AnswerFinder at QA@CLEF 2007

Menno van Zaanen, Diego Mollá
Macquarie University
{menno,diego}@ics.mq.edu.au

Abstract

In this article, we describe our experiences with modifying and applying AnswerFinder, a generic question answering system that was originally designed to perform text-based question answering in English only, to multi-lingual question answering. In particular, we participated in the Dutch-English task. To enable handling of Dutch questions, we added a machine translation phase, where we used Systran to provide the Dutch to English translations. The translated questions were passed verbatim to AnswerFinder. Additionally, a simple form of anaphora resolution was implemented and since AnswerFinder did not have a document retrieval phase, this was added as well. The document retrieval system used is based on Xapian and simply searches for relevant documents based on the words in the question. Due to the nature and quality of the translated questions and the lack of tuning, we expected (and received) low quality results. The main purpose of our participation was to investigate the flexibility, portability, and scalability of the generic question answering system.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; I.2.7 [**Natural Language Processing**]: Text Analysis

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, Multi-lingual retrieval

1 Introduction

Within the AnswerFinder project, the main research question is: what shallow semantic representations are best suitable for representing the meaning of questions and sentences in the context of question answering? To experiment with this, over the last several years we have developed the AnswerFinder question answering system.

The AnswerFinder system participated in TREC competitions during the last several years [3, 5, 6, 11]. In this context, English has been the language of the questions and documents. The document collection that has been used is the Aquaint corpus, which consists of 1,033,461 news articles.

When redesigning AnswerFinder in 2005, in addition to performance requirements (speed, memory usage, accuracy of finding answers, etc.), two requirements were recognised that have had

a big impact on the design and implementation of the system. These requirements stem from the fact that AnswerFinder is a research and development system.

Flexibility The system should be flexible in several ways. It should be possible to easily modify AnswerFinder to handle new situations. For example, different input and output formats (of documents, questions, and answers), types of questions and answers, and new algorithms (in all the phases) should be easy to integrate in the system. This allows for easy testing of new ideas in different environments.

Configurability Having a system with many different algorithms that can be used in the phases, it should be easy to configure the system to run using specific parameters. Parameters in this case mean, not only the actual values needed in the algorithms, but also selecting a particular algorithm in a phase.

Applying AnswerFinder to a multilingual question answering problem will show in how far we have been successful in reaching the requirements as described above. Our participation in QA@CLEF is a pilot project where we concentrate mainly on the flexibility of the system. We needed to modify the system to work in a multilingual environment and the document collection is larger and contains different kinds of documents. Furthermore, we did not have enough time to fine-tune the system to this particular task.

In the next sections, we will first discuss the general AnswerFinder system, followed by a description of the original plans and the actual changes we made to the system to allow it to be used in the QA@CLEF competition. Then we discuss the results and briefly say something about the improvements we expect to make.

2 AnswerFinder

The AnswerFinder question answering system is essentially a framework, consisting of several phases that work in a sequential manner. For each of the phases, a specific algorithm has to be selected to create a particular instantiation of the framework. The aim of each of the phases is to reduce the amount of data the system has to handle from then on. This allows later phases to perform computationally more expensive operations on the remaining data.

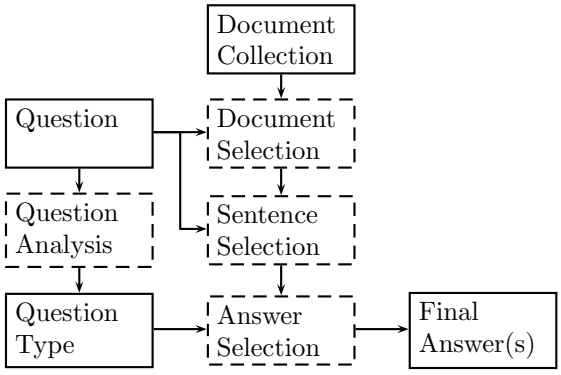

Figure 1: AnswerFinder system overview

Figure 1 provides an overview of the AnswerFinder framework. The first phase is a document retrieval phase that selects relevant documents. AnswerFinder was developed to work on large document collections and this phase typically reduces a great amount of text that will be handled in subsequent steps. In the TREC competitions, a list of relevant documents was always provided and AnswerFinder used these documents. We have implemented a new document retrieval phase specifically for QA@CLEF, which is described in more detail in section 3.3.

