Hunting answers with RAPOSA (FOX)

Luís Sarmento
Faculdade de Engenharia da Universidade do Porto & Linguateca
las@fe.up.pt

Abstract

In this paper, we will present RAPOSA, an early prototype of a question answering system for Portuguese that heavily relies on a named-entity recognition (NER) system to discover answers for several types of factoid and simple definition questions. We will describe the architecture of RAPOSA and explain the role of the NER system in the question answering procedure. We will also compare and discuss the results of the two runs submitted to the QA@CLEF05 evaluation track, each one using a different question answering strategy. We will analyze the major sources of error and we will then propose several solutions to solve the most severe problems.

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; H.3.7 Digital Libraries; H.2.3 [**Database Managment**]: Languages—*Query Languages*

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, named-entity recognition

1 Introduction

RAPOSA (FOX) is a prototype question answering (QA) system for Portuguese that is being developed in Faculdade de Engenharia da Universidade do Porto as a subsidiary of a larger project that aims at developing wide-scope semantic analysis tools for Portuguese. The RAPOSA project was started mainly because the question answering task provides a good application scenario for validating the capabilities of our semantic analysis tools, and also for guiding its future developments.

The version of RAPOSA that participated in the QA@CLEF06 track is still very simple and is able to address only a very limited type of questions, mainly simple factoid questions that involve people, locations, dates and quantities. Contrary to other question answering systems for Portuguese that make use of extensive linguistic resources [2] or deep parsing techniques [8], RAPOSA uses shallow parsing techniques and relies on the semantic annotation produced by our named entity recognition (NER) system SIEMÊS [9], one of the key components of the suite of analysis tools being currently developed. As we will explain more thoroughly in the next sections, SIEMÊS is used to tag a list of text snippets where candidate answers is believed to be found, extracted from the answer collection. For the type of questions being currently addressed,

RAPOSA assumes that the correct answer is one of the entities tagged by SIEMÊS, and its job is thus to select the right one(s). SIEMÊS is also used during the question parsing stage in order to identify relevant entities that may exist in the question. Other similar approaches that also heavily rely on named-entity recognition as the major source of semantic information are described in [3] and [12].

As a result of its dependency to SIEMÊS, RAPOSA is currently limited to answer questions that involve factoids and simple definitions, although the number of different entities that SIEMÊS is able to tag is relatively high (more than 100). Therefore, RAPOSA is not able to answer list questions and definition / "how to" questions that involve an explanation sentence. SIEMÊS does not yet resolve co-reference and anaphoric references, which is also a severe limitation when addressing questions whose answer can not be explicitly found in the scope of one sentence. However, RAPOSA has two different search strategies, one of which allows coping with some of these limitations.

The positive side of this dependency is that we have now an additional method for testing SIEMÊS through a user-centered application. This indirect evaluation method is complementary to the direct evaluation information that may be obtained by participating in NER evaluation campaigns for Portuguese [11], and is especially interesting because it re-focus the task on practical needs rather than on a definition. Question answering imposes some very practical requirements over the named-entity recognition task (and semantic analysis in general) that helped us defining new goals for SIEMÊS. During the development of RAPOSA, we came across with some apparently simple questions that led us to improving SIEMÊS with new sets of rules for tagging elements that are not usually considered named-entities but are extremely useful for finding anwers to certain types of questions (namely those involving people).

2 The architecture of RAPOSA

The architecture of RAPOSA follows the standard approach, and consists of a pipeline of 4 blocks, each one responsible for a different stage in the question answering problem:

1. Question Parser: receives a question in raw text, identifies the type of question, its arguments, possible restrictions and other relevant keywords, and transforms it to a canonical form. The admissible types of answer for the question are also identified at this stage.

2. Snippet Extractor: this block is responsible for retrieving snippets of texts (currently sentences) from the answer collection, using the information present in the canonical form of the question produced, by the previous block. The Snippet Extractor may return several text snippets.

3. Candidate Generator: after tagging the previously found snippets with SIEMÊS, this block tries to find candidate answers using two optional strategies. Candidate answers are restricted to the set of admissible types found by the Question Parser. Several candidate answers may be found

4. Answer Selector: this block select one answer from the list of candidates found by the previous block. At the moment selection is being made based on redundancy, i.e., on the number of supporting snippets for each candidate.

