SZTAKI @ ResPubliQA 2010

David Mark Nemeskey

Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary (SZTAKI)

Abstract. This paper summarizes the results of our first participation at ResPubliQA. Lacking a true question answering system, we relied on our traditional search engine to find the answers. We submitted two runs: a pure IR baseline and one where definition question identification and corpus-specific techniques were employed to improve IR performance. Both runs performed well, and our second run obtained better results than the baseline. However, the drawbacks of relying solely on an IR phase are clearly seen in our runs.

1 Introduction

In this paper, we present the SZTAKI team participation in Question Answering@CLEF 2010, also known as ResPubliQA. In this track, participants receive a set of questions they are expected to answer. A single answer may be returned for every question. This year, two separate tasks were proposed for ResPubliQA. In the Paragraph Selection (PS) task, participants had to return a numbered paragraph containing the answer to the query. In the Answer Selection (AS) task, systems were required to identify the exact answer within the paragraph text.

In our first time participation we only focused on the first part of the QA pipeline: question analysis and information retrieval. Although important for question answering [3], we did not perform document analysis and answer selection.

We relied on our text search engine to find the paragraphs relevant to the question in the corpus. The paragraph with the highest score was returned as the answer. This required good precision from our IR phase. Question analysis was employed to formulate queries that can fulfill this requirement.

Due to the reasons mentioned above, we participated only in the Paragraph Selection task. Furthermore, we concentrated our efforts on the English Monolingual task, which consisted of a body of 200 questions.

The rest of the paper is organized as follows. Section 2 describes the components of our system and the techniques we used. Section 3 presents the submitted runs and a brief analysis of the results. We draw our conclusions in Section 4.

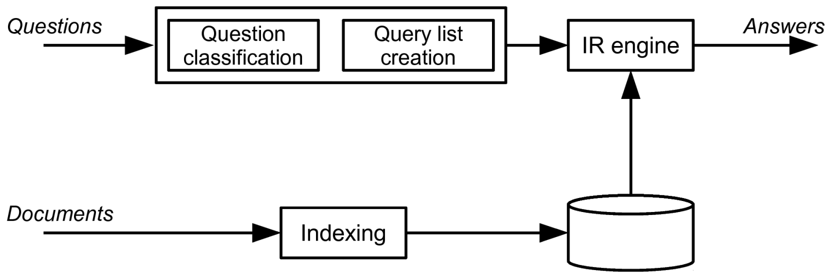
2 System Overview

The architecture and functions of our system can be seen on Figure 1. The system is built around the Hungarian Academy of Sciences search engine [1] based on

Okapi BM25 ranking [6] with the proximity of query terms taken into account [5, 2]. The engine accepts queries that consist of words in an arbitrary tree of OR and AND relationships. Exact matching (“”) is supported as well.

As already mentioned, the system is not a complete question answering system, as it lacks any form of document analysis, document validation or answer selection. The answer to a question is simply the first document returned by the IR module. The role of the question processing module is to formulate queries that maximize IR performance. These components are described in more detail in the next sections.

Fig. 1. A system architecture



2.1 Indexing

Our indexer module generates a simple unigram index from the documents in the collection. We employ the Porter stemmer [4] both on the corpus and the topics. An unstemmed index is created as well to identify exact matches.

The selection of indexing units needed some effort. In the JRC-Acquis corpus, a file contains one document, which is divided into numbered paragraphs. In the EuroParl collection, an XML file consists of several *chapters*, which represent what can be safely considered a ”document”. These chapters are, in turn, segmented into numbered paragraphs. We considered two options.

Document: the whole file (Acquis) or chapter (EuroParl) is indexed, including the title, the contents of all paragraphs and certain metadata, such as the executor id in EuroParl Texts Adopted documents.

Paragraph: Only the paragraph contents are indexed. This is also the obvious choice for the Paragraph Selection task.

We tested both options with the questions in the 2009 training data. Our results are summarized in Table 1.

Even if the numbers are not entirely accurate (the 2009 Gold Standard lists only one valid paragraph for a question, and it disregards duplicates), it can be seen that higher MAP can be achieved for document retrieval. However, due to

Table 1. MAP values achieved by document- and paragraph-based indexing

	EN-EN (95 questions)	**All-EN (500 questions)**
Document	0.528	0.537
Paragraph	0.432	0.476

the lack of an answer selection component, we had to resort to paragraphs as the indexing unit. To compensate for this loss of accuracy, the title of a document or chapter was added to the text of all its paragraphs, albeit with lower weight.

