Component Evaluation in a Question Answering System

Luís Fernando Costa[1], Luís Sarmento[2]

[1] Linguateca node at SINTEF ICT, Norway
Pb 124 Blindern, 0314 Oslo, Norway
luis.costa@sintef.no
[2] Linguateca node at FLUP University of Porto, Portugal
Via Panorâmica s/n, 4150-564 Porto, Portugal
las@letras.up.pt

Abstract

Automatic question answering (QA) is a complex task, which lies in the cross-road of Natural Language Processing, Information Retrieval and Human Computer Interaction. A typical QA system has four modules – question processing, document retrieval, answer extraction and answer presentation. In each of these modules, a multitude of tools can be used. Therefore, the performance evaluation of each of these components is of great importance in order to check their impact in the global performance, and to conclude whether these components are necessary, need to be improved or substituted.

This paper describes some experiments performed in order to evaluate several components of the question answering system Esfinge. We describe the experimental set up and present the results of error analysis based on runtime logs of Esfinge. We present the results of component analysis, which provides good insights about the importance of the individual components and pre-processing modules at various levels, namely stemming, named-entity recognition, PoS Filtering and filtering of undesired answers. We also present the results of substituting the document source in which Esfinge tries to find possible answers and compare the results obtained using web sources such as Google, Yahoo and BACO, a large database of web documents in Portuguese.

1. Esfinge

Esfinge (http://www.linguateca.pt/Esfinge/) is a question answering system developed for Portuguese and it is based on the architecture proposed by Eric Brill (2003).

It relies heavily on regular expressions. These expressions are used to classify the questions and to generate a set of patterns of plausible answers used for searching snippets of text where the answers hopefully can be found (either in a document collection and/or in the Web). When Esfinge is not able to recover any snippets using the standard patterns, it uses the module Lingua::PT::Stemmer freely available at CPAN to create more general search patterns. If neither of these strategies is helpful in recovering snippets of text, the system stops its execution and returns the answer NIL (meaning that it was not able to answer the question).

When Esfinge finds snippets of text, it proceeds to extract word n-grams from these snippets (length 1 to 3).

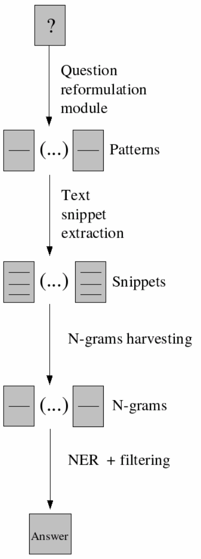

Figure 1: The architecture of Esfinge

Then Esfinge proceeds by ranking these n-grams according to their frequency, length and the patterns used to recover the snippets where the n-grams were found (these patterns have an a priori score associated). The n-grams are scored using the formula:

N-gram score = $\sum$ (F * S * L),

through the first 100 snippets resulting from the web search; where F is the n-gram frequency, S is the score of the search pattern that recovered the document and L is the n-gram length.

Identifying the type of question is very useful in (the task of) searching for an appropriate answer. For example a question beginning with *"When…"* suggests that most likely the answer will be a date. Esfinge has a module that uses the named entity recognition (NER) system SIEMES (Sarmento, 2006a) to detect specific types of answers. When the type of question leads to one or more of those named entity categories (e.g.: Human, Country, Date and Quantity), the 200 best scored word n-grams from the previous modules are submitted to SIEMES. The results from the NER system are then analyzed in order to check whether it recognizes named entities classified as one of the desired categories. If such named entities are recognized, their position in the ranking of possible answers is pushed to the top (and they will skip the filter "Interesting PoS" described ahead).

The list of possible answers obtained is then submitted (by ranking order) to several filters:

1. A filter that discards words contained in the questions. Ex: the answer *Eslováquia* is not desired for the question *Qual é a capital da Eslováquia?* (What is the capital of Slovakia?) and should be discarded.
2. A filter that rejects answers included in a list of "undesired answers". This list includes very frequent words that do not answer questions alone (like *pessoas/persons*, *nova/new*, *lugar/place*, *grandes/big*, *exemplo/example*). It was built with the help of Esfinge log (which records all the answers analysed by

the system). Later some other answers were added to this list, as a result of tests performed with the system. The list includes now 92 entries.