Note that SIEMÊS is used in two of these blocks, namely in the Question Parser and in the Candidate Generator block. All the four blocks are still in a very preliminary stage of development.

2.1 The Question Parser

The Question Parser operates in two steps. First it invokes SIEMÊS in order to identify named entities in the question, namely people, places, organizations, dates, titles (eg: book titles), which are usually either the arguments of the question to be parsed or some important restrictions.

SIEMÊS also helps to find other important elements such as for example ergonyms, personal titles and jobs, that are relevant for parsing the question. In fact, this last funcionality was added to SIEMÊS during the developement of RAPOSA motivated by the need to produce a better parse for the questions.

In a second step, the Question Parser uses a set of 25 rules to analyze the type of question and to identify its elements. After the type of question has been found, these rules try to identify the arguments of the question, argument modifiers, temporal restrictions and other relevant keywords that may represent good clues for retrieving snippets from the answer collection.

Currently, these rules are only able to address some types of questions, namely those that refer to people ("Quem...?" / "Who...?"), to places ("Onde...?" / "Where...?"), to time ("Quando...?" / "When...?" or "Em que ano...?" / "In what year..?") and to quantities ("Quanto...?" / "How many...?"). Among these, a greater care has been given to questions refering to people, while for the others we still do not have enough rules to parse all the possibilities.

Depending of the type of question, the Question Parser also generate a list of admissable answer types that are compatible with the tagging capabilites of SIEMÊS. For example, a "Onde...?" / "Where...?" question one expects the answer to be a location, while for a "Quem...acção?" / "Who...action?" the admissible answer may be a person, a group or an organization, which are all types of entities that SIEMÊS is able to classify. After all the mentioned information has been identified, the question is transformed into a canonical form that can be easily manipulated by subsequent blocks.

With all these parsing limitations, RAPOSA was only able to address a rather small number of the 200 questions test set given for CLEF06. We will explain the experimental scenario in later sections, but for illustrating this point we may say that RAPOSA tried to answer only 34 questions in the first run submitted and 74 in the second run submitted, after relaxing several processing restrictions.

2.2 The Snippet Extractor

The Snippet Extractor queries a MySQL database (http://www.mysql.org) that stores the entire answer collection to obtain a set of text snippets where the answer to a given question might be found. Currently, the answer collection is stored in raw text, one sentence per tuple and the text is indexed using the fulltext index provided by MySQL database engine. This is a simple approach that relies on the text indexing and search capabilities provided by the MySQL engine.

The Snippet Extractor receives the canonical form of the question and produces a SQL query to search the database. Interesting snippets (currently a snippet is equivalent to a single sentence) are those that contain all the arguments in the question (which most of the times are multi-word expressions), and that may also contain some or all argument modifiers and other keywords previously identified during the question parsing stage. Temporal restrictions are still ignored. The Snippet Extractor may optionally perform a "dumb" stemming procedure over all words used in the query *except the words the belong to the arguments* to increase the number of snippets found, although we were not able to assess the impact of this option in the global performance of RAPOSA.

Blindly relying on the MySQL indexing mechanism, as we did, also brings some problems. One of those problems is that the default options of the MySQL exclude from the index any word with less than 4 charecters, even if it is rare enough to be possibly considered a content word. There are however many important words in Portuguese with only three characters that might be relevant to factoid questions such as some family relations ("pai", "mãe", "tia", "avô"...) locations ("rio", "rua", etc...), proper names ("Rui", "Ana", "Sá", etc) and many other common words ("cor", "mil", sul",...) or acronyms. In some cases, these words are automatically ignored by MySQL when performing a search using the fulltext search functions, which usually results in a very large number of noisy text snippets because the smaller words are not used to restrict the search. This problem can be avoided by performing a sequential scan search with a regular expression, but this option becomes impractical for large collections due to performance issues. A more "content-aware" indexing mechanism is needed in order to effectivelly deal with this problem.

Eventually, the Snippet Extractor may be unable to find in the document database any match for the query produced. At the moment, the Snippet Extractor does not try to expand or reformulate the query - which could helpful in some cases - and RAPOSA returns NIL as the answer for the question. RAPOSA gives a relatively high value of confidence to that NIL answer because it believes that there is not enough information in the document collection to produce the appropriate answer. The value of confidence given is fixed to 1.0 but a more reliable evaluation of confidence could eventually be performed based on the number of words in the query and on the result of shorter queries.