2.2 Query List Creation

Instead of generating one query for every question, we decided to create a *list* of query variations. The first query formulates the strictest constraints of the documents it accepts, and it is followed by increasingly more permissive variants. The default query consists of all non-stopword terms of the question in an OR relationship. The queries are sent to the IR module in this order. Processing of the list stops if at least one paragraph is found for a particular query.

The idea behind using a list of queries is that we assume that for some question types, the passages we are looking for may exhibit certain peculiarities, which might be exploited with an adequately phrased query. We considered the following features:

"Codes": words that contain both letters and digits (legal entries, etc.) We considered this feature important, so such words resulted in two query variants: one, where the word is in an AND relationship with the rest of the words, and the default one.

Quotations: in the JRC-Acquis corpus, quotation marks (") are denoted by the *%quot%* string. This enabled us to include these in the search with regular IR techniques. Therefore, two queries were created for questions that contained quoted word(s): one with the *"AND quot"* string appended to the query, and one without.

2.3 Question Classification

Our system does not yet include a full-fledged question classification module. However, we assumed that in legal texts, such as the JRC-Acquis and the EuroParl corpora, great care is taken to define terms and expressions clearly and unambiguously. If this was the case, recognition of definition questions would benefit the precision of our system substantially. Therefore, an experimental component that identifies definition questions was implemented.

Pattern creation We classified the questions in the 2009 training data manually, and collected a set of patterns that occurred in definition questions. Typical examples include "What is a XY?" and "What is meant by XY?". These informal patterns were then formalized as regular expressions, and matched against the questions in this year's evaluation set. Regular expressions allowed us to decide if the question was a definition question and retrieve the terms or expressions to be defined at the same time.

Out of the about 24 definition questions we found manually (some of these choices may be debatable), our method successfully identified 18, or 75%.

Exact matching With the expressions to be defined extracted, we needed a way to use them to enhance IR precision. We use question 66: "*What is sports footwear?*" as an example to present our method.

Our first assumption was that the passage where the expression (e.g. "*sports footwear*") is defined may either include it in the same form as in the question, or at least contain all the terms that make up the expression ("*sport*" and "*footwear*"). If both of these assumptions are false, we need to fall back to the default OR query. Therefore, three query variations are added to the list for every definition question:

- the whole expression in quotation marks, which denotes exact matching;
- the terms connected by an AND relation;
- the terms connected by an OR relation (the default).

Based on the formality of the language used, we also assumed that the words "mean", "definition" and their synonyms may actually occur in the paragraph where the term is defined. Hence, for every query in the list, we created an expanded variation that included these words in and AND relation with the rest of the query, and added these new, stricter queries to the list as well.

The listing below shows the query list created from our example question.

```
<query id="66">
  <subquery index="1">"sports footwear" AND (mean OR define OR ...)
  </subquery>
  <subquery index="2">"sports footwear"</subquery>
  <subquery index="3">sport AND footwear AND (mean OR define OR ...)
  </subquery>
  <subquery index="4">sport AND footwear</subquery>
  <subquery index="5">sport OR footwear</subquery>
</query>
```

2.4 Information Retrieval

In our system, the IR phase is responsible for selecting the paragraph that answers the question. We used our own search engine, which ranked the passages with Okapi-BM25 ranking function. Our implementation takes document structure into account and it is possible to assign different weights to the various fields (e.g. title, body, meta-data).

BM25 is a TF-IDF-like ranking, with two additional features. Firstly, it is possible to adjust the effect of term frequency on the score via the k_1 parameter. Secondly, a normalization based on document length is applied to the final score. The degree of normalization can be adjusted by the b parameter.

The following settings were used for passage retrieval:

– The b parameter was set to 0. While document length is usually an important feature in ad-hoc retrieval, our experiences with the 2009 training set indicate no correlation between paragraph length and relevancy. Therefore, we decided against length normalization.
– The weight of the title field is set to 0.5, as it does not belong to the paragraph itself.

As mentioned earlier, the queries in the query list associated with a question are processed until a paragraph is retrieved. If more than one is found, as is usually the case, the one with the highest BM25 score is returned as the answer to the question. Our system did not tag any questions as NOA.

It is worth mentioning that out of the 23 questions that had a query list of at least two elements (in line with section 2.2 and 2.3), there was only one for which the first query did not return any paragraphs. On the one hand, this means that our assumptions about the corpus were right. On the other, since it would be difficult to add further, meaningful constraints to the queries, it also shows that perhaps not much improvement potential is left in this method.