Next is the sentence selection phase. It is possible to select a sequence of algorithms that are applied to the set of sentences that remain. Each algorithm selects a subset of sentences returned by the previous algorithm (or taken from the relevant documents selected by the document retrieval phase). During sentence selection, all sentences that are still left (e.g. all sentences in the selected documents in the first step) are scored against the question using a relevance metric. The most relevant sentences according to this metric are kept for further processing.

After the sentence selection phase, the remaining text is analysed during the answer selection phase. This phase searches for possible answers in the sentences. In the question analysis phase, the question is analysed, which provides information on the kind of answer that is required. This question information is then matched against the list of possible answers and those answers that match the type of answer required by the question are selected and scored.

Finally, the best answer is returned to the user. The best answer is the answer that has both the highest score and an answer type that matches that of the question, or simply the answer with the highest score if none of the possible answers fits the expected answer type.

3 Extensions

The AnswerFinder system so far has been applied in the TREC question answering competitions. With the design requirements in mind, we have looked at alternative situations where the system could be applied. This year, we participated in QA@CLEF to investigate flexibility in languages and scalability (as we will describe later) and QAst to investigate flexibility and robustness of the system.

Firstly, we will describe the original ideas we had when we decided to work on multilingual question answering. Unfortunately, we did not have time to implement many of the ideas raised during these preliminary discussions. Next, we will describe extensions that were required for the system to work in the CLEF multilingual environment.

3.1 Original idea

The idea behind applying AnswerFinder to multilingual question answering arose from discussions with researchers from the University of Groningen in the Netherlands, who are working on the Joost question answering system for Dutch [2, 1]. The underlying ideas of Joost are quite similar to those of AnswerFinder. They have a similar internal representation of the meaning of the questions and the texts. The assumption here is that this internal representation can be used to link parts of the two systems together, which allows the analysis of questions in one language and the analysis of and search in documents in another language.

In a way, both Joost and AnswerFinder are quite dissimilar, for example, they work on different languages, use different knowledge sources and depend on different tools. However, the most important aspect of the systems is very similar. Both systems use a similar shallow internal meaning representation of the questions and the texts.

The meaning representation can be used as a pivot between the two systems. If a mapping between the representations is made, it is possible to transfer the internal knowledge of one system to the other and continue from there. Effectively, this means that the analysis of the question into the internal meaning representation can be done using one system. This representation is then mapped to the other's representation (which is relatively straightforward as the representations are quite similar) and the second system can be used to find the answer.

This idea led to the identification of four distinct phases, each more complex and intertwining both systems more.

1. The least complicated solution is to translate questions from one language to the other (for example, using the Systran machine translation system) and to feed the translated question in the correct question answering system. This system does not benefit from question analysis of the question in the original language and as such relies entirely on possibly incorrectly translated questions.

System	2004	2006
Freetranslation	0.383712	0.369556
Babelfish	0.405939	0.389921
Systran	0.416436	0.398048
DEMOCRAT	0.407023	0.390922

Table 1: Blue score results of translation of CLEF questions Dutch to English

2. To benefit from language specific analysis, the result of the question analysis of the question answering system of the source language can be handed to the question answering system in the target language in addition to the translation of the question given by the machine translation system. This approach requires a mapping between the output of the question analysis module of the source language to the required input of other modules of the target language.

3. The most challenging approach would be one that formalises and implements the mapping between the two internal representations that underlie all modules of both question answering systems. The benefits of this approach are that all analyses of the questions and the texts are done using tools that are developed for that particular language.

In the end, only the first approach has been implemented here. There is no integration of both systems, the questions are translated using a machine translation system and used in the AnswerFinder system. In addition to this machine translation step, AnswerFinder had to be modified in two aspects.

Originally, AnswerFinder did not have a document retrieval phase, as in the TREC competitions a list of relevant documents was always provided. During TREC 2006, we briefly experimented with our own document retrieval phase, but this required modifications to allow it to work in this context.

Additionally, AnswerFinder has a simple anaphora resolution phase. This phase is designed specifically for the TREC competition and had to be rewritten for the QA@CLEF competition.

3.2 Machine translation

AnswerFinder expects to find documents and questions in English. Several algorithms that are currently implemented in AnswerFinder require the text to be in English as both sentences and questions are parsed using a dependency parser. In the Dutch–English task, the documents are in English already, but the questions are in Dutch.

