Open-domain Surface-Based Question Answering System

Aaron Galea

Department of Computer Science and AI,
University of Malta

Abstract. This paper considers a surface-based question answering system for an open-domain solution. It analyzes the current progress that has been done in this area so far, while as well describes a methodology of answering questions by using information retrieved from very large collection of text. The solution proposed is based on indexing techniques and surface-based natural language processing that identify paragraphs from which an answer can be extracted. Although this approach would not solve all the problems associated with this task the objective is to provide a solution that is feasible, achieves reasonable accuracy and can return an answer in an acceptable time limit. Various techniques are discussed including question analysis, question reformulation, term extraction, answer extraction and other methods for answer pinpointing. Besides this further research in question answering is identified, especially in the area of handling answers that require reasoning.

1 Introduction

Information Retrieval (IR) techniques were developed with the intention of helping users to find the information required. The search engines developed for this purpose were adequate for some time but they are now finding their limitations. Usually search engines return a huge list of relevant documents for a user query, which still requires the user to skim through the documents to find the necessary information. The invention of Question answering (QA) systems is to avoid this user overhead and present them with the direct answer to the question. For example if a user asks the question "Who is the president of Malta?" as an answer he/she gets the name of the Maltese president rather than a whole range of documents describing everything that might be related to the question. In this paper an attempt is made to achieve a QA system that can answer users' questions with relatively good accuracy and provide the ability to answer factual questions supplied by the users.

This paper is structured as follows. In section 2 the previous work done in this area is discussed. After this an overview of the approach followed for our system is presented in section 3. Section 4 will then describe some modules making up the system. Section 5 presents the evaluation results of the system compared with others. Finally Section 6 concludes the paper and gives future enhancements.

2 Approaches

Research in question answering systems is not something innovative since interest in them started from the era of knowledge bases. However they were always employed in closed domains [1, 2]. The interests have now shifted into scaling these QA systems to an open-domain solution and enhance search engines to provide this granularity of detail to users. Achieving this is still not widely available even though some attempts have been made like Google QA[1] and Ask Jeeves[2].

[1] https://answers.google.com/answers/main

[2] http:/www.ask.com

Open-domain question answering is being attempted either by using deeper-level or surface-based approaches. Deeper-level approaches offer the ability to handle more complicated questions but at the same time has limitations since a knowledge base has to be constructed. Knowledge in this kind of systems is represented either in KIF or CGIF format [3] (pg 427). One problem with this kind of QA systems is that an ontology has to be built so that it can reason about the world being modeled [4, 3]. An attempt to build an ontology for these situations is being handled by Cyc[3] while another research makes use of a linguistic resource to include background knowledge [5]. Despite these efforts such an approach is still unfeasible to implement for open-domain solutions. This is the same idea of the semantic web[4] but in the case of QA systems it is impossible to create an ontology for every document used.

The other approach that offers more chances of success is a surface-based approach. One problem with such an approach is that they can handle only factual question that appear directly in text. When asking questions that require reasoning a deeper-level approach is the only alternative. In fact most of the QA systems competing in TREC applied this approach [6–9] and offers a number of advantages. First of all performance can be measured by using the TREC evaluation method [10]. These systems as well are based on current state-of-the-art technology like parsers [11–13], entity name recognizers [14–16] and search engines [17, 18]. However such QA systems can be extended to handle complex questions. In LASSO [4], a surface-based QA system was extended to incorporate a theorem-prover to verify whether a piece of text is an answer to a question.

Considering the benefits of a surface-based approach to question answering and the extensionality it provides, in this paper a QA system for educational purposes is presented based on this technique. The motivation behind this work is to build a QA system to help students with their difficulties in particular subjects. Apart from this it also shows how an FAQ module that detects semantically equivalent questions can improve system accuracy. We employ the Winnow algorithm [19] for question classification rather than the traditional rule-based approach followed in other QA systems [6–9]. One major difference in this QA system is that it assumes that a question could have been previously met in a variant form while other QA systems assumed that all previously submitted questions were never seen.

