AnswerFinder at TREC 2004

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

AnswerFinder combines lexical, syntactic, and semantic information in various stages of the question answering process. The candidate sentences are preselected on the basis of (i) the presence of named entity types compatible with the expected answer type, and (ii) a score combination of the overlap of words, grammatical relations, and flat logical forms. The candidate answers, in turn, are extracted from (i) the set of compatible named entities and (ii) the output of a logical-form pattern matching algorithm.

1 Introduction

This document describes the AnswerFinder system as it stood at the time of the TREC 2004 question answering track.[1] Section 2 describes the architecture of the system as a whole, Section 3 details the function of each module within the AnswerFinder system, Section 4 gives the system performance on the TREC 2004 question set and Section 5 gives the conclusions and future work.

The AnswerFinder question answering system is designed to answer TREC-style factoid questions with an exact answer as below:

Q: *How far is it from Mars to Earth?*

A: *416 million miles*

[1]This work is supported by the Australian Research Council, ARC Discovery grant n. DP0450750.

In 2004, TREC questions consisted of factoid, list, and "other" questions and they were XML-formatted in groups of questions about a single topic as shown in Figure 1.

All the questions in the example of Figure 1 refer to *Fred Durst*, so question with ID number 2.2 is a factoid question asking *What record company is Fred Durst with?*. Question with ID number 2.3 is a list question in which all correct factoids should be listed in response, and question with ID number 2.5 is an "other" question in which relevant nuggets of information not given in response to earlier questions should be returned.

We spent most of our effort to handle the factoid questions and very little effort was done for the list and "other" questions. In this paper we will therefore focus on AnswerFinder's handling of factoid questions.

2 System Overview

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

1. The questions are normalised by resolving pronominal anaphora, so that *What record company is he with?* 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

