sydney, Australia
diego@ics.mq.edu.au and gardiner@ics.mq.edu.au

## Abstract

We present a question answering system that combines information at the lexical, syntactic, and semantic levels, in the process to find and rank the candidate answer sentences. The candidate exact answers are extracted from the candidate answer sentences by means of a combination of information-extraction techniques (named entity recognition) and patterns based on logical forms. The system participated in the question answering track of TREC 2004.

## 1 Introduction

Question answering is an area that is becoming increasingly active in research and is currently being deployed into practical applications. Research in question answering has recently been fostered by large-scale programs like AQUAINT[1] and evaluation frameworks like TREC[2], NTCIR[3], and CLEF[4]. Such research and the current need to cope with large volumes of text has led various companies to produce practical question answering systems. For example, research groups from Microsoft, IBM, NTT, Oracle, and Sun have participated in the question answering track of TREC. In addition, there are several attempts to provide question-answering extensions to the current Web search engines, with demos available by MIT[5], LCC[6], and BrainBoost[7], among others.

AnswerFinder is an open-domain question answering system that combines information at the lexical, syntactic, and semantic levels in various stages to find the exact answer to the user question. This paper describes the AnswerFinder system as it stood at the time of the TREC 2004 question answering track. Section 2 introduces TREC and the question answering track. Section 3 describes the architecture of the system. Section 4 details the function of each module within the AnswerFinder system. Section 5 gives the system performance on the TREC 2003 question set. Section 6 mentions related work and Section 7 lists problems with the AnswerFinder system that should be addressed in the near future.

## 2 The TREC 2004 Question Answering Track

The Text REtrieval Conference (TREC) started in 1992 as part of the TIPSTER text program. A fundamental goal of the conference is to provide an evaluation framework for the comparison of the results of independent information retrieval systems. The concept of information retrieval is to be understood in a broad sense, and this conference has developed various tracks that focus on specific areas of information retrieval, such as ad-hoc (the name given to document retrieval), routing, speech, cross-language, web, video, and very large corpora (Voorhees, 2003).

The question answering track started in 1999 and ever since its creation it has been the most popular track. Every year the complexity and difficulty of the task increases. Thus, in 1999 the competing systems were asked to retrieve small snippets of text containing the answer. The questions were designed by the participants and the answer was guaranteed to be in the text corpus. In the 2004 competition, in contrast, the questions were extracted from logs of real questions, the answer is not guaranteed to be in the corpus, and the systems were asked to find the exact answers of factoid questions and list questions. The questions were grouped into targets, each target containing fact-based questions and list questions (explicitly marked as such), plus a question asking to find any other

[1] www.ic-arda.org/InfoExploit/aquaint/index.html
[2] trec.nist.gov
[3] research.nii.ac.jp/ntcir/index-en.html
[4] clef.iei.pi.cnr.it
[5] www.ai.mit.edu/projects/infolab/
[6] www.languagecomputer.com/
[7] www.brainboost.com/

information relevant to the target. The questions were encoded in XML as shown in Figure 1. In this example, the target is *Fred Durst*, so question with ID number 2.2 in the figure is asking *What record company is Fred Durst with?*.

```
<target id = "2" text = "Fred Durst">

<qa>
 <q id = "2.1" type="FACTOID">
  What is the name of Durst's group?
 </q>
</qa>

<qa>
 <q id = "2.2" type="FACTOID">
  What record company is he with?
 </q>
</qa>

<qa>
 <q id = "2.3" type="LIST">
  What are titles of the group's releases?
 </q>
</qa>

<qa>
 <q id = "2.4" type="FACTOID">
  Where was Durst born?
 </q>
</qa>

<qa>
 <q id = "2.5" type="OTHER">
  Other
 </q>
</qa>

</target>
```

Figure 1: A hand-made example of a group of questions using the TREC 2004 format

The corpus of supporting text was the AQUAINT corpus, which comprises over 1 million news articles taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. This corpus is not large in comparison with the terabites of text available via the Internet, but it is still large enough to require the need to resort to shallow-processing preselection methods before performing a real attempt to find the answer.

