A Naïve Approach for Monolingual Question Answering

Rohit Bharadwaj, Surya Ganesh, Vasudeva Varma,

{bharadwaj, suryag}@research.iiit.ac.in, vv@iiit.ac.in
LTRC, IIIT-Hyderabad

***Abstract*— This paper talks about the system which we have submitted for the ResPubliQA task. We participated in building the QA system for en-en part. We followed a different method for each question type. In this paper we outline the methods which we adapted and the results which we obtained.**

1 INTRODUCTION

THE main focus of QA is to gain the knowledge of the user's question and retrieve the sentences that are close to the answer. The ResPubliQA task expects the system to understand the question and retrieve the corresponding passage in the text which contains the answer. The architecture of our QA system is 1) Question Analysis 2) Passage retrieval and 3) Passage selection. Question analysis involves the classification of the question into pre-defined question types, extraction of query words and determining the answer type. Passage retrieval searches for passages in the document collection which are likely to contain the answer. Passage selection ranks the list of candidate answers to determine the final answer. First, we introduce the task, and then we describe the pre-processing we carried over the data in section 2. In section 3 we describe the approach which we have followed for answering the questions. Section 4 describes the results we obtained while section 5 presents the analysis and conclusion.

2 PREPROCESSING THE DATA

In ResPubliQA task, we are provided with the data that is delimited into passages and we are expected to return the passage that contains the answer. We are provided with both the question language and the target language in which the answers are to be present. The task is mainly directed towards cross language question answering. We participated in the track with both source and destination languages as English. We indexed the data using Lucene, an open source search library. Lucene implements okapi BM25 retrieval model [6]. Using this search library we have built passage index, that is, each passage in a document is considered as a retrieval unit.

3 OUR APPROACH

Our QA system incorporates pipeline architecture as shown in figure 1. It consists of three core components: 1) Question analysis, 2) Passage retrieval and 3) Passage selection. The implementation details of all the three components are described below.

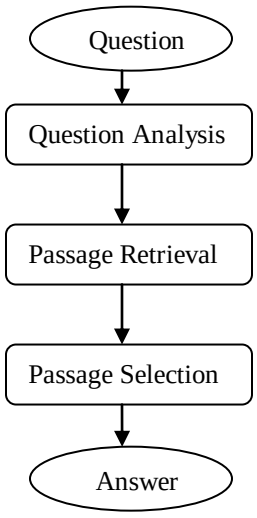

Figure 1: Pipeline Architecture of our QA system

Question Analysis

In question analysis, we classified the given 95 questions into one of the pre-defined classes. The pre-defined question types or classes are *Factoid, Definitive, Reason, Procedure, purpose.* The classification is semi-automatic. As the question classes are fixed, by observation we identified patterns for each of the classes. The patterns for Factoid and Definitive are inter-related and hence we classified the questions under these categories into a single class *FactDef* which was later

sub-divided into factoid and definitive classes. The observed patterns are shown in Table 1.

Question class (Answer type)	Words that must be present in the question (case folded)	Words that should not be present in the question (case folded)	Number of questions
FactDef	what, how, defin, who, where, name	aim, goal, objective, reason, procedure, purpose	46
Reason	why, reason	-	33
Procedure	Procedure	-	10
Purpose	aim, objective, goal, purpose	-	6

Table 1: Generic patterns in questions from different classes

Out of the 46 questions in *FactDef* class, 27 questions are from factoid class and remaining 19 are from definitive class.

The methods followed for each of the question class which includes both passage retrieval and passage selection methodologies are described below.

Factoid

To answer a factoid question, first, we retrieve a set of relevant passages. So, a keyword query constructed by stripping of all the stop words and interrogative words (when, where, which etc.) in the question. This keyword query is given to Lucene to retrieve a ranked set of relevant passages. From this set, one of the passages is given as the answer to a question. Our approach for selecting an answer containing passage is a two step process as described below.

1. **Answer type:** Using the answer type of a question, we identify a set of passages which contain answer candidates. To obtain the answer type of a question, we have implemented a question classifier using support vector machines (SVM) [3]. The classifier was trained on UIUC [2] dataset which consists of 5,500 questions for training and 500 questions for testing. Every question in the dataset was labeled into a coarse grained and a fine grained category from a total number of 6 and 50 categories respectively. We have used the bag-of-words feature to predict the category, that is, the answer type of a question. The classifier showed an accuracy of 86.8%, when tested on 500 questions from the UIUC dataset under the coarse grained classification. And, an accuracy of 78.2% under fine grained classification. As the classification accuracy is higher for coarse grained classification and also because of the limitations of many NER systems to recognize fine grained named entity types in passages, only coarse answer type is used to identify passages with answer candidates.
2. **Density:** Tellex et al. [5] showed that density based measures work well for passage retrieval in QA. So, the passages resulting from the above step are then re-ranked based on the density of the question keywords in them. Density is defined as the average distance between the answer and question keywords in a passage. There are several ways to compute density. We adopt a simple formula as described in [4] to compute density of query terms in a passage.