3 Overview Of Approach

Figure 1 represents the approach followed in our QA system. Initially a user submits a question, which gets processed by the FAQ module. It tries to identify semantically equivalent questions that have been asked before. If it succeeds to find a semantically equivalent question its answer is returned. On the other hand if it fails, the questions is forwarded to the Question identification module whose purpose is to classify the question. For example, if the question asked is “Who is Vidocq?” then the question should be classified as a Person. This is useful during later stages when entity extraction rules are run on the text segments retrieved. The user question is further processed by a Term Extraction module. Its objective is to extract useful terms to search in the indexed documents. The Terms extracted together with the classified question are passed to an Answer Extraction module. Its purpose is to identify a text segment that might contain an answer to a question. It uses the terms to identify a number of documents that might contain the answer. These documents are further processed to extract sentences that can form potential answers. Finally entity extraction rules are run on the text segments retrieved to better score the sentences that can form an answer. Since the system supports an FAQ module that is constantly being updated the system also supports feedback from the user. This allows a list of questions to be accumulated by

[3] `http://www.cyc.com/products2.html`

[4] `http://www.w3.org/2001/sw/`

the FAQ module so that future users asking similar questions would get the answer quicker rather than taking the time consuming task of extracting answers from documents.

In the next section these modules are described in more detail.

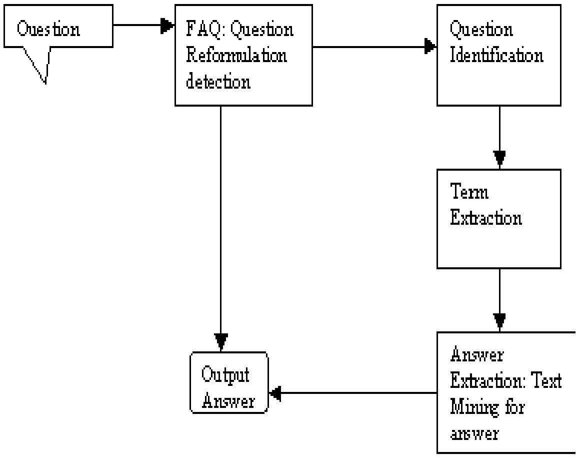
Question
FAQ: Question Reformulation detection
Question Identification
Term Extraction
Answer Extraction: Text Mining for answer
Output Answer

Figure 1: Answer Extraction process

4 Architecture

The system architecture presented in figure 2 represents the modules mentioned in the previous section together with other tools being used by the system. These modules are described in the following subsections.

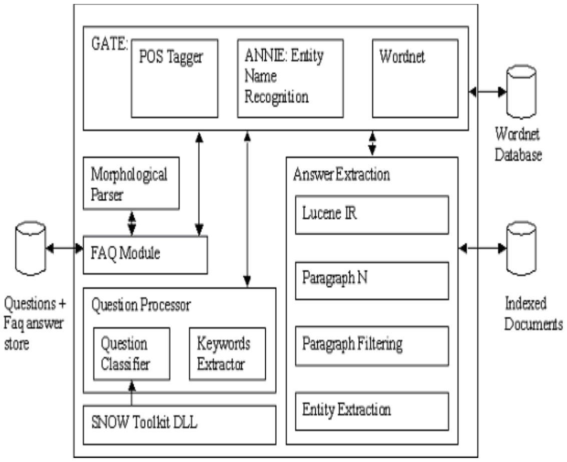

Figure 2: System Architecture

4.1 FAQ (Frequently-Asked question) modules.

The FAQ module purpose is to identify semantically equivalent questions. The algorithm applied is borrowed from [20]. Its attractiveness is that it does not depend on well-formed questions. This is something important for our QA system since we cannot assume that the questions supplied are grammatically correct. This FAQ module tries to improve the system performance without needing to use expensive operations like answer extraction. If the FAQ module fails the rest of the modules come into play to extract an answer from the indexed documents. The first module to start the whole extraction process is the Question Processor (QE) module

