Question Answering for Dutch: Simple does it

Arjen Hoekstra Djoerd Hiemstra Paul van der Vet Theo Huibers

Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217 7500AE Enschede, ahoekstr@cs.utwente.nl, hiemstra@cs.utwente.nl, p.e.vandervet@utwente.nl, t.w.c.huibers@utwente.nl

Abstract

When people pose questions in natural language to search for information on the web, the role of question answering (QA) systems becomes important. In this paper the QA-system simpleQA, capable of answering Dutch questions on which the answer is a person or a location, is described. The system's algorithm does not use a lot of complex NLP-techniques, but instead uses the magnitude of and redundancy on the World Wide Web to its advantage. The system has been evaluated on the DISEQuA corpus and performed quite well: MRR near 0.5. For further improvements it can easily be extended by adding more rewrite rules and applying more sophisticated filtering and tiling.

1. Introduction

Most information need on the Internet is satisfied by sending queries to web search engines. These queries normally aren't posed in natural language, but by providing some keywords. However, there is a part of the web search engine users that does use natural language as a query. Search engine log files show that 1% of ten million queries are posed as questions in natural language, for instance "Where does Sean Hannity live?" [6]. If this habit becomes more and more common the need for systems that can interpret questions posed in natural language becomes more important. Systems that can fulfil this task are called Question Answering (QA) systems.

Nowadays, the QA-systems to which much of the research is devoted, are easily becoming quite complex, as much Natural Language Processing (NLP) is involved. One can think of part-of-speech tagging, parsing, use of grammars and lexicons and much more. But how much can be achieved if not a lot of fancy NLP techniques are used, but a relatively simple algorithm instead? Dumais, Banko and Brill investigated this by developing an algorithm that does not use much NLP, but uses the gigantic size of the Web to their advantage [2]. Their system was able to answer English factoid questions formulated in natural language.

The main goal of this research is to examine whether their method is a feasible one, only this research will focus on Dutch questions instead of English. This study is limited to questions on which the answer type is a person or a location.

In the following section the method used to design and implement the simpleQA-system is described. Afterwards the results of an evaluation performed on the system are presented and discussed. Finally conclusions will be drawn and recommendations for future work will be made.

2. The simpleQA system

The simpleQA system was designed making use of the algorithm as described in [2]. This algorithm is composed of the following steps:

1. Question analysis
2. Query rewriting/weighting
3. Web retrieval
4. N-gram mining and filtering
5. N-gram tiling
6. Pick top N-gram

The system consists of five core components: Analyzer, Retriever, Extractor, Tiler and Selector. These components and their relation to the algorithm steps will be described in the next paragraphs. Some parts of the architecture of a QA-system from van Langen [4], personQA, were reused for the development of simpleQA.

2.1. Analyzer

The purpose of the analyzer is to convert a question in multiple queries which can be sent to a search engine. This is accomplished by analyzing what kind of question is posed and subsequently rewriting the question in multiple queries and weighting each of these queries.

Question analysis

Two types of questions can be posed: questions on which the answer is a person or questions on which the answer is a location. The system does not have to distinguish between these kinds of questions, because the answer type is the same (a named entity). But it has to recognize what kind of pattern the question contains. Patterns are recognized by making use of regular expressions, using only limited knowledge of natural language. Only question words and auxiliary verbs are recognized. Based on this information, positions of VP's and NP's in the sentence are determined. No grammar or part-of-speech tagging is used. The possible Dutch person question patterns that can be distinguished can be found in Table 1.One might think of other kinds of Dutch person questions (like "Hoe

luid(t|de) de ([Adj/Prefix]) naam van [NP]?") which are not covered by the rewrite rules of the system. The reason for this is that those are considered as old fashioned, they are not used much anymore in everyday conversations and hence it will not be likely they will be used when posing questions. However the system can handle these questions by making a simple "AND" query of the non-stop words. It just does not add any other queries.