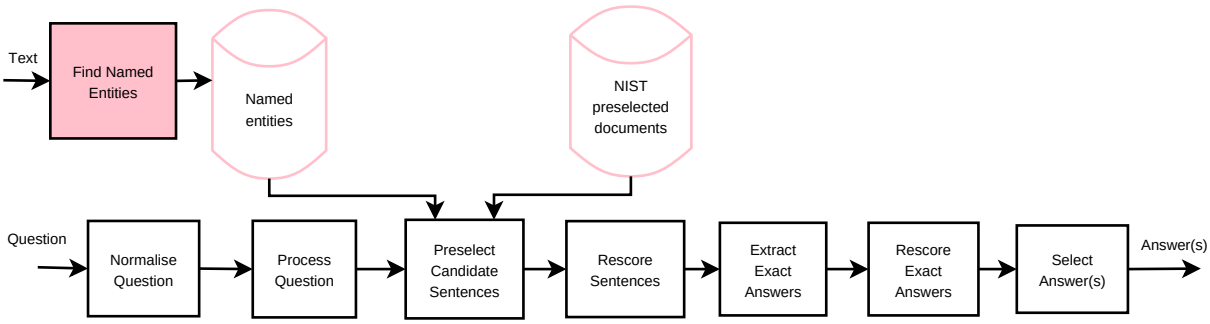

Figure 2: System overview

```
<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: An example of a group of questions using the TREC 2004 format

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 being factoid, list, or "other".

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). The named entity types are listed in Table 1. The named entity detector was run off-line and all the named entities of the AQUAINT corpus was stored in a set of files prior to the processing of any questions.

- The list of documents provided by NIST, containing for each target entity the 1000 top scoring documents for that entity.

Date
Location
Money
Organization
Person

Table 1: Named entity types recognised by GATE

3 Modules

3.1 Question Normalisation

Given that the questions may contain anaphoric references to information external to the questions, AnswerFinder performs simple anaphora resolution on question strings.

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$_2$**:** *Fred Durst*

Q$_{2.2}$**:** *What record company is he with?*

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

Target$_{21}$**:** *Club Med*

Q$_{21.1}$**:** *How many Club Med vacation spots are there worldwide?*

Q$_{21.2}$**:** *List the spots in the United States.*

Questions might co-refer with another question:

Target$_2$**:** *Fred Durst*

Q$_{2.1}$**:** *What is the name of Durst's group?*

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$_{46}$**:** *Heaven's Gate*

Q$_{46.3}$**:** *When did the mass suicide occur?*

AnswerFinder normalises questions in the first case only, where the question co-refers with the target pronominally. It performs a simple replacing of the pronouns of the question with the target's identity. Since AnswerFinder needs syntactically correct questions, the target is transformed to the plural form or the possessive form where necessary. The transformation uses very simple morphological rules to transform the questions as shown in Table 2.

In addition, "other" type questions containing the single word *Other* are transformed into *What is TARGET?* so that question 2.5 in Figure 1 is transformed into *What is Fred Durst?*. This is a very crude attempt at doing something useful with the "other" type questions. Clearly more targeted processing of these questions is advisable.

3.2 Question Classification

Particular questions signal particular named entity types expected as responses. Thus, the example below expects 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 the expected named entity type. These regular expressions are the same used in our contribution to TREC 2003 and they target the occurrence of *Wh-* question words. In addition, specific keywords in the questions indicate expected answer types as shown in Table 3.

3.3 Candidate Sentence Extraction

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

Candidate sentences are selected in the following way:

1. The 1000 preselected documents provided by NIST for each target are split into sentences. The sentence splitter follows a simple approach based on the use of a fixed list of sentence delimiters.

*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 2: Examples of pronoun resolution performed by AnswerFinder

Keyword	Answer Type
city, cities	Location
percentage	Number
first name, middle name	Person
range	Number
rate	Number
river, rivers	Location
what is, what are, what do	Person, Organisation or Location
how far, how long	Number

Table 3: Examples of question keywords and associated answer types

2. Each sentence is assigned a numeric score: 1 point for each non-stopword overlapping with the question string, and 10 points for the presence of a named entity 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.

3.4 Sentence Re-scoring

The 100 candidate sentences are re-scored based upon the combination of lexical, syntactic, and semantic features:

Lexical: The combined word overlap and named entity score.

Syntactic: The grammatical relation overlap score.

Semantic: Two ways of computing overlaps with flat logical form patterns.

The use of lexical information has been described in Section 3.3. Below we will briefly explain the use of syntactic and semantic information.

3.4.1 Grammatical Relation Overlap Score

The grammatical relations devised by Carroll et al. (1998) encode the syntactic information of questions and candidate answer sentences. We decided to use grammatical relations and not, say, parse trees or dependency structures for two reasons:

1. Unlike parse trees, and like dependency structures, grammatical relations are easily incorporated into an overlap-based similarity measure.

2. Parse trees and dependency structures are dependent on the grammar formalism. In contrast, the grammatical relations have been devised as a means to normalise the output of

parsers for their comparative evaluation. As a result, the grammatical relations are independent of the grammar formalism or the actual parser used. Our choice of parser was the Connexor Dependency Functional Grammar and parser (Tapanainen and Järvinen, 1997). Since it is dependency-based, the transformation to grammatical relations is relatively straightforward.

An example of the grammatical relations for question and candidate sentence follows:

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