There are several on-line machine translation systems that translate from Dutch to English. We concentrated on Freetranslation[1], Babelfish[2], and Systran[3].

To decide which of these systems would provide us with the best translation of the questions, we translated QA@CLEF questions of the 2004 and 2006 competitions. For these years, we were able to find aligned Dutch and English questions. This allowed us to evaluate the quality of the questions against a gold standard. The results of the machine translation systems can be found in Table 1. We used the Blue score [8] to compute these results[4]. The results show that Systran is best, followed closely by Babelfish. Babelfish's underlying machine translation system seems to be a slightly different version of the Systran engine.

To see whether we could further improve on the translation quality of these three systems, we applied the DEMOCRAT [12] consensus translation system to the translated questions. DEMOCRAT, which stands for DEcides between Multiple Outputs CReated by Automatic Translation, is designed specifically for free on-line MT systems. It tries to construct from them a consensus

[1] `http://www.freetranslation.com`
[2] `http://world.altavista.com/babelfish`
[3] `http://www.systranbox.com/systran/`
[4] We used the evaluation program provided by our colleague Simon Zwarts, to whom we are most grateful.

translation which, it is hoped, will take the best elements of the contributing systems and produce an output as good as or better than any of the individual MT systems on their own.

The main idea behind DEMOCRAT is to combine the output of machine translation systems. This is done by selecting and recombining the "best" sequence of words from each output. The underlying assumption here is that if several machine translation systems use the same word or phrase, it is the correct translation. Single words are selected based on a majority vote of the output of the systems.

Unfortunately, as can be seen in Table 1, the DEMOCRAT system does not provide any better results. Choosing DEMOCRAT's output is better than selecting one of the machine translation systems at random. However, the results on the old questions show that Systran has consistently generated the best translations. This means that it is better to use Systran to translate the questions, which we have done in these experiments.

When looking at the translated questions, it turned out that some questions resulted in translations that were completely incomprehensible, for example, "Call some olive oil see.", or nearly incomprehensible, "What was the name of its woman?". Also, we noticed some consistent translation errors. For example, all Dutch questions starting with "Noem..." are translated starting with "Call...", whereas the proper translation would be "Name". Also, the translated questions often have a preposition as the first word, for example, "To which countries provide the EU agriculture subsidies?".

Our question analysis phase (which assigns expected types of answers to a question) is based on matching regular expressions to the question. This is a very simple system that can probably be improved much. In particular it could be adapted to the kinds of mistranslations that occurred in the questions. However, due to time constraints we did not modify it. The translation problems mean that none of these regular expressions match. This, in turn, means that AnswerFinder did not assigned expected answer types to the questions and therefore was unable to filter out any possible answers. Answers can only be selected based on their scores.

3.3 Document retrieval

The AnswerFinder submission for TREC 2006 contained a first attempt at document retrieval. In TREC, a list of relevant documents is provided per topic (where just like in QA@CLEF, a topic contains several questions). We looked at performing document retrieval specific to each each question and only take those documents that can be found in both the list of relevant documents found by our document retrieval system and the list provided by NIST. We used the indexing and retrieval methods of the Xapian[5] toolkit for this.

The list of relevant documents is found by taking the words in the question as search keywords in the retrieval system. Note that in the current version, we do not remove any stop words, which may lead to query drift.

Even though the Xapian-based document retrieval system slightly improved results in the TREC competition setting, the system has only been evaluated in combination with the list of relevant documents provided by NIST. Furthermore, the document collection used in that context, Aquaint, is smaller than the collection used in QA@CLEF. The Wikipedia collection is not only much larger, it also contains structure that our document retrieval system could not handle. We decided to convert the Wikipedia format to a format that is quite similar to that of the Aquaint documents. By doing this we may have lost some useful data.

3.4 Anaphora resolution

Anaphora resolution is not something we have looking into very much. During our TREC participation we always replaced the anaphora with the name of the topic. (In the TREC question answering track, the topics all have a name attached to it.) In QA@CLEF, several questions in the question collection also contain anaphora. Examples are "they" in the question "In which year

[5] `http://www.xapian.org`

were they boasted?", and "he" in the question "Of which political party was he member?". The anaphora often refers to a named entity or an important part of a previous question.

We decided to use previous answers as replacements for the anaphora, as this was easiest to implement in the current version of AnswerFinder. However, this obviously leads to the following problem: incorrect answers to a question means that the next question (with anaphora) will also be incorrect.