In this version of RAPOSA we decided not to pre-process the entire answer collection with SIEMÊS but, instead, we opted to store the collection in raw text and invoke SIEMÊS for each text snippet retrieved. Preprocessing the collection with SIEMÊS would allow us to obtain text snippets already tagged with semantic information, and to query the collection using both keywords and tags. It would also allow us to find answers more quicky and efficiently because invoking SIEMÊS on all snippets found for a given question implies a significant overhead in processing times. However, since RAPOSA is mainly intended to help the development of SIEMÊS (and other semantic tools), it is much more convenient to perform the named-entity recognition procedure for each question, which allows us to imediatly observe the impact of the changes in SIEMÊS over the final results. In future versions of RAPOSA, when the performance of SIEMÊS increases to a higher level, we may opt to pre-proccess the entire test collection and develop an alternative text indexing procedure.

2.3 The Candidate Generator

The Candidate Generator takes as input the question in canonical form and the list of snippets given by the Snippet Extractor. The Candidate Generator then invoques SIEMÊS for each snippet to obtain information about the named-entities and other related elements. RAPOSA assumes that the answer to the question is one of the elements tagged by SIEMÊS, and since the type of admissable answers for the question at stake has already been determined by the Question Parser, the answer is usually already quite constrained.

The Candidate Extractor may use one of two available strategies to find candidate answers. The first one is based on a set of context evaluation rules. For each tagged text snippet, RAPOSA tries to match certain contexts around the position of argument (note that the argument of the question must be present in the text snippets). For example, for a question like "Who is X?" with X being the name of a person, the answer is expected to be a job title or an ergonym so that checking the existence of those elements around ocorrences of X in certain contexts might lead to the candidate answer. In this case, the rule might check for patterns like "... < job title > X ..." or "... X, < job title > ..." with < job title > standing for the element in the text snippet tagged by SIEMÊS as a job title. RAPOSA has currently 25 of such rules for dealing with question of the type "Who is < job title >?" and 6 rules to deal with questions of the type "Who is < person name >?".

In this version of RAPOSA we were not able to develop similar rules for dealing with other types of questions, so we developed a second strategy for finding candidate answers in the tagged text snippets. This is a much simpler strategy that collects as a possible answer any element tagged with a semantic category that is considered an admissable answer to the question at stake. For example, in answering a question like "When was < EVENT > ?" the Candidate Generator will collect all elements tagged as < date > in the text snippets (which match the string < EVENT >) provided by the Snippet Extractor. Although this strategy may potentially be very noisy, since more than one compatible element may exist in the snippet, one expects the correct answer to be one of the most frequent candidates extracted, provided that there is enough redundancy in the answer collection. We call this *simple type checking strategy.*

The output of the Candidate Generator, using any of the two strategies, is a set of tuples containing the candidate answer and the text snippets as the supporting evidence for the answer. This list may be rather long, especially when the second strategy is employed. There is also the possibility of not finding any candidate answer in the text snippets given by the Snippet

Extractor. In those cases, RAPOSA produces a NIL answer for the question, but it gives a low value of confidence to the answer (0.25) to acknowledge the fact that such result is possibly due to the lack of better analysis capabilites.

2.4 The Answer Selector

The job of the Answer Selector is (i) to choose one of the candidate answers produced by the Candidate Generator, (ii) to select the best supporting text snippets and (iii) to assign a confidence value to that answer. For now, RAPOSA is not dealing with list questions, so answers are expected to have only one element (factoid or simple definition). The job of the Answer Selector is in theory quite complex, especially the task of deciding the best supporting snippets and giving an appropriate confidence value to the answer. But, again, the strategies employed in the current version of RAPOSA are very simplistic.

For deciding which of the candidates is the best answer, the Answer Selector calculates the number of supporting snippets for each candidate and chooses the one with the higher number of snippets. When candidates are generated using the context matching strategy of the Candidate Generator, this simple procedure is expected to lead to good results since those rules tend to be quite precise in choosing candidates, and it is quite difficult to generate wrong candidates with the same number of supporting snippets. Also, if the candidate is generated using the simple type checking strategy, and assuming the answer collection has enough redundancy, the best snippet-supported candidate is probably the right answer. However, using this strategy, it is quite possible that, due to lack of redundancy in the answer collection, the Candidate Generator obtains more than just one "top" candidate of the same type, all supported by the same number of snippets, possibly only one snippet. In those cases the Answer Selector chooses one of the candidate randomly, which obviously leads to many incorrect answers.