3 Runs Submitted

We submitted two runs for the English monolingual task.

Run 1 was our baseline run. Only the paragraph texts were indexed, and default Okapi parameters ($k_1 = 2$, $b = 0.75$) were used. However, quotation and code detection – and the resulting query variants – were included in this run as well, mostly by mistake.

Run 2 featured all additional techniques described in the earlier chapters, namely:
  – the paragraphs were augmented with the document titles,
  – Okapi parameter b was set to 0,
  – query expansion was performed and exact matches were preferred for definition questions.

Table 2 summarizes our results[1]. The cells show the number of correct answers and average c@1 for the specified run and question type.

Regarding the results, we can observe that run 2 generally yielded a higher number of correct answers. However, only the improvements for definition questions can be considered significant. Nevertheless, it shows the potential of adapting parameters to the collection.

[1] We believe that some of the questions were judged incorrectly. Hence, while the table shows the official results, the numbers may not be 100% accurate.

Table 2. Results for the English monolingual runs.

Run	**Regular**	**Quotation**	**Definition**	**Sum**
# of questions	177	5	18	200
Run 1	116 (0.655)	4 (0.8)	9 (0.5)	129 (0.65)
Run 2	118 (0.666)	5 (1.0)	13 (0.72)	136 (0.68)
Difference	+2 (0.011)	+1 (0.2)	+4 (0.22)	+7 (0.035)

The higher rate of correct answers for definition questions validates our assumptions about the nature of the corpus and proves that exact matching and using meta-information, such as including synonyms of "mean" and "define" in the query can indeed increase precision. However, this naïve implementation may also mislead the retrieval engine by giving too much importance to terms not actually part of the question, as shown by the following example. This paragraph was returned because the term "port facility" occurred in the document title.

Q. *What is a port facility?*
A. *2.4 Terms not otherwise defined in this part shall have the same meaning as the meaning attributed to them in chapters I and XI-2.*

This problem underlines the importance of paragraph validation and analysis, which could have filtered such a passage from the result list. Analysis of the returned paragraphs may also increase precision. Table 3 lists the recall at 1, 5 and 10 passages our system achieved on the 2009 English training data. (Gold standard for the 2010 evaluation data is not available yet.) An answer validator component that examines the top ten candidates may, in the optimal case, improve precision by as much as 77%.

Table 3. Recall achieved on the 2009 training set.

	r@1 = p@1	**r@5**	**r@10**
Recall	0.40625	0.6458	0.71875

4 Conclusions and Future Work

In this paper, we have presented our QA system and our results for the English monolingual task. Our system did not use any post-processing of answer candidates and relied completely on the IR module. Our efforts were mainly focused on improving IR performance by tuning retrieval settings, preferring exact matching for definition questions and including collection-specific elements in the queries.

The results have proven the validity of our approach. We achieved an almost 50% increase in accuracy for definition questions, compared to our baseline run. Tuning of retrieval settings, however, has not yielded significant improvement.

Our results have also shown the fragility of depending on the IR phase alone for question answering. In the future, we intend to extend our system to a complete QA system by implementing question classification for all question types, paragraph validation and answer extraction. Several options to improve the IR phase, such as using thesauri for query expansion and fine tuning retrieval parameters are also yet to be explored.

References

1. Benczúr, A.A., Csalogány, K., Friedman, E., Fogaras, D., Sarlós, T., Uher, M., Windhager, E.: Searching a small national domain—preliminary report. In: Proceedings of the 12th World Wide Web Conference (WWW). Budapest, Hungary (2003), http://datamining.sztaki.hu/?q=en/en-publications
2. Büttcher, S., Clarke, C.L.A., Lushman, B.: Term proximity scoring for ad-hoc retrieval on very large text collections. In: SIGIR '06. pp. 621–622. ACM Press, New York, NY, USA (2006)
3. Monz, C.: From document retrieval to question answering. Ph.D. thesis, University of Amsterdam (2003)
4. Porter, M.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
5. Rasolofo, Y., Savoy, J.: Term proximity scoring for keyword-based retrieval systems. In: Advances in Information Retrieval. pp. 207–218. LNCS, Springer (2003)
6. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. In: Document retrieval systems, pp. 143–160. Taylor Graham Publishing, London, UK, UK (1988)