4 Results

We submitted two runs generated by AnswerFinder. Both runs used questions translated using Systran. Anaphora in the questions were resolved during question answering as described above, all anaphora were replaced by the answer to the previous question. Document retrieval was performed using our Xapian-based document retrieval module. The best 100 documents were retained in each run. After document retrieval, the sentence selection phase was started. In both runs, the best 100 sentences were selected based on word overlap. This metric counts the number of words in the question that can also be found in the sentence under consideration. Words that can be found in a list of stop words are not counted towards this score. The remaining sentences are handed to a named entity recogniser and the found named entities are taken to be possible answers. The possible answer with the highest score is then returned as the exact answer.

To generate the named entities, we used AFNER [7], which is a named entity recogniser that is built within the AnswerFinder project. The idea behind AFNER is that a high recall in named entities is required for question answering. AFNER uses a combination of gazetteers, regular expressions, and general features in a maximum entropy classifier to assign tags to words. These tags indicate whether the word is the beginning of a named entity, inside a named entity or outside any named entities. The general MUC named entity types [9] are used: organisation, person, location, date, time, money, and percent. These types are used to select answers according to the expected answer type that is found during question analysis. However, as mentioned above, since the question analysis phase did not classify any of the questions, there was no selection on the bases of the expected answer type. All found entities were treated as possible answers.

The settings of both runs were quite similar with only one difference. The second run performed a graph-based sentence selection phase, which finds additional possible answers. The algorithm that is used in this phase is described in more detail in [4]. The question and the sentence is parsed using Connexor [10]. This results in a dependency parse, which is converted into a shallow semantic representation in graph form. The score that is assigned to the sentence (for the question) is related to the size of the overlap of the graphs of the question and sentence.

The score computed using the graph logical forms is actually not used in this experiment, but the algorithm also has the ability to find exact answers. During a training phase, where we have questions aligned to sentences with the exact answer annotated, the overlap between the question and sentence is computed. Next, a path from the overlap to the answer is found. The overlap including the path is stored as a graph rule. During the actual sentence selection phase, these graph rules are applied to the overlap of the question and sentence. This indicates sub-graphs in the sentence that are considered to be possible answers.

In general, the score of a possible answer is computed by summing the scores that are found for that answer. If an answer is found using the named entity recogniser, the score is the number of times the answer is found by the named entity recogniser (in all selected sentences). If the graph-logical form algorithm is used, the score computed using the algorithm is used (and this score is added to the scores of the named entities).

Unfortunately, neither runs of the multilingual version of AnswerFinder generated any correct results. All answers generated were numbers. There may be several reasons for this. Firstly, the named entity recogniser finds numbers using regular expressions. Most of the other named entity types require other features. AFNER has been trained on the BBN corpus[6] which is somewhat

[6]Ralph Weischedel and Ada Brunstein, 2005, BBN Pronoun Coreference and Entity Type Corpus Linguistic Data Consortium, Philadelphia

different from the document collection used here. Secondly, the question analysis did not return specific question types (as described above). This means that the most often found named entity will be selected, which will often be numbers.

5 Future improvements

The discussion of the extensions of the AnswerFinder system and the results already indicated several areas that need improvement. We will briefly discuss some of them here.

The translation of the questions in imperfect. Not only does this impact the question analysis phase (mentioned above), it obviously impacts the whole system. For example, named entities are sometimes translated or partially translated ("General engines" for "General Motors", and some words are translated incorrectly "woman" instead of "wife" for "vrouw", "bluntest" instead of "bumped" for "botste". Some regular words are not translated at all, such as "ongeluk" (accident), or "zat" (sat). Post editing of the questions may solve some of these problems, although other AnswerFinder phases will need to be adapted as well to allow for partially incorrect questions.

The quality of the document retrieval phase that is used at the moment is unknown. We have not investigated in how far the documents found by this system contain information on the topic of the question. We probably need to do a more in depth analysis of the system. For example, the documents are selected based on the original question before anaphora resolution. This means that important information (referenced to using the anaphora) is missing in the query in the document retrieval phase.

The question analysis phase is based on a relatively short list of regular expressions. This list has not been tested against the translated questions at all. Many of the translated questions are somewhat different from the questions the system handled during the TREC competitions. Modifying the regular expressions and adding several new ones that handle incorrectly translated questions will probably increase performance of this phase greatly.