Choosing the supporting snippets and determining the confidence level of the answer is also done in a simplistic way. Chosing the "best" supporting snippets is quite straight forward when the Candidate Generator uses the context matching strategy: supporting snippets match a very specific patterns, and almost always have explicit information for supporting the answer. However, if the candidate answers are obtained using the simple type checking strategy, the Answer Selector has no way, at the moment, of deciding if a given snippet (containg both the argument and the chosen candidate answer) really supports the answer. So, in both cases, the procedure is simply to randomly choose up to 10 supporting snippets associated with the chosen answer. The confidence level assigned to the such an answer is 1.0, which obviously disregards important information such as the number of alternative candidates available and the corresponding number of supporting snippets. Improving the way the confidence level is assigned is a matter of future work.

3 Results in CLEF06

We were able to submit two runs. The first run, R1, was configured to extract candidate answers using only the context matching rules, i.e. using the most restrictive and hopefully the highest precision strategy of the Candidate Extractor. Because we were only able to develop rules for "Who is < job title >?" and "Who is < person name >?" questions, for R1 only 34 questions from the 200 question set were addressed. The second run, R2 applied the same strategy for the questions the R1 was attempting to answer but it used the more relaxed type checking strategy for trying to answer place ("Onde...?" / "Where...?"), time ("Quando...?" / "When...?" or "Em que ano...?" / "In what year..?") and quantities ("Quanto...?" / "How many...?") questions. Using this combined strategy RAPOSA tried to answer 74 questions in run R2.

The results we present in the following tables refer only to the questions that RAPOSA has tried to answer and therefore diverge from the official results given by the QA@CLEF organization. We have manually checked the corresponding answers against the list of correct answer given by the organization. We considered an answer correct if and only if it exactly matches the answer given in the answer list. Using this criteria we considered all partially correct answers as incorrect,

type	correct	incorrect	$\sum$
ANS	5	11	16
NUL	1	14	15
NUL2	0	3	3
$\sum$	6	28	34

Table 1: Results of run R1

type	correct	incorrect	$\sum$
ANS	10	24	34
NUL	7	29	36
NUL2	0	4	4
$\sum$	17	57	74

Table 2: Results of run R2

as well as answers that contained more information that what it was required (e.g: a complete date instead of just the year). We will make the distinction between three types of cases among the questions that RAPOSA tried to answer:

1. ANS: questions that were answered and supported by a certain number of text snippets.
2. NIL: questions to which RAPOSA did not find any answer after analysing snippets provided by the Snippet Extractor.
3. NIL2: questions to which RAPOSA was not able to find any snippet in the document collection to search for the answer

We used the confidence values assiged to answers to identify these three possible cases. The results of both runs are given in Table 1 and Table 2. Precision values for both runs are given in Table 3.

3.1 Discussion of the results

The results in Table 1 and 2 show several interesting differences between runs R1 and R2. RAPOSA was able to answer more than twice as many questions in run R2 than in R1. At first sight, it seems surprising that run R2 obtained much better global precision values than R1 - 0.25 against 0.18 - since R1 supposedly used a much more precise strategy. In fact, if we consider just the questions that RAPOSA was able to answer, run R1 exhibits a slightly better perfomance than R2 - 0.31 against 0.29 - but this difference is not enough to compensate for the extremely low cability of R1 in obtaining answers after analysing the text snippets given by the Snippet Extractor. This means that the context analysis rules we developed apply only to a very limited number of cases, so that in almost half the cases RAPOSA can not extract any candidate answer with them.

On the other hand, the rather simpler type checking strategy that was combined in R2 was able to maintain a value of precision comparable to run R1 for those questions to which an answer was found (ANS). However, run R2 was much more efficient in avoiding false negatives, i.e., in

	R1	R2
P(ANS)	0.31	0.29
P(NUL)	0.07	0.19
P(NUL2)	0	0
P($\sum$)	0.18	0.23

Table 3: Partial and Global Precision values for runs R1 and R2

assigning a NUL answer when there was in fact one answer in the collection. Apparently, this simpler strategy provides a better recall without comprimissing to much the precision, by being more efficient in avoiding the false negatives. But one has to consider that the chance of avoiding false negatives may be easier for questions that refer to dates and quantities because identifying and classifying these type of expressions is easier than for other named-entities (namely people, locations, organizations, book titles, etc.).