## 3 System Overview

The question answering procedure used by AnswerFinder follows a pipeline structure that is typical of rule-based question answering systems. The process is outlined in Figure 2 and is as follows:

1. All questions are normalised, so that *What record company is he with?* in Figure 1 becomes *What record company is Fred Durst with?*
2. The questions are classified into types based upon their expected answer. So the question *How far is it from Mars to Earth?* would be classified as a "Number" question as it expects a numeric value in response.
3. 100 candidate answer sentences are extracted from the corpus.
4. The 100 sentences are re-scored based upon their word overlap, grammatical relations overlap, and flat logical form overlap with the question text.
5. Exact answers —fragments like *416 million miles*— are extracted from the candidate answer sentences.
6. The exact answer list is sorted, re-scored and filtered for duplicate exact answers.
7. A number of exact answers from the top of the list are selected, depending on the question type.

AnswerFinder uses the following knowledge sources to analyse the question and to select from among possible answers:

**Named entity data** generated by the GATE system (Gaizauskas et al., 1996), marking pieces of text in the AQUAINT corpus as one of the types *Date*, *Location*, *Money*, *Organization*, and *Person*. These data are generated off-line before any question is processed. GATE's analysis was extended with a simple set of regular expressions that detect numbers as well.

**The list of preselected documents** provided by the US National Institute of Standards and Technology (NIST), containing for each target entity, the 1,000 top scoring documents for that entity. NIST co-sponsors the TREC conferences and it obtained the list of preselected documents by running the target query through the PRISE (Harman and Candela, 1990) document retrieval system.

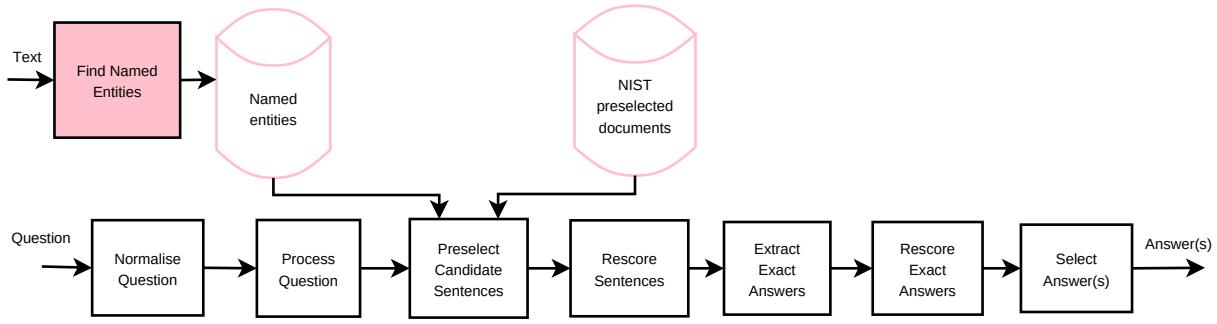


Figure 2: System overview

# 4 Modules

## 4.1 Question Normalisation

AnswerFinder relies heavily on the common information found between a question and the candidate answer sentence. Therefore questions like *What record company is he with?* need to undertake an anaphora resolution process to determine that *he* in fact refers to *Fred Durst*.

Questions in the TREC 2004 competition co-referred with previous questions or with their target in a number of ways.

Questions might co-refer with their target pronominally:

> **Target**: *Fred Durst*
>
> **Q**: *What record company is he with?*

Questions might co-refer with their target using a definite noun phrase:

> **Target**: *Club Med*
>
> **Q**: *How many Club Med vacation spots are there worldwide?*

Questions might co-refer with another question:

> **Target**: *Fred Durst*
>
> $\mathbf{Q}_{2.1}$: *What is the name of Durst's group?*
>
> $\mathbf{Q}_{2.3}$: *What are titles of the group's releases?*

Finally, questions may relate to their target associatively, that is, there may not be a direct co-reference:

> **Target**: *Heaven's Gate*
>
> **Q**: *When did the mass suicide occur?*

AnswerFinder normalises questions in the first case, where the question co-refers with the target pronominally. It performs a simple replacing of pronouns in the question with the target text, forming a regular plural and possessive where necessary, as shown in Table 1.

Finally, "other" type questions, which were of the generic form *other*, were transformed into *What is TARGET?* so that question 2.5 in Figure 1 is transformed into *What is Fred Durst?* This was a crude attempt at doing something useful with the "other" type questions. Clearly a more detailed processing of these questions is required.

