

# AnswerFinder: Question Answering from your Desktop

**Mark A. Greenwood**

Department of Computer Science

University of Sheffield

Regent Court, Portobello Road

Sheffield S1 4DP UK

m.greenwood@dcs.shef.ac.uk

## Abstract

For many years Internet search engines have made a valiant attempt at quickly finding documents relevant to a users query. As the size of the Internet continues to grow, however, these search engines are returning more and more documents for a single query, leaving the user to wade through a vast amount of text. What is required are a new range of systems that are not only easy to use but are capable of returning just the answer to the user's question. The AnswerFinder application outlined in this paper attempts to meet both of these requirements.

## 1 Introduction

Question Answering (QA) is by no means a new field (Simmons (1965) reviews no less than fifteen English language QA systems) although it has undergone a resurgence in recent years due, in no small part, to the introduction of a QA evaluation at the annual Text REtrieval Conference (TREC).

The introduction of the QA evaluation at TREC has not only provided a framework in which systems can be evaluated against each other but has also resulted in the construction of invaluable data sets which can be used to aid further research.

The majority of the questions used to evaluate systems participating in TREC and the questions we aim to answer are simple questions requiring a single fact as answer, e.g. "*When was Mozart born?*" and "*Who invented the paper clip?*" – interested readers should consult Voorhees (1999; 2000; 2001; 2002) for further details of the TREC QA evaluations.

Many of these systems have made use of a fine grained system of answer types as part of more complex systems. For example the system detailed in (Harabagiu et al., 2000) has an answer type DOG BREED and the answer typology described in (Hovy et al., 2000) contains 94 different answer types. AnswerFinder was originally developed as a baseline

system for entry in the 2003 TREC QA evaluation with the aim of determining how well a system could perform which used only fine grained answer types to locate possible answers. AnswerFinder was developed into a general purpose web-based question answering tool when developmental testing over the TREC 2002 question set showed that it was capable of correctly answering approximately 26% of the questions.

The remainder of this paper is divided into two sections. Firstly the underlying question answering system will be outlined in some detail to give an idea of how answers are found and ranked. The remainder of the paper will then give details of the AnswerFinder application.

## 2 System Description

The question answering system, underlying AnswerFinder, consists of three main phases (outlined in the following sections) namely determining the expect answer type, using a search engine to find relevant documents and a final stage in which possible answers are located within the relevant documents.

### 2.1 Question Typing

The first stage of processing attempts to determine the expected answer type. This is currently carried out using a set of hand coded rules which work over the question text.

These rules form a hierarchy in which rules can only be fired if their parent rule has fired. The top level of this hierarchy consists almost entirely of the main question words of English; who, when, where, why and how. This division is useful as certain of these words immediately suggest an answer type:

**who:** suggests that the answer will be a person, company name or a job title.

**when:** suggests that the answer will be a date.

**where:** suggests that the answer will be a location.

It should, however, be clear that these rules do not cover many of the ways in which questions can be formulated. Further rules are required which can be just as simple (i.e. a single word) or slightly more complicated, i.e. multiple words, a phrase, the absence of word(s), a named entity or a combination of these. For example, the following are questions and the rules which assigned their expected answer type:

**Question:** Name the author of ‘Jane Eyre’.

**Rule:** if question contains *author* then the expected answer type is *Person*.

**Question:** How much are tickets to Disneyland?

**Rule:** if question contains *how much* and *ticket* then the expected answer type is *Money*.

Determining these rules is a relatively simple but extremely time consuming task.

The expected answer types are currently constrained to being entities which the question answering system is able to recognise. Entities are currently recognised using modified versions of the gazetteer lists and named entity transducer supplied with the GATE 2 framework<sup>1</sup> (Cunningham et al., 2002). This allows us to not only recognise the standard named entities (Person, Location, Organization, etc.) but also many other entities such as planets, birthstones, measurements, gods, state flowers. The system can currently recognise 46 different types of entity. Clearly the more entities that can be recognised the more questions the system can attempt to answer.