Still, it is rather suprising to see how low the precision of answers obtained using the context analysis rules actually is: 0.31. Looking in more detail at the 11 wrong answers produced by RAPOSA in run R1, we observe that most of the incorrect answers have only one supporting snippet and, except for one case, the problem does not have to do directly with the context analysis rules. In two of the cases the problem comes from SIEMÊS, which was not able to correctly delimit the elements to be extracted. For example, in one case the correct answer was "primeiro imperador da China" but SIEMÊS only chunked "imperador da China". In two other cases, the problem came from the inability of RAPOSA to choose the "best" answer from the set of candidates generated, all of them supported by the same number of snippets (only one). In such cases, RAPOSA chooses randomly among the alternatives, and since some might be wrong, incomplete or under-specified, there is a great probability of choosing a wrong answer. The problem here is clearly with the Answer Selector component that is not able to differentiate among these answers.

In other cases, the supporting snippet obtained (again usually only one) is only slightly related to the topic of the question and is actually misleading for the context analysis rules. This may happen for several reasons, but an important factor is related to the rather naive text indexing capabilities provided by MySQL (at least for Portuguese) that, when combined with the rudimentary stemming procedure we implemented produces many spurious snippets. Since some of the rules are less restrictive than others, some false candidates are thus generated. This shows the importance of the Snippet Extractor component that needs to be able to obtain more relevant snippets for subsequent analysis.

If we analyse the answers from run R2 in more detail it is possible to see that a about 20% of the incorrect answers result from incorrently choosing the answer candidate among those elements in the snippet that in fact have the correct admissable semantic type. For example, RAPOSA answered "Lisboa" to the question "Onde morreu Kurt Cobain?" / "Where did Kurt Cobain die?" because there was a reference to Seattle and to Lisboa in the snippet extracted. Although each one of the admissable elements was correctly tagged and generated an answer candidate, the Answer Selector decided randomly, ignoring any possible semantic or heursitic information (such as preferring the candidate that is closer to the argument in the text snippet).

Another very frequent type of errors in run R2 reveals yet one more weakness of the Snippet Extractor component, that could be easily avoided. After querying the database based on keywords (i.e. the argument and other words in the question) the Snippet Extractor does not check the keywords in the sentence to make sure that they actually refer to expected semantic type. For example, if the argument in the question refers to place, we would want to make sure that the snippets retrived using the keywords actually refer to the argument as a place, and not to any of the possible homographs. The problem with homography is especially severe for question involving places or people because it is very common for people to have surnames that are also names of places. This led to some unexpected answers simply because the Candidate Generator was looking at the wrong text snippets. Most of other errors in R2 came again from a combination of various problems in Snippets Extractor, which collects many irrelevant snippets, with the inability of the Answer Selector to choose the "best" candidate.

3.2 Some preliminary conclusions

After briefly analyzing the results of both run, R1 and R2, there are some preliminary conclusion that can be drawn. First of all, and rather surprinsingly, the simple type checking strategy is able to provide a comparable performance to the context checking strategy, for a much wider set of question types. Apparently, this is the result of the superior capability of the type checking

strategy in avoiding false negatives (the NIL answer when there is in fact one answer in the collection) in many cases. However, for those questions that RAPOSA is able to answer (the ANS line in tables 1, 2 and 3), it is not clear if the tight difference between the precision of R1 and R2 is due of the flexibility of the type checking strategy or if it is due to the tight bottleneck imposed by the Snippet Extractor and, especially, the Answer Selector.

Another conclusion is that any problems that may exist in the Answer Selector become more evident when RAPOSA tries to address questions for which there is not enough redundancy in the answer collection. The Answer Selector may provide a suitable answer whenever the redundancy is high enough to promote one of the candidates, but it is extremelly poor in deciding among equally supported candidates, usually by only one text snippet. This suggests that one way of helping the Answer Selector (besides making it more intelligent) is to try to obtain additional evidence from other large text collection available (for example the WPT03 web collection, or the Wikipedia).