3. A filter that uses the morphological analyzer jspell (Simões & Almeida, 2002) to check the PoS of the various tokens in each answer. This filter rejects the answers whose first and last answer are not common or proper nouns, adjectives or numbers. Using this simple technique it is possible to discard incomplete answers beginning or ending with prepositions or verbs as for example *George Bush chegou* (George Bush arrived). Most likely the right answer will be just *George Bush.*

The final answers of the system are either the best scored candidate answers that manage to go through all the previously described filters or NIL if neither N-gram passes all filters. There is yet a final step in the algorithm where the system searches for longer answers. These are answers that include one of the best candidate answers and also pass all the filters. For example, the best scored answer for the question *Who is the British prime minister?* might be just *Tony*. However, if the system manages to recover the n-gram Tony Blair and this n-gram also passes all the filters, it will be the returned answer.

2. Evaluating the system

Esfinge participated in the last two editions of the QA track in CLEF (Vallin, 2005) as described in (Costa, 2005a; Costa, 2005b). In this evaluation contest, the participating systems received 200 questions and a document collection and had to return the answers, as well as name the documents in the document collection supporting them.

Two runs were submitted: in the first one, the system searched the answers in the Web (submitting queries to Google and parsing its results page) and used the CLEF document collection to confirm these answers (24% right answers). In the second one, it searched the answers in the CLEF document collection only (22% right answers).

After results were made public, the organization supplied a file with the solutions, which was used to evaluate the experiments described in this paper.

Three different types of component evaluation are reported in this paper: error analysis based on the system's log file, component removal and component substitution.

These evaluation techniques can be used to evaluate different types of components in a QA system. It is difficult to detect components with minor contributions just analyzing the logs, so the second technique can be useful in this task. On the other hand, it is not possible to run the system without, for example, the document recovery module, since all the following modules depend on that one. Therefore, log analysis is more appropriate to evaluate that module. This evaluation technique can also be used to identify components causing severe errors. The logical step to take after using the latter evaluation techniques is to test alternative components and check whether they can improve the results.

Each of the aforementioned techniques is described and discussed in the following sub-sections.

2.1. Error analysis

The first technique is an error analysis based on the system's log file. This file contains all the answers that were checked, including the reason why they were rejected. The analysis of this data makes it possible to determine reasons for system failure such as if no source documents were retrieved or if the right answers were not present in the retrieved documents, etc.

Table 1 presents typical causes for wrong answers for the two runs submitted to CLEF 2005: (i) Run A where Esfinge searched the answers in the Web and used the CLEF document collection to confirm these answers and (ii) Run B where Esfinge searched the answers in the CLEF document collection only.

Problem	No. of wrong answers	
	Run A	Run B
No documents retrieved in the document collection	52	51
No documents retrieved containing the answer	22	4
Tokenization	16	1
Answer length >3	10	9
Answer scoring algorithm	26	58
Missing patterns in "question pattern"/"answer type" file	6	6
Named Entity Recognition	7	20
Filter "answer contained in question"	1	1
Filter interesting PoS	0	1
Answer justification	10	4
Search for more complete answers algorithm	2	2
Total	152	157

Table 1: Causes for wrong answers

The results of this analysis pointed out that most of the errors were caused by the document retrieval module. This module did not find documents in 25% of the cases (even though there were documents in CLEF document collection containing relevant answers). In addition (predominantly in the run that uses text from the Web) there are also cases where the retrieved documents are not relevant to answer the questions. Run B performs better in the task of finding documents containing answers, but the answer scoring algorithm performs worse (which is not surprising since the algorithm explores information redundancy and the Web has much more documents than CLEF document collection).

2.2. Component Removal

This technique consists in running the system without some of its components/features. Experiments were performed without the NER system, without the morphological analyzer, without the stemmer and without using the list of 'undesired answers'. The results obtained in each of these experiments (summarized in table 2) can be compared with the complete system (where all the components/features are used) and allow one to measure how each of the components contributes (or not) to the global result.

The error analysis (condensed on table 1) provided an insight on the problems affecting the system performance at CLEF 2005. Some effort was invested in the problems that seemed easier to solve. Namely on the "Error in tokenization", "Named Entity Recognition" and "Missing patterns in the file question pattern/answer type". The results of the complete system after these improvements using the same strategy as in Run A are presented in column Run C (Control Run).