Limiting the answer types to entities in this way is a significant drawback as there are many questions for which the answer is not an entity, such as “*How did Patsy Cline die?*” for which the answer is “*in a plane crash*”. Surprisingly this method still allows us to determine the expected answer type for a wide range of questions (see section 3.1).

If the system is unable to determine the answer type then the user is informed of this, otherwise processing continues into the second stage.

## 2.2 Information Retrieval

The information retrieval stage is dependent on the collection from which answers are being drawn. For answering the TREC 2002 questions, Okapi<sup>2</sup> was used to access the AQUAINT collection, whereas

when using AnswerFinder as a general purpose question answering system Google is used to search for relevant documents on the Internet.

Regardless of which search engine–collection combination is being used the query is the same; simply the question, exactly as entered by the user. Assuming we manage to find at least one relevant document then we continue on to the final stage in which the answers are located.

## 2.3 Locating Possible Answers

As has already been discussed, the only answers we can currently locate are entities which the system can recognise. This means that locating possible answers is simply a case of extracting all the entities, of the correct type, from the documents returned by the information retrieval system.

These entities are then retained as possible answers, unless they fail one of the following tests:

- If the document in which the entity was found does not contain all the named entities that appear in the question, then the document and any entities occurring within it are discarded.
- Any entities which overlap in any way with the words in the question (ignoring question specific stopwords) are discarded. This is so that for questions such as “*Where is Mount Everest?*”, the system cannot return “*Mount Everest*” as a possible answer.

Any entities that still remain are grouped to form *answer groups* based on the following equivalence test (Brill et al., 2001): *Two answers are said to be equivalent if all of the non-stopwords in one are present in the other or vice versa.*

These answer groups are then ordered based firstly upon the number of occurrences of each answer group<sup>3</sup> within the documents that were processed and then (where necessary) on the rank of the document in which the answer was first found, giving precedence to those answers found in documents regarded as highly relevant by the information retrieval system. This ordered list is then presented to the user of the system.

1. <http://gate.ac.uk>

2. <http://www soi.city.ac.uk/~andym/OKAPI-PACK/>

3. This is equal to the sum of the occurrences of each unique answer within the group.

### 3 System Evaluation

Evaluation of the underlying question answering system was carried out over the five hundred factoid questions used in the TREC 2002 question answering track, which had not been used during development of the system. This allows us to easily compare the results with those from the best performing question answering systems entered in that evaluation<sup>4</sup>.

To be able to pinpoint any major flaws in the system the three separate stages of processing will be evaluated as well as the final outcome of the system.

#### 3.1 Question Typing

Evaluating the output of the question typing stage revealed the following:

- 16.80% (84/500) of the questions could not be typed.
- 1.44% (6/416) of the questions that were typed were assigned an incorrect type.

Given this information it should be clear that the maximum score attainable by the entire system, irrespective of any further processing, is 82% (410/500).

#### 3.2 Information Retrieval

In this experiment the Okapi information retrieval system was used to find relevant documents in the AQUAINT collection<sup>5</sup> as this was the collection used in the TREC 2002 question answering evaluation.

Unfortunately Okapi achieved a coverage (Roberts and Gaizauskas, 2003) of only 62.4% over the top twenty documents (i.e. at least one answer bearing document was found for only 256 of the 410 questions correctly typed in the previous stage). This information means that the maximum achievable score for the entire system, irrespective of further processing, is now limited to 51.2% (256/500).

#### 3.3 Locating Possible Answers

Running the final processing stage over the entire five hundred questions shows that 25.6% (128/500) of the questions were correctly answered. If we had considered using the 200 most relevant documents then although Okapi would achieve a coverage of 74.6% the overall score drops to 23.6% (118/500) most probably due to an explosion in the number of entities of the correct type being considered as possible answers. These results are not

particularly impressive especially when compared with the best performing systems which can answer approximately 85% of the same five hundred questions (Moldovan et al., 2002).

Users of Internet search engines are, however, used to looking at more than one possibly relevant document and hence are likely to be willing to look at a handful of short answers, especially if supporting snippets of text are provided. AnswerFinder defaults to returning the top five answers for each question, so evaluating over these answers shows that 35.8% (179/500) of the questions can be correctly answered, which is 69.9% (179/256) of the maximum attainable score.

