Question Answering System: Retrieving relevant passages

Hitesh Sabnani and Prasenjit Majumder

Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India.

{hiteshsabnani3, prasenjit.majumdar}@gmail.com

Abstract. This paper discusses the QA system submitted by Dhirubhai Ambani Institute of Information and Communication Technology, India in the ResPubliQA 2010. We have participated in the monolingual en-en task. Our system retrieves a candidate paragraph that contains the answer to a natural language question. Depending on the n-gram similarity score of the candidate paragraph, a decision is made whether to answer the question or not. The objective of our participation is to test our implementation of various strategies like Query Expansion, n-gram similarity matching, and non-answering criteria.

Keywords: Question Answering, Information Retrieval, Natural Language Processing.

1 Introduction

A question-answering system returns textual strings (answer) from a given document collection (corpus), in response to a natural language question [1]. The task of developing a question-answering system can be decomposed into sub-problems like question processing, passage retrieval, and answer extraction [2]. A generic question-answering architecture is shown in Fig. 1. Here, the first step is to index the corpus for facilitating fast and accurate retrieval. In step 2, the natural language questions are converted to structured queries that are to be used on the passage retrieval engine. In step 3, the structured queries developed are used against the index to retrieve a ranked list of passages. Finally, the semantics or the structuring of the question is used along

Step1: CORPUS → INDEX

Step2: NATURAL LANGUAGE QUESTION → STRUCTURED QUERY

Step3: STRUCTURED QUERY + INDEX → RANKED LIST OF PASSAGES

Step4: QUESTION SEMANTICS + PASSAGES → ANSWER

Fig. 1. A generic question-answering architecture

with the ranked list of passages to get an answer to the question.

For every question, our system either returns an answer passage or chooses not to answer it. We are using Lemur Toolkit [3] for passage retrieval. Further sections of this paper discuss the Lemur Toolkit, Methodology of our system, Results, and Conclusion.

2 Lemur Toolkit and the Indri Query Language

Lemur Toolkit is a joint initiative of the CIIR from the University of Massachusetts and LTI from the Carnegie Mellon University for the facilitation of research in the field of Information Retrieval. Indri Search Engine is a part of the Lemur Toolkit. The toolkit is used for indexing the corpus and retrieving the passages for the queries against an index.

The structured query file generated by processing the natural language questions is fed as an input to the Indri Search Engine. This file contains various parameters like the query operators, number of paragraphs to be retrieved, index of the corpus, etc. An example of a query operator is 'combine' which computes the ranked list for a query based on the score calculated by a scoring function [4]. For passage retrieval, the field to be retrieved can be specified in the structured query file.

Some of the query operators of the Indri Query Language [5] that have been used are described below:

1. #combine[](x1 x2): It ranks the documents/passages based on the occurrences of the query terms x1 and x2. If we wish to extract a field rather than a document, we tag the field while indexing and retrieve it by using the query #combine[p] where 'p' is the field. For instance, in our case, we have to retrieve paragraphs instead of documents, which are described by <p> tags in the corpus. So, we index at paragraph level and retrieve using the query #combine[p]. If search is also to be performed at the paragraph level, we use #combine[p](x1.(p) x2.(p)). This ranks the passages according to the occurrences of x1 or x2 in the paragraph tag 'p' rather than the complete document. The scoring function of #combine operator is

$$b_{\#combine} = \prod_{i=1}^{n} b_i^{(1/n)}$$

   Here, b_i is the i^{th} term and n is the number of terms, in the #combine operator. For e.g. in the query #combine[](x1 x2), score would be (score for x1)$^{(1/2)}$ * (score for x2)$^{(1/2)}$.

2. #n(x1 x2): It is used for finding the occurrences of x1 and x2 within proximity of 'n' words. We have used #1(x1 x2) for phrases where x1 and x2 are phrase terms and come one after the other.

Experimentally (ResPubliQA 2009 data), it was found that the results of #combine operator were best suited to our requirements and it gave better retrieval results. So, we have preferred #combine operator over other operators of the toolkit.

3 Methodology

We begin by indexing the corpus with the IndriBuildIndex application of the Lemur Toolkit. Krovetz's algorithm is used for stemming of the terms in the index. For enabling the retrieval of paragraph tags rather than complete documents, documents are indexed at paragraph level. This is done by indicating the paragraph tag <p> in the index parameter file. So, the paragraphs (<p> tags in the corpus) are treated as different documents and retrieved in the retrieval phase.