Finally, it is obvious that the Snippet Extractor needs to be improved because spurious snippets were the cause of many wrong answers. This will require a much better indexing mechanism that does not ignore certain short yet important words, and an appropriate query expansion method that is capable of performing stemming in a safer way, or of substituting words by semantically similar ones (except for the argument).

4 Improving RAPOSA

There is plenty of room for improvement in RAPOSA, and it may be achieved through two complementary ways.

First of all, we would like to improve RAPOSA automatically just by improving SIEMÊS. This is in fact related to our main concern, which is using the question answering scenario to motivate the development of our analysis tools. At this level, RAPOSA will immediately benefit if SIEMÊS improves its capability to identify entities with several modifiers, like "primeiro imperador da China". In this particular case, it would be quite easy to improve SIEMÊS with immediate impact on RAPOSA. But processing references to people like "o vice primeiro ministro russo responsável pelas Nacionalidades" ("the russian vice prime-minister responsible for the Nationality Affairs"), or to locations like "a 900 quilómetros a norte dos Açores" ("at 900 kilometers north from Azores") or to time/date like "nos dois primeiros meses de 1994" ("in the first two months of 1994") seems more complex because of the various internal dependencies involved. Since SIEMÊS is still unable to process these kind of structures, this is a good opportunity to prepare it for dealing with such dependencies, and improve RAPOSA almost automatically as a result.

One of the weakest points in RAPOSA is the Snippet Extractor, that produced many spurious snippets. One of the reasons for that to happen was that our stemming procedure too basic, because it simply replaced the last 2 or 3 characters of a word for a wildcard. The reason we used our stemming procedure in the first place (even knowing it was rudimentary) was because it seemed unrealistic to use only the exact forms of the words in the question to retrieve the snippets. However, the way stemming was performed resulted in an over-generalization of the queries. If we, instead, could make use of some sort of controlled thesaurus for expanding the queries we could avoid using the stemming procedure, and even solve other difficulties during the candidate generation stage. For example, for answering a question like "Quem escreveu 'A Capital'?" / "Who wrote 'A Capital'?", instead of generating a query like "escre* 'A Capital'" it would be very interesting to generate several queries by expanding "escreveu" ("wrote") to {"escritor" ("writer"), "autor" ("author"), ...} or to some other paraphrastic constructions. Such controlled expansion requires semantic resources for Portuguese that, as far as we know, are not publicly available. Several author have proposed automatic ways of creating similar resources - for identifying relations among verbs see [4], or see [7] for the discovery of paraphrases - and it seems interesting to replicate those methods for Portuguese in the Snippet Extractor. A interesting approach that has been directly applied in a question answering scenario uses Wikipedia a source for synonyms and hypernyms [6] .

Note that such semantic resources and techniques may also be very useful for the Answer

Selector. After analyzing the results in more detail, we noticed that it is very common that, during the candidate generation stage, two lexically different yet semantically equivalent candidates are found. Lets suppose that for answering a question like "Who is X?" we obtain several candidates each one supported by one snippet only. Among those candidates we find "portuguese musician", "portuguese composer" and several other spurious candidates. If the Answer Selector is able to infer that "composer" and "musician" have some kind of relation, than it might admit that the two candidates are equivalent, and it may consider them as the same answer supported by two text snippets. This would avoid choosing one the answer randomly among the candidates, with great chances of choosing a wrong one. There are several automatic ways of deciding the similarity of two of such answers using large corpora [13] that we might explore in the near future.

The performance of the Answer Selector could also be improved by using an alternative document collection for checking answers and for estimating the confidence value of the answers. Since the lack of redundancy greatly increases the possibilities of selecting a wrong answer, using more documents to obtain additional evidence about each candidate found might provide the level of repetition that the Answer Selector needs to choose the right candidate. Obvious options for such alternative collections are large crawls of the web in Portuguese (e.g.: WTP03 or WBR99) or the Wikipedia. This approach is being already followed by several question answering systems (for example [1] and [5]). We made some experiments with RAPOSA using a MySQL encoded version WPT03 collection [10] as the answer collection and, although several new difficulties arise in parsing the web documents, we were able to find the correct answer for some of the questions in the CLEF set. It seems thus possible to obtain confirmation regarding one of the candidates found using the official CLEF answer collection. Future experiments of RAPOSA will be run over the WPT03 collection. This will also impose new requirements on SIEMÊS, which will need to cope with the rather messy text that one usually finds in web collections.