### 4 AnswerFinder

#### 4.1 User Interface

The AnswerFinder application was designed to make it as easy as possible for an average computer user, familiar with web browsers, to make use of the question answering technology detailed in this paper.

The application uses a multi-document interface to allow the user to ask numerous questions at the same time and each window consists of an area for the user to type their question and an area in which to display any possible answers.

Examples of the application correctly answering the questions “*What is the August birthstone?*” and “*When was Gustav Holst born?*” are given in Figure 1. As you can see not only are the possible answers displayed as links to the supporting documents but a snippet of text is also given showing the answer in context. A limited confidence level (the percentage of all possible answer entities which fall within the answer group) for each answer is also given.

#### 4.2 Performance

On average it takes less than two seconds to process the ten most relevant documents and to display the answers<sup>6</sup>. This is in addition to the time taken to download the relevant documents which is not a

4. Another benefit to using these questions is that an automatic evaluation is possible using the Perl patterns available from NIST.

5. Previous experimentation with Okapi suggested that retrieving the twenty most relevant passages of at most one paragraph in length (referred to as documents in the remainder of this paper) would provide us with a reasonable amount of answer bearing text.

6. Using a 2.4Ghz Intel P4 with 512Mb of memory.

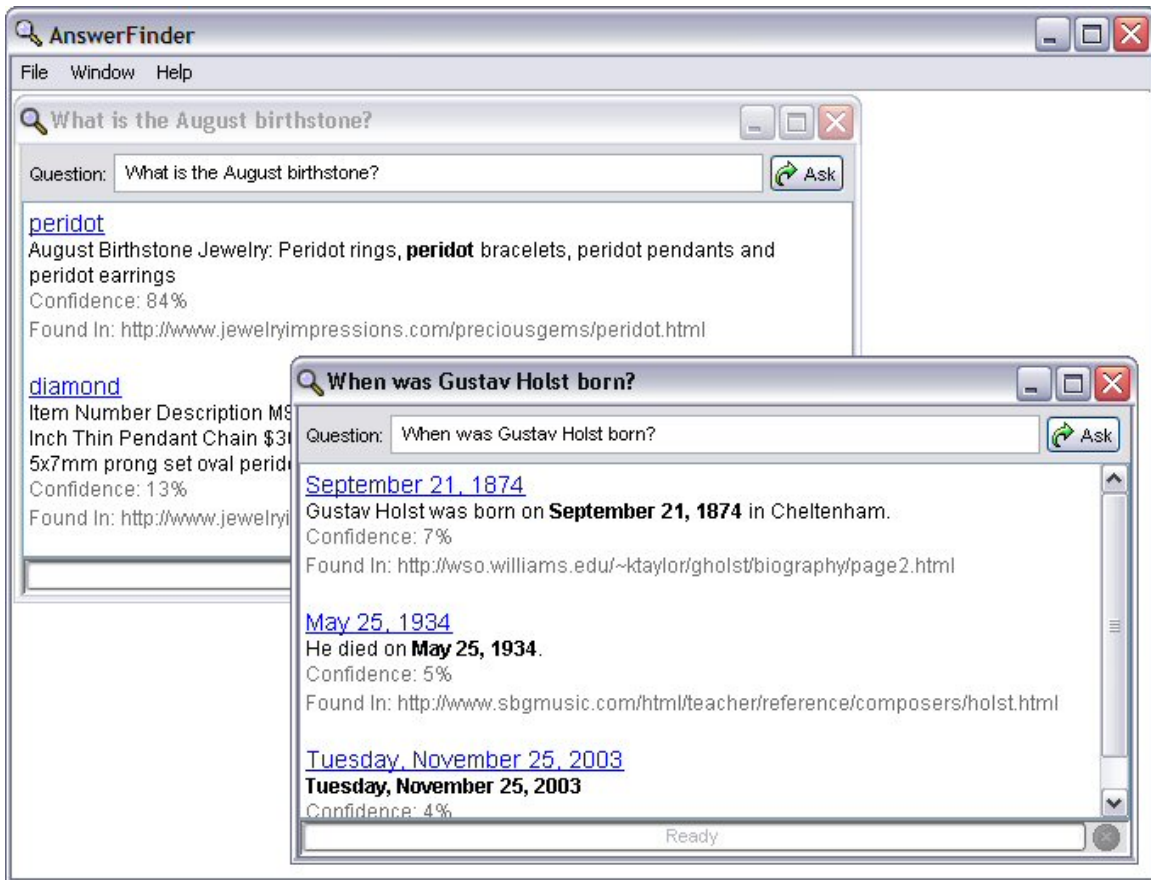


Figure 1: AnswerFinder showing successfully answered questions.

concern as the user would have to manually download the documents if they were using a search engine to find the answers. AnswerFinder can also be configured to use only the snippets returned by Google, alleviating the delay caused by downloading the full documents.

Clearly the percentage of questions correctly answered will be influenced by the number of relevant documents Google is able to locate; if the coverage is less than that achieved by Okapi (see Section 3.2) then the percentage of questions correctly answered will be reduced. Although if Google achieves better coverage than Okapi, the percentage of correctly answered questions should increase.

## 5 Related Work

The idea of building an easily accessible question answering system which uses the web as a document collection is not new. Unfortunately it is difficult to determine the first system of this kind due mainly to the fact that the authors of many systems claim to have been the first to develop and make

public such a system. All of these systems seem to be accessed via a web browser and unlike AnswerFinder involve no client-side software<sup>7</sup>. Their aim, however, is the same – to go beyond standard document retrieval. In the remainder of this section we will compare a number of these systems to AnswerFinder.