## 4.2 Question Classification

Particular question words signal particular named entity types required as a response. The example below requires a person's name in response to the question:

> *Who founded the Black Panthers organization?*

AnswerFinder uses a set of 29 regular expressions to determine what named entity type a question requires in its response from the list *person, date, location, money, number, city, organization, percent, country, state, river, name, unknown*. The regular expressions were developed with the question set from TREC 2002, and they produced an accuracy of 78.6% correct classifications. This figure is lower than the one reported by other systems like the ones by Paşca and Harabagiu (2001) or Zhang and Lee (2003), each of which reported an accuracy of 90% or over. The question classification module clearly needs further refinement, but an evaluation with the question set from TREC 2003 showed an accuracy of 77%, thus indicating that the regular expressions generalise well.

| | | |
|---|---|---|
| *What record company is **he** with?* | $\longrightarrow$ | *What record company is **Fred Durst** with?* |
| *How many of **its** members committed suicide?* | $\longrightarrow$ | *How many of **Heaven's Gate's** members committed suicide?* |
| *In what countries are **they** found?* | $\longrightarrow$ | *In what countries are **agoutis** found?* |

Table 1: Examples of pronoun resolution performed by AnswerFinder

### 4.3 Candidate Sentence Extraction

Given the set of AQUAINT documents preselected by the NIST document retrieval system, AnswerFinder selects 100 sentences from these documents as candidate answer sentences.

Candidate sentences are selected in the following way:

1. The 1,000 preselected documents provided by NIST for each target are split into sentences by means of a simple sentence splitting process.
2. Each sentence is assigned a numeric score: 1 point for each distinct non-stopword overlapping with the question string, and 10 points for the presence of one or more named entities of the right type. This way we reward heavily the presence of a string of the expected answer type.
3. The 100 top scoring sentences are returned as candidate answer sentences.

As an example of the scoring mechanism, consider this question/sentence pair:

> **Q**: *How far is it from **Mars** to **Earth**?*
>
> **A**: *According to evidence from the SNC meteorite, which fell from **Mars** to **Earth** in ancient times, the water concentration in Martian mantle is estimated to be **40 ppm**, far less than the terrestrial equivalents.*

The question and sentence have 2 shared non-stopwords: *Mars* and *Earth*. Further, this sentence has a named entity of the required type (Number): *40 ppm*, making the total score for this sentence 12 points.

### 4.4 Sentence Re-Scoring

The goal of all the above modules is to reduce the corpus of text to a list of the 100 sentences with highest likelihood to contain an answer. The sentence re-scoring module uses a combination of lexical, syntactic, and semantic information to perform a more detailed analysis of these sentences:

**lexical:** The combined word overlap and named entity score.

**syntactic:** The grammatical relation overlap score.

**semantic:** Overlaps with flat logical form patterns.

We have seen the use of lexical information in Section 4.3. Below we will see the use of grammatical relations and flat logical form patterns, and the final combinations used in TREC 2004.

#### 4.4.1 Grammatical Relation Overlap Score

The grammatical relations were initially devised by Carroll et al. (1998) as a means to normalise the output of parsers for their comparative evaluation. The set of grammatical relations represent some of the common relations that exist between the words in a sentence, a selection of which is shown in Table 2. To build the grammatical relations of questions and answer candidate sentences, AnswerFinder processes the output of the Connexor Dependency Functional Grammar, which is a dependency-based robust parser with a wide-coverage grammar of English (Tapanainen and Järvinen, 1997). Below is an example of the grammatical relations of a question and an answer candidate sentence.

> **Q**: *How far is it from Mars to Earth?*
> **(subj be it _)**
> (xcomp from be mars)
> (ncmod _ be far)
> (ncmod _ far how)
> **(ncmod earth from to)**
>
> **A**: *It is 416 million miles from Mars to Earth.*
> **(ncmod earth from to)**
> **(subj be it _)**
> (ncmod from be mars)

| *Relation* | *Description* |
|---|---|
| CONJ(type,head+) | Conjunction |
| MOD(type,head,dependent) | Modifier |
| CMOD(type,head,dependent) | Clausal modifier |
| NCMOD(type,head,dependent) | Non-clausal modifier |
| DETMOD(type,head,dependent) | Determiner |
| SUBJ(head,dependent,initial_gr) | Subject |
| OBJ(head,dependent,initial_gr) | Object |
| DOBJ(head,dependent,initial_gr) | Direct object |
| XCOMP(head,dependent) | Clausal complement without an overt subject |

Table 2: Grammatical relations used in this paper

(xcomp _ be mile)
(ncmod _ million 416)
(ncmod _ mile million)

The score is the number of relations shared between question and sentence. In the example above, the overlap between the grammatical relations of question and candidate sentence is 2, corresponding to the two grammatical relations marked in boldface.