In the next step, natural language queries are converted to Indri Query Language queries. Initially, the natural language queries are annotated using a POS tagger. We are using Stanford Log-Linear Part-of-Speech Tagger [6] developed by Stanford NLP Lab. The first set of structured queries (used for the baseline run – run1) is generated by having the noun, pronoun, adjective, adverb, preposition, and non-auxiliary verb terms as the parameters of the #combine operator.

This set of structured queries is inputted to the Indri Search Engine, and against the index of step 1 (in Fig. 2), we retrieve the top 100 passages for each question and save the results in a file.

In the next step, the file (containing the top 100 passages for each query) is re-indexed. Further retrieval (in the forthcoming steps) is done on this new index in an effort to reduce the retrieval time by performing the retrieval on a smaller index. Here, the assumption is that in most cases the passage containing the answer exists in the ranked list of top 100 passages. In our experiments on the ResPubliQA 2009 data, we found this was the case for 449 out of 500 questions. The maximum score (c@1) that any participant achieved in ResPubliQA 2009 was 0.61 [7]. So, it was a fair decision to re-index the data in order to reduce the retrieval time.

Step1: CORPUS → INDEX

Step2: NATURAL LANGUAGE QUESTION → INDRI STRUCTURED QUERY

Step3: INDRI STRUCTURED QUERY + INDEX → TOP 100 PASSAGES

Step4: TOP 100 PASSAGES → NEW INDEX

Step5: INDRI STRUCTURED QUERY + QUERY EXPANSION → NEW QUERY

Step6: NEW QUERY + NEW INDEX → NEW SET OF TOP 100 PASSAGES

Step7: NEW SET OF TOP 100 PASSAGES → TOP PASSAGE SELECTED (BASED ON NAMED ENTITY RECOGNIZER AND N-GRAM SIMILARITY)

Step8: TOP PASSAGE → PASSAGE / NO ANSWER (BASED ON THE NON-ANSWERING CRITERIA)

Fig. 2. Architecture of the system

The query analysis on the ResPubliQA 2009 data helped us to know how the common questions are structured and answered. For e.g. a reason question can be asked by 'Why', 'What is the reason', 'What is the purpose' and its answer usually contains terms like 'reason', 'in order', 'due to', and 'because'. Similarly, answer to a question asking a definition is likely to contain terms like 'means' and 'is defined as'. This analysis is used to expand the queries in the query expansion phase. The type of questions has been found by lexical analysis of the terms. We have also extracted phrases from a natural language question. The chunks in which a noun term follows an adjective or a noun, and vice-versa have been identified as phrases. 'United Nations', 'migrating workers', and 'Jason Gibbs' are some of the examples. These phrases have been added to the query using '#1' operator which ensures that the terms in the #1 operator will not be separated by a distance of more than 1 word. For e.g. #1(United Nations) will rank the passages containing 'United Nations' but not containing 'United States of America and Canada are two big nations in North America.' Reason and definition questions have been expanded the most in the sense that they include both phrase terms along with the added terms which are not actually part of the question.

The expanded query in the previous step is now inputted afresh to the Indri Search Engine. The index developed in step 4 (in Fig. 2) is used here. The result of this step is a new set of 100 candidate passages for each query. Further processing is done on this new ranked list.

We use 2 approaches for predicting the final answer. For questions whose expected answer type is found to be a location, person, or an organization, we have used the Stanford Named Entity Recognizer [8], developed by Stanford NLP Lab. These questions are basically the 'Who', 'Whose', 'Whom', 'Which country' questions. The Named Entity Recognizer tags the candidate passage with the 'location', 'person', or 'organization' tags. The top ranked passage having these tags is selected to be the final selected passage. We then use n-gram similarity approach. Previous ResPubliQA results have shown that there is a strong correlation between terms in question and answer passages and n-gram similarity can be used to exploit this correlation for achieving better results [9]. We compute the score for a passage by summing all possible x-gram matches where x is less than or equal to the number of terms in the question. We do this summation till the end of question is reached. Then, we divide the above generated sum by the maximum possible n-gram score (i.e. n*(n+1)/2). The final score is a fraction between 0 and 1.