The consistently best performing system at TREC (Moldovan et al., 1999; Harabagiu et al., 2000; Harabagiu et al., 2001; Moldovan et al., 2002) forms the backbone of the PowerAnswer system from Language Computer<sup>8</sup>. From a users point of view the system is similar to AnswerFinder in that the full question is given to the system and then answers are displayed. The difference is that the answers are very much what you would expect from a search engine in that each answer is a sentence and no attempt is made to cluster (or remove) sentences which contain the same answer. This means

7. Requiring client-side software is not an issue if the popularity of Google's toolbar and deskbar are any indication.

8. <http://www.languagecomputer.com/demos/>

that the user still has to read the sentences to locate the answer to their question. This is strange given that fact that at TREC the underlying technology has been shown to be highly accurate (approximately 85%) even when returning only a single exact answer (Moldovan et al., 2002).

A system called AnswerBus<sup>9</sup> (Zheng, 2002) behaves in much the same way as PowerAnswer; returning full sentences containing duplicated answers. The reason for mentioning it here is that the questions can be given to the system in either English, French, Spanish, German, Italian or Portuguese with the system automatically determining the language, although answers are only given in English. The performance of the system is claimed to be 70.5% over the TREC 8 question set although we believe the performance would decrease if exact answers were being evaluated as experience of the TREC evaluations has shown this to be a harder task than simply finding answer bearing sentences.

Much closer to AnswerFinder is a system called NSIR<sup>10</sup> from the University of Michigan. NSIR uses a standard search engine to locate relevant documents, just as AnswerFinder does, and returns ranked exact answer. Unfortunately no context is provided along with the answers so a user still has to read the original document to verify that a given answer is correct. The system was entered into the TREC 2002 evaluation (Qi et al., 2002) and correctly answered 24.2% of the questions (this includes those marked as inexact or not supported) which is similar to the 25.6% obtained by AnswerFinder over the same test set.

The system most comparable with AnswerFinder, from a user's perspective, is IONAUT<sup>11</sup> (Abney et al., 2000). IONAUT uses its own crawler to index the web with specific focus on entities and the relationships between them in order to provide a richer base for answering questions than the unstructured documents returned by standard search engines. The system returns both exact answers and snippets. Unfortunately the exact answers are not tied to a specific snippet, so it is not immediately clear which snippet supports which answer. This problem is compounded by the fact that multiple snippets may support a single answer as no attempt has been made to cluster/remove snippets supporting the same answer.