5 Conclusions

Developing a question answer system, even as simple as RAPOSA, provides valuable insights regarding the development or improvement of semantic analysis tools in general. In our case, during the development of RAPOSA we were faced with many analysis requirements that we did not considered before, and they made us rethink some of the analysis capabilities of our named-entity recognition system, SIEMÊS. Because of this experience, we are now considering extending the number of elements that SIEMÊS will try to identify and classify so that the information required for question answering becomes more readily available after tagging text snippets. This will require both the sophistication of SIEMÊS and the development of more advanced lexical databases that are used by SIEMÊS. By helping us to focus on very specific and practical analysis problems, we consider that the first participation of RAPOSA in the QA track at CLEF was very useful and motivating.

As far as results are concerned, RAPOSA has still a long way to go in order to achieve the results that other more advanced systems already do. But before trying to answer other types of questions, we will now focus on improving the performance of RAPOSA in answering the simpler factoid questions. As we have shown in the previous section, there are multiple possibilities for improvement, both in RAPOSA and in SIEMÊS. Future efforts on RAPOSA will concentrate in improving the query processing capabilites of the Snippet Extractor, in order to reduce the number of spurious snippets found, and in developing better selection machanisms for the Answer Selector, possibly using additional external document collections for choice validation.

6 Acknowledgments

This work was partially supported by grant SFRH/BD/ 23590/2005 from Fundação para a Ciência e Tecnologia (Portugal), co-financed by POSI.

References

[1] David Ahn, Valentin Jijkoun, Karin MUller, Maarten de Rijke, and Erik Tjong Kim San. The Universisty of Amsterdam at QA@CLEF 2005. In *Proceedings of the CLEF 2005 Workshop*, Vienna, Austria, September 2005.

[2] Carlos Amaral, Helena Figueira, André Martins, Afonso Mendes, Pedro Mendes, and Cláudia Pinto. Priberam's question answering system for Portuguese. In *Proceedings of the CLEF 2005 Workshop*, Vienna, Austria, September 2005.

[3] Lili Aunimo and Reeta Kuuskoski. Question Ansering using Semantic Annotation. In *Proceedings of the CLEF 2005 Workshop*, Vienna, Austria, September 2005.

[4] Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain, 2004.

[5] Luís Costa. Esfinge - a modular question answering system for Portuguese. In *Proceedings of the CLEF 2006 Workshop (to appear)*, Alicante, Spain, 20-22 September 2006.

[6] Boris Katz, Gregory Marton, Gary Borchardt, Alexis Brownell, Sue Felshin, Daniel Loreto, Jesse Louis-Rosenberg, Ben Lu, Federico Mora, Stephan Stiller, Ozlem Uzuner, and Angela Wilcox. External Knowledge Sources for Question Answering. In *Proceedings of the 14th Annual Text REtrieval Conference (TREC2005)*, Gaithersburg, MD, November 2005.

[7] Dekang Lin and Patrick Pantel. Discovery of inference rules for Question Answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[8] Paulo Quaresma and Irene Rodrigues. A logic programming based approach to the QA@CLEF05 track. In *Proceedings of the CLEF 2005 Workshop*, Vienna, Austria, September 2005.

[9] Luis Sarmento. SIEMÊS - a named entity recognizer for Portuguese relying on similarity rules. In *PROPOR 2006 - Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada*, pages 90–99, ME - RJ / Itatiaia, Rio de Janeiro - Brasil, 13 a 17 de Maio 2006.

[10] Luís Sarmento. BACO - A large database of text and co-occurrences. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, Genova, Italia, 22-28 May 2006.

[11] Nuno Seco, Diana Santos, Nuno Cardoso, and Rui Vilela. A Complex Evaluation Architecture for HAREM. In *PROPOR 2006 - Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada*, pages 260–263, ME - RJ / Itatiaia, Rio de Janeiro - Brasil, 13 a 17 de Maio 2006.

[12] Rohini K. Srihari and Wei Li. A Question Answering System Supported by Information Extraction. In *ANLP*, pages 166–172, 2000.

[13] Peter D. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, volume Lecture Notes in Computer Science 2167, pages 491–502, 2001.