#### 4.4.2 Flat Logical Form Patterns

In previous research we have developed a flat notation for the logical forms of sentences and a method to produce the logical forms from arbitrary sentences by traversing their syntactic structures (Mollá, 2001; Mollá and Hutchinson, 2002). These flat logical forms have been used to determine the likelihood that a sentence contains the answer by checking the semantic similarity of the question with the sentence. In a similar fashion to grammatical relations, the semantic similarity of two sentences is the number of logical terms shared between them. Thus if we have the following logical forms:

**Q**: *What is the population of Iceland?*
object(iceland, o6, [x6])
**object(population, o4, [x1])**
object(what, o1, [x1])
**prop(of, p5, [x1, x6])**
**A**: *Iceland has a population of 270000*
dep(270000, d6, [x6])
**object(population,o4,[x4])**
object(iceland,o1,[x1])
evt(have,e2,[x1,x4])
**prop(of,p5,[x4,x6])**

The semantic similarity between the two sentences is 2, as the number of overlaps between the logical form of question and answer is 2 (overlap shown in boldface). Note that the computation of the overlap is complicated by the fact that logical terms include variables and it is necessary to keep the relation between the variables in the overlapping terms. Thus, in the example above, the variable **x1** in the question terms corresponds with **x4** in the answer candidate sentence and therefore whenever **x1** is used in the question, **x4** must be used in the answer. A simple process of Prolog unification suffices to match the variables of the question terms with those of the sentence terms, by converting the question term variables into real Prolog variables.

Since there are several ways to answer a question, for TREC 2004 we have developed a set of patterns to capture the expected logical form of sentences that contain the answer to questions. Below is the matching pattern associated with the template that we labelled as "what2" and one of its replacement patterns:

**Template "what2"** :
**Pattern:**
object(ObjX,VobjX,[VeX]),
object(what,_,[VeWHAT]),
object(ObjY,VobjY,[VeWHAT]),
prop(of,_,[VexistWHAT,VeX])
**Replacement 1:**
dep(ANSWER,ANSW,[VeANSW]),
prop(of,_,[VeY,VeANSW]),
object(ObjX,VobjX,[VeX]),
evt(have,_,[VeX,VeWHAT]),
object(ObjY,VobjY,[VeY])

Borrowing the notation of Prolog variables, the above template uses forms in uppercase or "_" to express the slots that can unify with logical form components. As the logical form of *What is the population of Iceland?* matches the

pattern above (we use standard Prolog unification to perform the matching), then its logical form is transformed into:

**Q**: *What is the population of Iceland?*
dep(ANSWER,ANSW,[VeANSW]),
prop(of,_,[VeY,VeANSW]),
object(iceland,o6,[x6]),
evt(have,_,[x6,x1]),
object(population,o4,[VeY])

The semantic similarity between this logical form and the one of *Iceland has a population of 270000* is now 5, since all five terms of the modified question logical form can be found in the logical form of the answer and all variables unify.

In addition to returning the overlap between a candidate sentence and a matching answer pattern, AnswerFinder uses the instantiation of the ANSWER variable to determine the answer: "270000" in the case of our example.