The similarity-based score is the number of relations shared between question and sentence. In example 3.4.1, the overlap between the grammatical relations of question and candidate sentence is 2: (subj be it _) and (ncmod earth from to).

3.4.2 Flat Logical Form Patterns

Semantic information is represented by means of flat logical forms (Mollá, 2001). These logical forms use reification to flatten out nested expressions in a way similar to other QA systems (Harabagiu et al., 2001; Lin, 2001; Mollá et al., 2000, for example). The logical forms are produced by means of a process of bottom-up traversal of the dependency structures returned by Connexor (Mollá and Hutchinson, 2002).

Our contribution to TREC 2003 (Mollá, 2003) manipulated flat logical forms in the same way as grammatical relations are manipulated above, so that the score of example 3.4.2 would be computed as follows:

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])

In our 2003 system, the flat logical form score of example 3.4.2 would have been 2, as the number of overlaps between the logical form of question and answer is 2. Note that the process to compute the overlap of logical forms must map the variables from the question to variables from the candidate answer sentence. AnswerFinder uses Prolog unification for this process.

With the goal to take in consideration the differences between a question and the various forms to answer it, AnswerFinder in TREC 2004 uses patterns to capture the expected form of the answer sentence and locate the exact answer. Thus, if a question is of the form *what is X of Y?*, then a likely answer can be found in sentences like *Y has a X of ANSWER*. In contrast with other approaches, AnswerFinder uses flat logical form patterns. For example, the pattern for *what is X of Y?* is:

Question Pattern:
object(ObjX,VobjX,[VeX]),
object(what,_,[VeWHAT]),
object(ObjY,VobjY,[VeWHAT]),
prop(of,_,[VexistWHAT,VeX])

And the pattern of *Y has a X of ANSWER* is:

Answer Pattern:
dep(ANSWER,ANSW,[VeANSW]),
prop(of,_,[VeY,VeANSW]),
object(ObjX,VobjX,[VeX]),
evt(have,_,[VeX,VeWHAT]),
object(ObjY,VobjY,[VeY])

Borrowing Prolog notation, the above patterns use uppercase forms or '_' to express the arguments that can unify with logical form arguments.

As the logical form of *What is the population of Iceland?* matches the above question pattern, 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])

Now the transformed logical form shares all five terms with the logical form of *Iceland has a population of 270000*, hence the score of this answer sentence is 5. In addition, AnswerFinder knows that the answer is *270000*, since this is the value that fills the slot `ANSWER`.

The use of flat logical forms makes it possible to handle certain types of paraphrases (Rinaldi et al., 2003) and therefore we believe that patterns based on flat logical forms such as the ones described above are more appropriate than patterns based on syntactic information or surface strings for the task of identifying answer sentences. However, the resulting patterns are difficult to read by humans and the process of developing the patterns becomes very time-consuming. Due to time constraints we developed 10 generic templates only. Each template consists of a pattern to match the question, and one or more replacement patterns. The above example is one of the question/replacement patterns.

There were two flat logical form pattern overlap algorithms used by AnswerFinder in TREC 2004: an algorithm which can extract one value for the ANSWER variable, and an algorithm which can extract all possible values.

3.5 Exact Answer Extraction, Filtering and Scoring

Having selected and re-ranked the 100 candidate sentences, AnswerFinder then selects an exact answer string from within them. Ideally, the logical form patterns will have identified the exact answer. In practise, given the reduced number of patterns developed, many good answer sentence or even questions would not have a logical form that matches any of the patterns. As a result we integrate the expected answer type provided by the question analyser and the results of the named entity recogniser. In particular:

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 in this way:

1. If the exact answer is an identified named entity of the expected answer type, 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 answer 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 an answer is a duplicate of a higher scoring answer, the two answers are merged and the score becomes the sum of the scores of the duplicate answers.

3.6 Exact Answer Selection

AnswerFinder selects the answers in the following ways:

Factoid questions requiring exactly one answer, or "NIL" indicating no answer:

- if there are no answers with a score more than 0, return "NIL"; otherwise
- return one of the top scoring answers

List questions and "other" questions requiring a number of answers: return all top scoring exact answers. If there is no exact answer with a score of more than 0, return the top scoring candidate sentence.

4 Performance

We submitted three runs based on combinations of candidate answer sentence re-ranking scores and the use or not of all the possible answers returned by the logical form patterns:

answfind1 used the overlap of logical forms (lfo) and the logical form patterns returned the first possible answer (*single*).