4.2 QE (Question Processor) module

The QE processor purpose is to classify the question submitted and as well extract key terms from the question. Question classification employs a trained Winnow algorithm using the SNoW toolkit [21]. The term extraction part uses the question to extract terms that are either quoted terms, noun terms and does not belong to a stop word list. Besides this feature it also expands certain terms based on word synonyms. Once this information has been extracted it is passed to the Answer Extraction (AE) module.

4.3 AE (Answer Extraction Module)

The Answer Extraction module uses the terms extracted to search for documents that contain the terms. It uses an approach similar to the Lasso question answering system [22]. In order to reduce the documents returned a Paragraph N operation is applied that checks whether the terms appear in N consecutive paragraphs. Once this has been completed, paragraph filtering selects segments from the paragraphs that may contain the answer. The final stage is to score the segments retrieved based on the key terms present, the order of the key terms and entities present in the text.

5 Evaluation

We evaluated the question answering system in two stages. For the FAQ module we used the TREC-9 data that is available. However evaluating the AE module was performed on our own collected data since the TREC evaluation data was not available. This data consisted of comprehensions text together with their questions.

A total of 124 questions were used for evaluating the FAQ module purposes. Out of this 124 questions, 60 were indexed by the system and the rest are used to measure the precision and recall of the system. From the 64 questions to be answered, the FAQ module managed to answer correctly 48 questions, 15 had incorrect answer and only failed once to provide any answer. This gave us a precision and recall of 75% and 76%. The reason why it failed to correctly answer all questions was the problem of transitivity. As an example taking the following indexed questions, which are a reformulation of each other, could fail to identify the similar question “What could I see in Reims?”:

What tourist attractions are there in Reims?

What are the names of tourist attractions in Reims?

What attracts toursits in Reims?

Since the word “see” and “attractions” have no relation in wordnet this question reformulation identification fails. However if the question “What is worth seeing in Reims?” is added to the indexed questions it will then succeed because of the keyword “seeing” which has a root form of “see” (see+ing). This module will get more accurate as more questions are added to the index. Since no other QA systems have employed this technique comparing it with others was quite impossible.

The AE module as well scored quite a high precision and recall of 87.5% and 63.64% respectively. Evaluating this module involved indexing six comprehension texts and the use of 22 questions. From this questions 14 were answered correctly and 2 were incomplete answers. An answer was considered correct if the four top most results returned contain the answer. Analyzing these results the system failed to answer some questions because either it failed to identify key terms in text or otherwise the answer returned was incomplete. In other words, an incomplete answer is a text segment with a 250 bytes limit but where the answer to the question occurs outside the answer window. In our opinion the system competes well with the other QA systems appearing in the TREC-9. However since the TREC data was not available for us comparing its performance with other QA systems was not possible.

One major drawback of the system is the system response time. Answering questions that need to score several text segments from a huge number of documents is a lengthy process. Various attempts to speed things up were tried like changing from a DOM parser to a SAX parser, moving the entity name recognition during document indexing and spawning several threads working independently on each document. The latter effort has not been measurable on a single processor machine, though on a better hardware the system will surely perform better. Despite these efforts the POS (Part-Of-Speech) tagger and its dependency on XML data was seen as the major drawback for performance. Considering of rewriting certain modules to remove the need of XML data at this stage to achieve the same results was not seen as feasible. However these changes are outlined in the future enhancement section below.

6 Future Enhancements

The system can answer factual questions with a high degree of accuracy. Increasing this accuracy involves a better method for term expansion. Although this is implemented in the system by using Wordnet synonyms the expansion method was not seen very useful. A very effective method is to employ a Word Sense Disambiguity (WSD) method for the words so that it guides the expansion of terms and thus resulting in a better chance to retrieve interesting documents. Another effort is to remove the XML data for improving the system performance. One way to achieve this is either swapping the Lucene search engine with another that can perform indexing at passage/paragraph level or else modify the Lucene indexing method.