The possible location question patterns that can be recognized can also be found in Table 1.The pattern "Waar vond [NP] plaats?" is an example of a more aimed pattern. It also fits in "Waar [VP]?" but by explicitly recognizing it, better queries can be made. In the future the same can be done for queries like "Waar bevindt [NP]?" or "Waar ligt [NP]?" to boost the system's performance. A second thing that could catch the eye is that some location patterns much resemble person patterns. "Welk(e) [NP] is [NP]?" and "([Prep]) welk(e) [NP] [VP]?" are even exactly the same and thus merged in the question analysis part. Also "Wat is [NP]?" is merged because it is almost the same as "Wie is [NP]?". As the question words are stripped of the question and both are three letter words the same query rewriting can be applied.

Table 1. Person and location question patterns

Person question patterns	**Location question patterns**
Wie is [NP]?	Wat is [NP]?
([Prep]) wie [VP]?	Welk(e) [NP] is [NP]?
Welk(e) [NP] is [NP]?	([Prep]) welk(e) [NP] [VP]?
([Prep]) welk(e) [NP] [VP]?	Waar vond [NP] plaats?
Hoe heet(te) [NP]?	Waar [VP]?
Wat is/was de ([Adj]/[Prefix]) naam van [NP]?	

Query rewriting and weighting

For each question one or more queries are constructed. Two other variables are added to the query: the direction in which the answer is expected and the weight of the query. Direction can be "LEFT", "RIGHT" or "ANYWHERE".

Direction is only added when it is absolutely certain the answer must be in that direction. For example when the question "Wie is de minister-president van Nederland?" is posed, the query ["de minister-president van Nederland is?"] is constructed. One would expect the answer to be to the right of this query. However, there can well be web pages which contain something like "Jan-Peter Balkende, de minister-president van Nederland is vanochtend aangekomen in…". When in this case the direction "RIGHT" is added the correct answer will not be found in this snippet. Hence instead of "RIGHT", "ANYWHERE" is applied to the query.

The weight component is an integer value between 1 and 5 and is determined by the likelihood of finding a correct answer with the query. For a question like "Wie was president van de Verenigde Staten tijdens de Tweede Wereldoorlog"

it is more likely to find a correct answer with ["was de president van de Verenigde Staten tijdens de Tweede Wereldoorlog"] than with [president AND Verenigde AND Staten AND tijdens AND Tweede AND Wereldoorlog]. Weights will be used later on to score possible answers. All query details units (query itself, direction and weight) constructed for a question are passed on to the retriever.

2.2. Retriever

The Retriever implements the algorithm's third step: **web retrieval**. The search engine used for the QA system is Google. With help of the Google API [3] a maximum of 1000 queries a day can be sent to Google, which is able to return the data found by the queries in different formats. The query received from the Analyzer is sent to Google. The web pages found are restricted to pages in Dutch and a maximum of forty pages is returned. This is less than the number of pages that yielded the best results for [2], which were 200 pages. But because the collection of Dutch web pages is significantly smaller than the English one, the number of pages should be less. Otherwise to many nonsense answers will be retrieved. For each web page a snippet, which contains all of the query words, is extracted. Only unique URL's are investigated, as each query is weighted, it does not matter whether a different query returns the same web page. High weighted queries are the first ones to be sent, so it is guaranteed the highest weight is applied to a web page. All unique URL's are sent, together with their weights and answer directions from the analyzing part, to the Extractor.

2.3. Extractor

The Extractor's goal is to **mine N-grams** from the snippets and to **filter these N-grams**. An N-gram is just a sequence of N words. It returns a set with possible answers mapped on their weights. Each snippet passed through by the retriever is examined by the Extractor. The snippet is searched for possible answers in the direction provided by the query details. Possible answers are then filtered. Named entity recognition is performed by matching regular expressions. When a duplicate answer is found, its score is added to the score of the existing answer. Finally one other filter is applied to each answer: when an answer is contained in the question it is considered as an unwanted answer and removed from the map. The candidate answers are now further examined by the Tiler.