answfind2 used 3 times the grammatical relation overlap score added to the flat logical form pattern overlap score ($3gro+lfo$). This combination was determined empirically. The logical form patterns returned the first possible exact answer (*single*).

answfind3 used the same score combination as "answfind2" ($3gro + lfo$), but this time the logical form patterns returned all the possible exact answers (*multi*).

The results of the three runs (Table 4) were surprisingly similar. The F measure of the list question is remarkably close to the median considering our simple treatment, and the F measure of the "other" questions is understandably low given that we treated them as list questions of the form *what is TARGET?*.

Run Name	*Factoid Accuracy*	*List F*	*"Other" F*	*Final Score*
answfind1	0.100	0.081	0.080	0.090
answfind2	0.100	0.080	0.080	0.090
answfind3	0.100	0.080	0.080	0.090
TREC	0.170	0.094	0.184	

Table 4: TREC 2004 results

A subsequent analysis of the code revealed a bug that made the program to effectively ignore the use of the logical form patterns. This explains the nearly similar results in all runs. After fixing the bug the accuracy of our system changed as shown in Table 5. The evaluation was performed using Ken Litkowsky's patterns. Since this is an automatic evaluation, the results had lower accuracy than the ones presented in Table 4, but the relative increase or decrease of accuracy was significant.

Run Name	*Before Fixes*	*After Fixes*	*Accuracy Increase*
answfind1	0.086	0.071	-17%
answfind2	0.086	0.066	-23%
answfind3	0.086	0.096	11%

Table 5: Automatic analysis of factoid accuracy with Ken Litkowsky's patterns

The results of our internal evaluation indicate that the first answer returned by a logical form pattern is not usually the correct one. As a result, incorrect answers were given a score boost. The main reason for logical form patterns not giving the correct answer is that some of them were so general that they identified a large set of possible answers within a sentence. As an indication, Table 6 shows the coverage of the patterns used.

Template ID	*Number of Questions*	Template ID	*Number of Questions*
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 6: Number of questions triggering each question pattern; note that a question may trigger several patterns

The most frequent pattern by far is "what_generic", which was defined as:

Question Pattern:
object(what,_,[XWho])

Answer Pattern:
object(_,ANSWER,[XWho])

Clearly, this pattern will match any object of the candidate answer sentence. In the "answfind1" and "answfind3" runs, more often than not the answer chosen was the wrong one. However, in the "answfind3" run, since all matching answers were returned, those that were of the correct expected answer type would have an additional score boost.

Our results also indicate that even a small set of patterns such as the ones used in our system can help to obtain better results. By extending the set of patterns and reducing the use of general patterns the overall performance is expected to increase.

5 Conclusions and Future Work

AnswerFinder combines lexical, syntactic, and semantic information in a simple pipeline-based system. Lexical information is based on the list of non-stop words. Syntactic information is based on the use of grammatical relations. Semantic information is based on the use of flat logical forms. The main differences from the system participating in TREC 2003 are the use of named entities and the development of a process to handle logical form patterns and exact answer extraction.

Further work includes the fine-tuning of various parameters used by AnswerFinder such as the ideal combination of the lexical, syntactic, and semantic information. Time and resource permitting we will use machine learning methods to optimise the weights of each element in the final scoring formula. Another parameter that may be optimised by means of machine learning methods is the scoring threshold for NIL answers and for returning an answer to a list question.

An obvious direction of further research is the treatment of "other" questions. The transformation from *other* to *what is TARGET?* performed by AnswerFinder assumes that these questions are answered as definitions but this is not necessarily the case. We will explore the use of logical form patterns inspired in methods to answer definition questions.

Additional further research includes the refining of the candidate sentence scoring and the exact answer scoring, and the development of a more comprehensive and detailed set of logical form patterns.

References

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

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.

Jimmy J. Lin. 2001. Indexing and retrieving natural language using ternary expressions. Master's thesis, MIT.

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

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *Traitement Automatique des Langues*, 41(2):495–522.

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. Answerfinder in trec 2003. In *Proc. TREC 2003*.

Fabio Rinaldi, James Dowdall, Kaarel Kaljurand, Michael Hess, and Diego Mollá. 2003. Exploiting paraphrases in a question answering system. In *Proc. Workshop in Paraphrasing at ACL2003*, Sapporo, Japan.

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