The introduction of flat logical form patterns parallels the use of patterns based on regular expressions, but using the logical level of a sentence instead of the surface level. This way it is hoped that less patterns are required to cover a broader range of sentences. In practice, however, the difficulty to read logical forms by humans slows down the production of patterns and replacements. As a result, a small set of 10 patterns were developed for our experiments in TREC 2004. As we can see in Table 3, most of the questions from the TREC 2004 test set were covered by only 4 template patterns and there was an important number of questions that did not trigger any pattern.

| *Template ID* | *Num.* | Template ID | *Num.* |
|---|---|---|---|
| howmany1 | 0 | how1 | 1 |
| howmany2 | 0 | who_generic | 39 |
| what2 | 3 | what_generic | 116 |
| what3 | 0 | what_noun | 69 |
| what6 | 1 | *no match* | 78 |
| when1 | 47 | | |

Table 3: Number of questions triggering each template; a question may trigger several templates

The patterns that were triggered most frequently were generic patterns that were introduced to maximise the coverage of the pattern set. For example, the most frequent pattern, "what_generic", is defined so as to allow any noun to replace the word *what*:

**Template "what_generic"** :
**Pattern:**
object('what',_,[XWho])
**Replacement:**
object(_,ANSWER,[XWho])

### 4.5 Exact Answer Extraction, Filtering and Scoring

Having selected and re-ranked the 100 top-scoring candidate sentences, AnswerFinder then selects exact answer strings from within them. AnswerFinder combines the use of named entities with that of logical form patterns:

1. For each candidate sentence, extract all named entities that match the question classification.
2. For each candidate sentence, extract ANSWER values from any matching flat logical form pattern.

Exact answers are scored as follows:

1. If the exact answer is a named entity, its score is the score of the candidate sentence it is found in.
2. If the exact answer is an ANSWER value from a flat logical form pattern, its score is the score of the candidate sentence it is found in.
3. If the exact answer is both a named entity and an ANSWER value from a flat logical form answer pattern, its score is twice the score of the candidate sentence it is found in.

If the same string is extracted from two answer sentences the score becomes the sum of the scores of the duplicate answers. This way answer redundancy is rewarded.

### 4.6 Exact Answer Selection

AnswerFinder selects the answers depending on the type of question:

**Factoid questions** requiring exactly one answer: return the top scoring answer; or if there are no answers with a score more than 0, return "NIL" indicating that there were no answers.

**List questions and "other" questions** requiring a number of answers: return all exact answers within a threshold difference in score with the top score. If there are no exact answers with a score of more than 0, return the top scoring candidate sentence.

## 5 Performance

After testing several combinations of lexical, syntactic, and semantic information (see the work by Mollá (2003) for the sort of analysis that we performed), AnswerFinder used two combinations of scores for the runs submitted to TREC 2004:

**3gro+lfo** 3 times the grammatical relation overlap score added to the flat logical form pattern overlap score. This combination was chosen because it gave the best results in our preliminary experiments with question sets taken from past TREC QA conferences.

**lfo** The flat logical form pattern overlap score.

Although not explicitly expressed in the above combinations, lexical information is used implicitly because the scoring is based on the output of a preselection module that did use solely lexical information (word overlap and named entities), as we have seen in Section 4.3.

In a preliminary analysis of the system we used the answer patterns provided by Ken Litkowsky via NIST. These answer patterns cover all the answers found by the systems participating in TREC 2003. We tested our system with the TREC 2003 questions and checked the output of the sentence re-scoring module and the final output of the system. We found that the re-scoring module gave the highest score to a sentence containing the answer about 20% of times. In contrast, the final system returned a correct and exact answer about 5% of times.

The results of our participation in TREC 2004 are significantly better. In all runs, the accuracy of the factoid questions is 10%, the F-score of the list questions is 0.08, and the F-score of the "other" questions is 0.09. Since the system was not fine-tuned for the list or "other" questions only the results of the factoid questions need to be considered. We believe that the reason for the better results of the factoid question with respect to our preliminary analysis is that Litkowsky's patterns that we used did not cover all cases of good answers. In fact, we tried the patterns on the answers submitted to TREC 2004 and we obtained an accuracy of 8.59%, which is between the accuracy given by our preliminary experiments and the one given by the TREC human assessors. Comparatively with the other systems participating in TREC 2004, the results are below the median of the results returned by all the systems (which was 17%). Also, surprisingly, all of our runs had virtually the same results. These unusual results led us to suspect that a chain of bugs may have made the system ignore the information provided by the logical form patterns. Currently we are analysing the results.