2.4. Tiler

The Tiler performs the **N-gram tiling** part of the algorithm: answers with matching components are merged. To accomplish this, the Tiler runs through

the set of possible answers. When two different answers contain the same components they can be tiled. For instance, when possible answers are ‘Balkenende’ and ‘Jan Peter Balkenende’ these will be combined into one answer ‘Jan Peter Balkenende’. The actual tiling is only performed when the score of the answer which contains the other answer is higher than the score of the other answer itself. Otherwise there would be a high chance on bad tiling. The map of tiled answers is now passed on to the Selector.

2.5. Selector

The selector creates a ranked list of ten answers where the **best ranked can be picked as the actual answer** on the question. Answer formulation only consists of the answer itself. Parts of the question could be included (by distinguishing patterns again) in the answer formulation, but this is quite trivial and does not make answers on this kind of factoid questions more valuable.

3. Evaluation

In this section the evaluation of the simpleQA-system will be discussed. The first paragraph describes the used evaluation method, the second paragraph presents the results and in the final paragraph a brief discussion of the results will be given.

3.1. Method

To evaluate the developed system two question sets (a person set and a location set) were presented to the system. The questions originated from the DISEQuA corpus [5], which was developed for the Cross Language Evaluation Forum (CLEF). Because the answer type of each question was marked in this corpus, person and location questions could easily be extracted from it. Another advantage of using this corpus is that van Langen’s personQA-system [4] and Bouma’s system Joost [1] also used this one. Hence the systems can be compared well by using the same test corpus. Just like the evaluation of personQA, person answers are also considered correct when only the last name is correct. For instance, for the question “Wie is de Italiaanse minister van buitenlandse zaken?”, “Franco Frattini” and “Frattini” are both considered to be correct answers.

3.2. Results

The test results contained questions for which no answers were available in DISEQuA. These questions were removed leaving 87 person questions (from a total number of 90 in DISEQuA) and 72 location questions (from a total number of 85 in DISEQuA). The person questions are compared to the results of

personQA [4] and Bouma's system Joost [1]. The Joost system was evaluated more restrictive (only exact answers correct and only the top five answers were considered).

From Table 2 one can conclude the scores of van Langen's personQA system are improved by implementing our algorithm. The simpleQA system achieves an MRR of 0.4844 opposed to 0.2909 of personQA. When applying a paired sample t-test the difference between the MRR's proves to be significant at the .01 level. Also the percentage of answers found and the percentages first answers correct are higher than those of personQA. Compared to Joost, the MRR, the percentage of first and found answers are all slightly lower. Because further results of Joost were not available the significance of the MRR difference could not be determined.

The location question results are slightly lower than those of the person questions, but they are in the same range. For the location results no systems were found which performed a similar evaluation.

Table 2. Results on the CLEF person and location question set

	Person questions			**Location questions**
	simpleQA	**personQA**	**Joost**	**simpleQA**
N	87	87	87	72
MRR	0.4844	0.2909	0.5804	0.4350
Answers found	63.2 %	46.0 %	65.5 %	56.9 %
First answer correct	40.2 %	20.7 %	52.6 %	36.1 %

The Microsoft researchers [2], who used the same kind of algorithm as simpleQA, tested on an English TREC-9 question set and thus cannot be compared in a good way. It does, just like Joost, only consider the top five ranked answers and their test set contained more kinds of factoid questions than just location and person. It obtained an average MRR of 0.507 and found answers on 61% of the questions.

3.3. Discussion

In general the simpleQA system seems to perform quite well. It improved the score of personQA significantly and got a lot closer to scores of Joost. Of course it can still be improved. When analyzing the results some issues attract attention.

The first one is the number of "welk(e)" (which) – question that are answered incorrectly. This is 76 % of the total of person "welk(e)"-questions and 44% of the location ones. An explanation for this is that there are not many rewrite rules for this kind of questions, because one would need to apply more NLP to determine the parts of speech. For example, one would like to rewrite a question like "Welke Franse ex-minister werd tot een gevangenisstraf veroordeeld wegens corruptie?" into ["werd tot een gevangenisstraf veroordeeld

wegens corruptie" AND "Franse ex-minister"]. But to accomplish this one would have to know where the NP starts and ends. This would make things much more complicated. One could for instance determine where the verb is in the sentence to determine the end of the NP, but for this part-of-speech tagger would be needed. So, because there were less rewrite rules and hence not so many queries for "welk(e)"-questions, the system performs a less focused search, which results in worse answers.