Finally, among the re-ranked passages, the top ranked passage is produced as the answer given a question.

Definitive

We used answer patterns for definitive questions and used them for passage selection. The question focus or Qword are extracted by removing the stop words (a pre-compiled list) from the question.

The main answer patterns for definitive questions as given in [1] are (where A is the Qword and X is the expected answer)

1) <A; is/are;[a/an/the]; X>

2) <A; comma; [a/an/the]; X; [comma/period]>

   <X; comma; [a/an/the]; A; [comma/period]>

3) <A; [comma]; or; X; [comma]>

4) <A; [comma]; [also] called; X [comma]>

   < X; [comma]; [also] called; A [comma]>
   <X; is called; A>
   <A; is called; X>

5) <X, dash; A; [dash] A; dash; X; [dash]>

6) <X; parenthesis-; A; parenthesis >

As our system does not need to extract the answer but to retrieve the passage, we modified the patterns and extended them by adding few more patterns like "*Qword + means/ mean/ has/*". So effectively the queries used to search the index are the modified queries which are formed by adding the answer patterns. We also queried the index by adding various versions of the modified query like "Qword means", Qword + means, "Qword, called" etc. This resulted in various results for each modified query. For identifying the correct

answer, we performed various experiments like giving boost to results of a particular query, giving weight to each query and calculating the final weight of each result, performing various intersection and union operations for finding the final result on the development dataset. From these experiments, the one that gave most correct results was "prioritizing the search patterns and preserving that order while searching". The optimal priority order of search patterns is shown below.

1. "Qword means"
2. "Qword mean"
3. "Qword is/are"
4. "Qword, called"
5. "called Qword"
6. "Qword was"
7. Qword

So if we achieve result for the first query, then it is given as the final answer. If there are no results then we proceed to the next query. Finally if we have no results for any of the patterns, then the system doesn't answer the question.

All the remaining questions, that is, questions from Reason, Procedure and Purpose types are answered naively by giving a top ranked passage, with a minimum length of N words, from the passage retrieval component as the answer. In our experiments on the development data, we have observed better results for N=25. So, we used the same value for finding the answer to the questions in test data.

4 RESULTS

The test dataset for ResPubliQA task consists of a subset of the JRC-ACQUIS Multilingual Parallel Corpus4, and 500 questions distributed over factoid, definitive, procedure, reason and purpose classes. JRC-ACQUIS is a freely available parallel corpus of European Union legal documents. It comprises selected texts written between 1950 and 2006 with parallel translation in 22 European languages. We used English documents in the corpus. Out of the 500 questions in all languages, we have answered 95 questions which are categorized under monolingual English QA task. The results of our system as provided by the CLEF are shown in table 2.

Total questions	95
Question answered correctly	54
Incorrectly answered questions	37
Questions unanswered	4

Table 2: Results for monolingual English ResPubliQA task

5 CONCLUSION

We described our participation in ResPubliQA task. We have developed a monolingual English QA system. Our system does not rely on any external knowledge resources or any complex information retrieval, information extraction and natural language processing techniques. Instead, it uses an effective combination of naive techniques from the above areas to achieve a decent performance. Our system analyses passages from passage retrieval output to identify the correct answer in the case of factoid and definition questions, whereas, the top ranked passage was produced as an answer for the remaining questions in the test set. The analysis method used for selecting the answer differs for factoid and definition questions.

6 References

[1] Soubbotin Soubbotin Insightsoft-M. Patterns of Potential Answer Expressions as Clues to the Right Answers. In Proceedings of the Tenth Text REtrieval Conference (TREC), 2001.

[2] Xin Li Dan, Xin Li, and Dan Roth. 2002. Learning question classifiers. pages 556–562.

[3] Zhang, Dell and Lee, Wee Sun. Question classification using support vector machines. In proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, 2003.

[4] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[5] Stefanie Tellex and Boris Katz and Jimmy Lin and Aaron Fernandes and Gregory Marton. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2003.

[6] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, 1995.