## 6 Related Work

AnswerFinder as it stood in TREC 2004 differs from previous versions in several aspects. First of all, now AnswerFinder uses the Named Entity data that has been pre-calculated on the entire AQUAINT corpus. This way the system does not need to spend precious time during the on-line stage when the user is waiting for the answer of the question. Also, in contrast with AnswerFinder's participation in TREC 2003 where it focused on the extraction of passages containing the answer, now AnswerFinder extracts exact answers and attempts to answer list and definition questions. In the process, AnswerFinder uses a set of templates based on patterns of logical forms.

The overall architecture of AnswerFinder is similar to that of other question answering systems. The aim is to gradually reduce the amount of text to process through several levels of increasing complexity. We use an information retrieval system to preselect the documents and information from named entities and the expected answer type obtained from the question to reward the sentences that may contain the answer. One difference that sets our system apart from the majority is the use of logical forms in the process to further scope the sentences that are most likely to contain the answer. Other question answering systems use logical forms (Harabagiu et al., 2001, for example) that were developed independently from our research.

But the main difference with respect to other systems is the use of patterns derived from logical forms to determine the exact answer. A baseline method that would return the text tagged by the named entity recogniser has been used by various systems. Adding further com-

plexity, systems like the one developed by Echihabi et al. (2004) use patterns based on named entities and parts of speech. However, we are not aware of any other system besides AnswerFinder that tries to use logical form patterns.

## 7 Conclusions and Further Work

AnswerFinder is a question answering system that uses a combination of lexical, syntactic, and semantic information to find the answer to the user question. An early version of this system participated in the passages section of the 2003 TREC question answering track. After the equivalent of only 55 person-hours work, the system ranked above the median of the seven participating systems. For TREC 2004 we have included a named entity recogniser and a process to find exact answers that uses a combination of patterns based on logical forms and named entities.

Current and future work focuses on the refining of the candidate sentence scoring, exact answer scoring, and pattern development. We also plan to work on a more detailed processing of list and definition questions.

For the refining of the sentence scoring, we are exploring the use of weighted measures for different types of terms in the flat logical forms. We are also exploring the integration of graph-based methods such as the ones developed by (Montes-y-Gómez et al., 2001).

For the exact answer scoring, we are developing further logical form patterns to increase their coverage. We will also explore fuzzy matching methods so that every question will match at least one pattern.

To facilitate the discovery and development of logical form patterns, we are studying methods to increase the readability of the flat logical forms by converting them into graph structures.

## References


John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC98*.

Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz, and Deepak Ravichandran. 2004. How to select an answer string? In Tomek Strzalkowski and Sanda Harabagiu, editors, *Advances in Textual Question Answering*. Kluwer.

Robert Gaizauskas, Hamish Cunningham, Yorick Wilks, Peter Rodgers, and Kevin Humphreys. 1996. GATE: an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's question-answering server. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC 2001*, number 500-250 in NIST Special Publication. NIST.

Donna K. Harman and Gerald Candela. 1990. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science*, 41(8):581–589.

Diego Mollá and Ben Hutchinson. 2002. Dependency-based semantic interpretation for answer extraction. In *Proc. 2002 Australasian NLP Workshop*.

Diego Mollá. 2001. Ontologically promiscuous flat logical forms for NLP. In Harry Bunt, Ielka van der Sluis, and Elias Thijsse, editors, *Proceedings of IWCS-4*, pages 249–265. Tilburg University.

Diego Mollá. 2003. Towards semantic-based overlap measures for question answering. In *Proc. ALTW03*, pages 130–137, Melbourne.

Manuel Montes-y-Gómez, Alexander Gelbukh, and Ricardo Baeza-Yates. 2001. Flexible comparison of conceptual graphs. In *Proc. DEXA-2001*, number 2113 in Lecture Notes in Computer Science, pages 102–111. Springer-Verlag.

Marius A. Paşca and Sanda M. Harabagiu. 2001. High performance question answering. In *Proc. SIGIR'01*, New Orleans, Luisiana, USA. ACM.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proc. ANLP-97*. ACL.

Ellen M. Voorhees and Lori P. Buckland, editors. 2003. *The Twelfth Text REtrieval Conference (TREC 2003)*, number 500-255 in NIST Special Publication. NIST.

Ellen M. Voorhees. 2003. Overview of TREC 2003. In Voorhees and Buckland (Voorhees and Buckland, 2003).

Dell Zhang and See Sun Lee. 2003. Question classification using support vector machines. In *Proc. SIGIR 03*. ACM.