All the web based question answering systems we were able to evaluate go some way to replacing standard search engines, however, all were in some

way deficient in the way they presented answers to the user – although the actual performance of some of the systems (notably PowerAnswer) far outstrips that of AnswerFinder over the TREC test sets.

## 6 Conclusions

The original aim in developing the question answering system detailed in this paper was to determine how well a system, using only a fine grained system of answer typing, could perform over TREC style questions. We have shown that this system can correctly answer approximately 26% of the TREC 11 test. For a baseline system this is quite respectable given that the average performance by participants in TREC 11 was approximately 22%.

It should be clear, however, that there is room for improvement. Even though the system is capable of answering approximately 26% of simple factoid questions there are clearly many questions which the system cannot currently answer. Any future work on the system could include:

- Improving the question typing stage through the addition of new rules or replacing the rules with a classifier acquired automatically from example question-answer type pairs (see Li and Roth (2002) and Zhang and Lee (2003) for possible approaches to this problem).
- The existing entity detection could be improved mainly through the addition of new gazetteer lists to cover other commonly occurring answer types.
- The system could be extended to cover questions whose answers are not named entities possibly by incorporate surface matching text patterns (Greenwood and Gaizauskas, 2003). This may also benefit questions which can already be answered by the method outlined in this paper.

Hopefully these, and other, improvements will lead to a better question answering system that can still be used by everyday Internet users.

### Obtaining AnswerFinder

The AnswerFinder application discussed in this paper can be freely downloaded from <http://www.dcs.shef.ac.uk/~mark/phd/software/>.

9. <http://misshoover.si.umich.edu/zzheng/qa-new/>

10. <http://tangra.si.umich.edu/clair/NSIR/NSIR.cgi>

11. <http://www.ionaut.com:8400>

## Acknowledgements

My thanks to Professor Robert Gaizauskas and Dr Horacio Saggion for their comments and feedback during the preparation of this paper.

## References

- Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer Extraction. In *Proceedings of ANLP 2000*.
- Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*.
- Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.
- Mark A. Greenwood and Robert Gaizauskas. 2003. Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In *Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03)*, pages 29–34, Budapest, Hungary, April 14.
- Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2000. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text REtrieval Conference*.
- Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, Paul Morărescu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's Question-Answering Server. In *Proceedings of the 10th Text REtrieval Conference*.
- Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question Answering in Webclopedia. In *Proceedings of the 9th Text REtrieval Conference*.
- Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*.
- Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju, and Vasile Rus. 1999. LASSO - A Tool for Surfing the Answer Net. In *Proceedings of the 8th Text REtrieval Conference*.
- Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. 2002. LCC Tools for Question Answering. In *Proceedings of the 11th Text REtrieval Conference*.
- Hong Qi, Jahna Otterbacher, Adam Winkel, and Dragomir R. Radev. 2002. The University of Michigan at TREC2002: Question Answering and Novelty Tracks. In *Proceedings of the 11th Text REtrieval Conference*.
- Ian Roberts and Robert Gaizauskas. 2003. Evaluating passage retrieval approaches for question answering. Research Memorandum CS-03-06, Department of Computer Science, University of Sheffield.
- Robert F. Simmons. 1965. Answering English Questions by Computer: A Survey. *Communications of the ACM*, 8(1):53–70.
- Ellen M. Voorhees. 1999. The TREC 8 Question Answering Track Report. In *Proceedings of the 8th Text REtrieval Conference*.
- Ellen M. Voorhees. 2000. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text REtrieval Conference*.
- Ellen M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text REtrieval Conference*.
- Ellen M. Voorhees. 2002. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text REtrieval Conference*.
- Dell Zhang and Wee Sun Lee. 2003. Question Classification using Support Vector Machines. In *Proceedings of the 26th ACM International Conference on Research and Development in Information Retrieval (SIGIR'03)*, Toronto, Canada.
- Zhiping Zheng. 2002. AnswerBus Question Answering System. In *Human Language Technology Conference (HLT 2002)*, San Diego, CA, March 24-27.