Other improvements that might be considered in the future is to go beyond answering factual questions and cater for more complex questions. This can involve adding a new module to the system that can handle reason questions by making use of a knowledge base and theorem provers. One drawback of this solution is the need of a domain dependent solution, which contradicts our main aim of an open-domain solution. Therefore research in this area on how to achieve an open-domain solution while using a knowledge base can be studied further and included with this system.

References

1. Terry Winograd, A Procedural Model of Language Understanding, Computer Models of Thought and Language, R. Schank and K. Colby, eds., 152-186, New York: W.H. Freeman, 1973.
2. W. A. Woods, "Semantics and Quantification in Natural Language Question Answering," Advances in Computers, vol. 17, M. Yovits, ed. 2-64, Academic Press, 1978.
3. John. F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Fundations, Brooks/Cole Thomson Learning, 2000.
4. Sanda M.Harabagiu, Marius A. Pasca and Steven J. Maiorano, Experiments with Open-Domain Textual Question Answering, In Proceedings of COLING-2000, Saarbruken Germany, August 2000.
5. Sanda M. Harabagiu and Dan I. Moldovan, A parallel System for Text Inference Using Marker Propogations, IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 8, August 1998.
6. Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk and Chin-Yew Lin, Question Answering in Webclopedia, University of South Carolina.
7. Cody Kwok, Oren Etzioni and Daniel S. Weld, Scaling Question Answering to the Web, 2002.
8. Christof Monz and Maarten de Rijke, Tequesta: The University of Amsterdam's Textual Question Answering System, 2001.
9. Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu and Armardeep Grewal, Probabilistic Question Answering on the Web, WWW Conference, 2002.
10. Ellen M. Voorhees and Dawn M. Tice, Building a Question Answering Test Collection, National Institute of Standards and Technology.
11. U. Hermjakob and R.J. Mooney, Learning Parse and Translation Decisions from Examples with Rich Context, In Proceeding of the 35^{th} Annual Meeting of the Association From Computational Linguistics and 8^{th} Conference of the European Chapter of the Association of Computational Linguistics, pages 482-489, 1997
12. Adwait Ratnaparkhi, Learning to Parse Natural Language with Maximum Entropy Models, Kluwer Academic Publishers, Boston
13. Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu, and Marin Dimitrov. Developing language processing components with GATE (a user guide). Technical report, Department of Computer Science, University of Sheffield, 2001.
14. Claire Grover, Colin Matheson, Andrei Mikheev and Marc Moens, LT TTT – A Flexible Tokenisation Tool, Language Technology Group, University of Edinburgh, 1999.
15. Claire Grover, Colin Matheson, Andrei Mikheev and Marc Moens, Description of the ltg system used for named entity recognition. Technical report, HCRC Language Technology Group, University of Edinburgh, 1998.
16. Daniel M. Bikel, Richard Schwartz and Ralph M. Weischedel, An Algorithm that learns What's in Name, BBN Systems & Tecnologies, 1999.
17. Baeza-Yates and Ribero-Neto, Modern Information Retrieval, Addison Wesley, 1999
18. Ian H. Witten, Alistair Moffat and Timothy C. Bell, Managing Gigabytes, Morgan Kaufmann, 1999
19. Xin Li and Dan Roth, Learning Question Classifiers, Coling 2002.
20. Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Griju, Vaile Rus and Paul Morarescu, FALCON: Boosting Knowledge for Answer Engines, Southern Methodist University, 2000.
21. Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen and Dan Roth, Snow User Guide, Urbana, Illinois, August, 1999.
22. Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Milhalcea, Richard Goodrum, Roxana Girju and Vasile Rus, LASSO: A Tool for Surfing the Answer Net, 1999.