Another matter that catches the eye is the way of responding on questions that require a very specific location as an answer. A lot of these questions were answered incorrectly. One logical explanation of this issue is that because the answer has to be very specific it is hard to find exactly that answer. An answer like "Pretoria" (on the question "Wat is de hoofdstad van Zuid-Afrika?") is easier to find than "25 kilometer ten westen van Bonn" (answer of the question "Waar ligt Euskirchen?"), because it is less specific. A second explanation for this matter is the use of the named entity filter causing to miss some answers. Because it is quite restrictive to avoid nonsense answers (for example by discarding answers starting with stop words), it could also remove this kind of answers. In the future the filter could be made somewhat more intelligent to adapt itself for this kind of questions.

Questions which the system can handle very well most of the time are questions like "Wie is de president van….?" or "Wat is de hoofdstad van…?". For thiskind of questions a lot of web pages are available and hence there is more information available to base an answer on.

4. Conclusions and future work

The main goal of this research was to determine whether the approach to use data redundancy on the Web for question answering, as described by Dumais, Banko and Brill [2], is also suitable for Dutch questions. To investigate this, their algorithm was implemented in the core of a former made QA-system by van Langen [4]. Subsequently the developed system, simpleQA, was evaluated on a CLEF corpus. Evaluation proved simpleQA significantly outperforms van Langen's system. Bouma's system, Joost [1], performs better, but this system uses a full-strength Dutch dependency parser. Whereas Bouma claims that advances in QA should come from "strengthening the Dutch NLP infrastructure", we believe that there is a lot of room for improvement using simple retrieval techniques that take advantage of the redundancy on the World Wide Web.

The scores between the Microsoft system and simpleQA cannot be compared one on one, because the systems capabilities and the evaluation methods are not the same. However, despite these differences, the scores of simpleQA and the Microsoft system seem quite similar. The Microsoft system method was more

restrictive, but used more NLP than the simpleQA system does. So roughly can be said the simpleQA system performs rather well, even though it has a smaller set of web pages available (the Dutch collection of web pages is a lot smaller than the English one).

Recommendations for future work to improve the performance of simpleQA are to extend the number of rewrite rules and to apply more sophisticated filtering and tiling.

More rewrite rules can be implemented by making minimal use of some NLP techniques, for instance a lexicon (e.g. Dutch WordNet) or a part-of-speech tagger. A lexicon would be convenient for making passive form queries or to identify other relations between words. A part-of-speech tagger could help to determine where parts-of-speech begin and end, which would for instance be of use when rewriting “welk(e)”-questions.

The actual filtering and tiling parts of the system certainly help it to perform better, but they can be improved. At the moment filtering is implemented in the same way for each question, but it could be more focused on the kind of question. Also with regard to the tiling part can be made some progress. At the moment only answers that exactly contain smaller answers are merged, but also answers that would differ in one or two characters could be merged (e.g. Tokyo and Tokio). This would contribute to the performance of the system.

References

[1] G. Bouma, Question answering for Dutch using dependency relations, September 2003, Groningen, the Netherlands, http://odur.let.rug.nl/~gosse/Imix/project_description.pdf.

[2] S. Dumais, M. Banko, E. Brill e.a., Web Question Answering: Is more always better?, in *Proceedings of the 25th Annual International ACM Sigir Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pp. 291-298, Microsoft Research, 2002.

[3] Google Web APIs, http://www.google.com/apis/.

[4] M.C.G. van Langen, Building a system for answering Dutch Person Questions, in *Proceedings of the 2nd Twente Student Conference on Information Technology*, pp 134-141, January 2005.

[5] B. Magnini, S. Romagnoli, A. Vallin e.a., Creating the DISEQuA Corpus: a Test Set for Multilingual Question Answering, In Working Notes for the CLEF 2003 Workshop, Trondheim, Norway, August 2003.

[6] M. de Rijke, Question Answering: Now and Next, Slides Invited Talk, University of Twente, October 2005.