The way anaphora are handled in the current system is very simple and most of the time incorrect. Instead of replacing the anaphora by previously returned (probably incorrect) answers, we need to analyse the previous question or questions in the same topic and based on the type of anaphora we need to select the part of the question that fits those anaphora best.

Finally, in section 3.1 we already discussed several extensions that we expect will improve the quality of the system. If the question analysis can be done in the source language, we do not have to modify the English question classifier to handle incorrectly translated questions. If it is possible to map the semantic representation of the two systems, we do not even have to use a machine translation system to translate the questions. We will still need to handle ambiguity in translating the concepts from Dutch to English, but the impact of syntactically incorrect questions will be reduced greatly.

6 Conclusion

Our submissions to the QA@CLEF competition this year, even though the results are particularly bad, shows much about certain aspects of AnswerFinder. We are particularly interested in the flexibility, portability, and scalability of the system. It turns out that AnswerFinder can easily be modified to work on a different domain and with a different language.

There are a few specific aspects of our participation in this competition that shows that much work is still needed. Firstly, the translation of the questions is clearly imperfect. Unfortunately, many algorithms in the rest of the question answering system require proper English questions. Similarly, our current way of handling anaphora means that many questions are incorrect. This has a major impact on the quality of the answers generated by the system.

The increase in size of the document collection can be handled by the document selection phase. However, so far we have not done much research into this phase. We applied a very simple word-based document retrieval system and the quality of the returned documents is unknown.

Overall, we think that our participation in this track has been successful. We have managed to identify several areas that need improvement. All of these improvements can easily be incorporated in the AnswerFinder framework. The framework itself functions effectively, although algorithms in several phases need improvements to generate satisfactory results.

References

[1] Gosse Bouma, Ismail Fahmi, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. The university of groningen at qa@clef 2006: Using syntactic knowledge for qa. In C. Peters, editor, *Working Notes for the CLEF 2006 Workshop; Alicante, Spain*, September 20–22 2006.

[2] Gosse Bouma, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. Question answering for dutch using dependency relations. In C. Peters, editor, *Working Notes for the CLEF 2005 Workshop; Vienna, Austria*, September 21–23 2005.

[3] Diego Mollá. Answerfinder in TREC 2003. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003); Gaithersburg:MD, USA*, number 500-255 in NIST Special Publication, pages 392–398. Department of Commerce, National Institute of Standards and Technology, November 18–21 2003.

[4] Diego Mollá. Learning of graph-based question answering rules. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing; New York:NY, USA*, pages 37–44, 2006.

[5] Diego Mollá and Mary Gardiner. Answerfinder at TREC 2004. In Ellen E. Voorhees and Lori P. Buckland, editors, *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004); Gaithersburg:MD, USA*, 2004.

[6] Diego Mollá and Menno van Zaanen. Answerfinder at TREC 2005. In *Proceedings of the Fourteenth Text Retrieval Conference (TREC 2005); Gaithersburg:MD, USA*, NIST Special Publication. Department of Commerce, National Institute of Standards and Technology, 2005. cd-rom.

[7] Diego Mollá, Menno van Zaanen, and Daniel Smith. Named entity recognition for question answering. In Lawrence Cavedon and Ingrid Zukerman, editors, *Proceedings of the 2006 Australasian Language Technology Workshop; Sydney, Australia*, pages 51–58, 2006.

[8] K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: A method for automatic evaluation of machine translation. In *40th Annual Meeting of the Association for Computational Linguistics; Philadelphia:PA, USA*, pages 311–318. Association for Computational Linguistics, July 2002.

[9] Beth M. Sundheim. Overview of results of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference MUC-6; Columbia:MD, USA*, San Francisco:CA, USA, 1995. Morgan Kaufmann Publishers.

[10] Pasi Tapanainen and Timo Järvinen. A non-projective dependency parser. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97); Washington:DC, USA*, pages 64–71. Association for Computational Linguistics, 1997.

[11] Menno van Zaanen, Diego Mollá, and Luiz Pizzato. Answerfinder at TREC 2006. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC 2006); Gaithersburg:MD, USA*, NIST Special Publication. Department of Commerce, National Institute of Standards and Technology, 2006. accepted.

[12] Menno van Zaanen and Harold Somers. DEMOCRAT: Deciding between multiple outputs created by automatic translation. In *The Tenth Machine Translation Summit, Proceedings of Conference; Phuket, Thailand*, pages 173–180, 2005.