Although CLEF organization has a taxonomy to classify the questions (ex: measure, person, location, organization, etc), we preferred to use a specific taxonomy more suited to evaluate the particular features of Esfinge to present evaluation results (Table 2). The categories of that taxonomy include mostly the types of question patterns and also some of the categories identified by the NER system. The percentages of exact answers in this table are relative to the total number of questions (column No. of Q.).

Type of question	No. of Q.	% of exact answers				
		Run C	Run D[1]	Run E[2]	Run F[3]	Run G[4]
People	47	30%	19%	28%	34%	28%
Which X	36	31%	--	19%	31%	22%
Place	33	30%	27%	36%	42%	36%
Who is <HUM>	27	26%	--	11%	26%	19%
Quantity	18	17%	6%	17%	17%	17%
Date	15	53%	20%	40%	47%	53%
What is X	15	27%	--	13%	33%	0%
What is X called	5	60%	--	40%	60%	40%
Name X	4	25%	--	0%	25%	0%
Total	200	31%	24%	24%	34%	26%

1) No NER, 2) No PoS Filtering, 3) No Stemmer, 4) No list of 'undesired answers'

Table 2: Results when removing some components of the system

Both the NER system on one side and the PoS filtering and the list of 'undesired answers' on the other were shown to improve the system's performance (for different types of questions). The results of Run D (which did not use the NER system) show that the use of such a system can improve the performance mainly in questions with answers of type People, Quantity and Date. In the results of Run E (where the PoS filtering was not performed) and Run G (where the list of 'undesired answers' was not used), one can see that this techniques improve the accuracy in questions matching patterns like *Which X*, *Who is <HUM>* and *What is X*.

On the other hand, the stemmer seems to slightly deteriorate the results (Run F).

2.3. Component Substitution

The third type of component evaluation is the next step to the previous two techniques. After detecting the components causing more problems or components that are not helping the system, one can check whether other components providing the same (or similar) functionalities can obtain better performances.

In the experiments described in this section we studied the use of different text sources. Namely searching the possible answer patterns using Google and Yahoo APIs for Web search (freely available for research purposes) and using the API for BACO (Sarmento, 2006b) which is a very large textual database built from the WPT03 collection, a publicly available crawl of the whole Portuguese Web in 2003. Table 3 and Table 4 summarize the results obtained in these experiments. The percentages of exact answers in table 3 are relative to the total number of questions (column No. of Q.), whereas in table 4 they are relative to the number of questions with an answer in CLEF document collection for which it was possible to find matching possible answer patterns in the auxiliary text source.

From a global point of view (last row in Table 3 and Table 4), results obtained show the expected performance curve: the larger the document base to be queried for possible answer patterns, the better the performance of the system. However, performance seems to grow sub-linearly with document base size. If our estimates regarding the number of documents written in Portuguese indexed by Google and Yahoo are correct (Costa, 2006), we may see that the impact of searching a document larger document base (Yahoo is 75% larger than Google) is proportionally inferior to the performance gain obtained (~7%). Using both Google and Yahoo also resulted in performance gains of approximately only 10%. The most surprising result is that using BACO as document base, which is substantially smaller than Google or Yahoo, resulted in decreasing the performance approximately just 52%.

Such a disparity between the performance and document base size may be explained by several factors. The first one is related to how duplicates are handled in Google or Yahoo. We have no estimate regarding the number of duplicate and quasi-duplicates that exist on the web-search engines. The second one is that even if Google and Yahoo had no duplicates, the information stored in these services is obviously unbalanced: there are many documents about the most popular subjects and facts and much less on certain topics. The effect of adding new documents to the text base does probably not have much impact on less popular topics and consequently Esfinge will not improve much its performance. The last reason for this disproportion, which may be of some theoretical importance is that although we present no proof of this claim, Esfinge may be reaching its performance limit, under its current implementation. The techniques used in Esfinge, may have a nearby asymptotic limit which will not be passed no matter how large the document size grows. This will be subject of future work.