Finally, we set a threshold value for the score match to be 0.15. If n-gram match score exceeds 0.15, we answer the question. Otherwise, we choose not to answer the question.

$$\text{n-gram sum} = \sum_{x=1}^{n} \sum_{y=x}^{n} (m*1)$$

where y is the x-gram under consideration, m = 1 if there is a match, otherwise m = 0

For maximum possible n-gram sum, in the above formula, m = 1 for all values of x and y as all x-grams match. So, maximum possible n-gram sum = (n*(n+1)/2)

$$\text{n-gram score} = \text{n-gram sum} / \text{maximum possible n-gram sum}$$
$$= \text{n-gram sum} / (n*(n+1)/2)$$

For the question, 'Who is the father of Tom Dickens?', and a candidate answer 'Ronald Dickens is the father of Tom Dickens', n-gram sum is equal to the number of x-gram matches where x is a number from 1 to n and n is the number of terms in the question which is 7 in this case. So, n-gram sum here is 21 (6 1-gram, 5 2-gram, 4 3-gram, 3 4-gram, 2 5-gram, and 1 6-gram matches). Maximum possible n-gram sum is 28 (7 1-grams, 6 2-grams, 5 3-grams, 4 4-grams, 3 5-grams, 2 6-grams, and 1 7-gram). So, n-gram score is 21/28 = 0.75. Hence, there is a high chance of this candidate answer to be correct.

4 Results

We participated in the monolingual en-en task of ResPubliQA 2010. Of the two runs submitted, the queries in the 1st run are created after treating the questions with the POS Tagger and removing the stop-words, and having important terms (nouns, pronouns, adjectives, adverbs, prepositions, non-auxiliary verbs) in the query. These queries are passed into the Indri Search Engine and we get a ranked list of top 100 passages (step 3 in Fig. 2). The top ranked passage is selected as the answer in this run. The 2nd run includes our implementation of techniques like Query Expansion and n-gram similarity matching. We have also included Named Entity Recognition. In Fig. 2, step 8 results in the final output of the 2nd run wherein a decision is made about whether to answer the question or not. In the case of unanswered questions, we submitted the top ranked passage in step 7 as a candidate answer. Of the 31 unanswered questions in the 2nd run, the candidates of 14 were incorrect. So, proportion of the answers correctly discarded is 14/31 = 0.45. The table below summarizes the results of our 2 runs.

Table 1: Runs of the system

Run	c@1	Correct	Incorrect	Unanswered	Accuracy
1	0.64	127	73	0	0.64
2	0.68	117	52	31	0.67

5 Conclusion

In this paper, we have discussed our system that has participated in ResPubliQA 2010. The results show that our implementations of Query Expansion and n-gram similarity help the system to achieve a better score. The re-indexing done in step 4 (Fig. 2) reduces the retrieval time which can be an important factor when the corpus size is large. Also, the fact that our c@1 score is higher than the accuracy (run2) shows that the use of our non-answering criteria is justified. However, there is scope for further experiments on large sets of data to decide on the threshold value (in step 8 of Fig. 2) for the n-gram similarity score.

References

1. Dang, H.T., Kelly, D., Lin, J.: Overview of the TREC 2007 Question Answering Track. In: Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007, Gaithersburg, Maryland, USA (2007)
2. Pasca, M.: Open-Domain Question Answering from Large Text Collections. Chicago University Press (2003)
3. The Lemur Project, http://lemurproject.org/
4. Indri Retrieval Model Overview, http://ciir.cs.umass.edu/~metzler/indriretmodel.html
5. Indri Query Language Quick Reference, http://ciir.cs.umass.edu/~metzler/indriquerylang.html
6. Stanford Log-linear Part-of-Speech Tagger, http://nlp.stanford.edu/software/tagger.shtml
7. Penas, A., Forner, P., Sutcliffe, R., Rodrigo, A., Forascu, C., Alegria, I., Giampiccolo, D., Moreau, N., Osenova, P.: Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In: Working Notes for the CLEF 2009 Workshop, Corfu, Greece (2009)
8. Stanford Named Entity Recognizer, http://nlp.stanford.edu/software/CRF-NER.shtml
9. Correa, S., Buscaldi, D., Rosso, P.: NLEL-MAAT at CLEF-ResPubliQA. In: Working Notes for the CLEF 2009 Workshop, Corfu, Greece (2009)