Type of question	No. of Q.	% of exact answers				
		Run H[1]	Run I[2]	Run J[3]	Run K[4]	Run L[5]
People	47	21 %	32%	34%	21%	21%
Which X	36	28 %	22%	31%	19%	22%
Place	33	45 %	45%	52%	21%	18%
Who is <HUM>	27	30 %	26%	30%	11%	15%
Quantity	18	17 %	17%	22%	17%	17%
Date	15	27 %	20%	20%	33%	27%
What is X	15	7 %	27%	13%	0%	0%
What is X called	5	60 %	40%	60%	40%	40%
Name X	4	0 %	0%	0%	0%	0%
Total	200	27 %	29%	32%	19%	19%

Table 3: Results using texts from different sources

Run	No. of docs in the auxiliary text source	No. of Q.	% of exact answers
Run H[1]	34.900.000[6]	178	22%
Run I[2]	60.500.000[6]	130	27%
Run K[4]	1.500.000	79	15%
Run L[5]	1.500.000	94	16%

Table 4: Results using texts from different sources (considering only the questions with an answer in CLEF document collection for which it was possible to find matching possible answer patterns in the auxiliary text source)

1) Snippets from Google, 2) Snippets from Yahoo,
3) Snippets from Google + Yahoo,
4) Snippets from BACO,
5) Snippets from BACO (using stemmer),
6) Estimation of the number of indexed documents in Portuguese (Costa, 2006)

3. Conclusions

The experiments described in this paper show that the use of the NER system, PoS filtering and the list of 'undesired answers' improve the performance of the QA system Esfinge (for different types of questions). Regarding the results obtained with the stemmer the results are not conclusive: with the setup used in CLEF-2005, the use of the stemmer deteriorates the results; on the other hand when using BACO as an auxiliary text source the stemmer improves the results slightly.

As expected, the larger the document base to be queried for possible answer patterns, the better the performance of the system. Nevertheless, there is a large disparity between the increases in the document base size with the corresponding gains in performance. This seems to indicate that Esfinge may be reaching a performance limit, under its current implementation.

4. Acknowledgements

This work is financed by the Portuguese Fundação para a Ciência e Tecnologia through grant POSI/PLP/43931/2001, co-financed by POSI.

5. References

Brill, Eric (2003). Processing Natural Language without Natural Language Processing. In Gelbukh, A. (ed.), *CICLing 2003*. Berlin Heidelberg: Springer-Verlag, LNCS 2588, pp. 360-9.

Costa, Luís (2005). First Evaluation of Esfinge - a Question Answering System for Portuguese. In Carol Peters, Paul Clough, Julio Gonzalo, Gareth Jones, Michael Kluck & Bernardo Magnini (eds.), *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum (CLEF 2004)* (Bath, UK, 15-17 September 2004), Heidelberg, Germany: Springer, LNCS 3491, pp. 522-533.

Costa, Luís (2005). 20th Century Esfinge (Sphinx) solving the riddles at CLEF 2005. In *Working Notes for the CLEF 2005 Workshop* (Vienna, Austria, 21-23 September 2005).

Costa, Luís (2006). Esfinge – a Question Answering System in the Web using the Web. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (Trento, Italy, April 3-7, 2006)

Sarmento, Luís (2006). SIEMÊS - a named entity recognizer for Portuguese relying on similarity rules. In *VII Encontro para o processamento computacional da língua portuguesa escrita e falada (PROPOR'2006)* (Itatiaia, RJ, Brasil, 13-17 de Maio de 2006).

Sarmento, Luís (2006). BACO - A large database of text and co-occurrences. In *Proceedings of LREC 2006* (Génova, Italy, May 22-28, 2006).

Simões, A. M. & Almeida, J.J. (2002). Jspell.pm - um módulo de análise morfológica para uso em Processamento de Linguagem Natural. In: Gonçalves, A. & Correia, C.N. (eds.): *Actas do XVII Encontro da Associação Portuguesa de Linguística (APL 2001)* (Lisboa, 2-4 Outubro 2001). APL Lisboa, pp. 485-495.

Vallin, Alessandro, Bernardo Magnini, Danilo Giampiccolo, Lili Aunimo, Christelle Ayache, Petya Osenova, Anselmo Peñas, Maarten de Rijke, Bogdan Sacaleanu, Diana Santos & Richard Sutcliffe (2005). Overview of the CLEF 2005 Multilingual Question Answering Track. In *Working Notes for the CLEF 2005 Workshop* (Vienna, Austria, 21-23 September 2005).