Team 7: QA4MRE Alzheimer’s Task

Jeff Gee, Zhiyu Li, Laleh Roostapour, Adam Rosini, Zhengxiong Zhang

Abstract:

This project was created to provide an opportunity for our group to expand upon the the existing general QA system built by Patel, Yang, Nyberg and Mitamura for the QA4MRE 2013 tasks. Our goal was to improve the performance of system in relation specifically to the Alzheimers task. Our group was initially given 4 gold standard (annotated) documents from the 2012 Alzheimer’s task and was later given a “blind” test set in which the correct answers were not marked. Despite not having the gold standard for the second set of data, our final submission runs are based upon error analysis from both sets of data. Through our 5 submissions, we used both PMI and similarity scores. Although our system generally scores higher using the PMI scoring, the runtime is significantly greater using the PMI scorer rather than the similarity scorer. Our groups best c@1 score for the gold standard data in our submission runs is 0.45. Our groups best c@1 score using our own correct answer annotations on the blind set was 0.41, although this score is not verified.

1. Introduction

Our system was extended from an existing general QA answer system, built to complete all tasks in QA4MRE. With an existing QA architecture in place, our group was able to add additional features, as well as modify some existing ones, in order to optimize the abilities of the system given the time constraints of the project.

While there are many ways to improve the baseline system provided, our team tried to choose enhancements which would be able to be completed in a time effective manner. Our team also tried throughout our development to first derive common mistakes from concrete error analysis to focus on individually. This proved difficult at times due to the differences in errors based on the scoring method used and current state of the pipeline. Our task specifically related to document’s concerning Alzheimer’s, however our modifications to the system would most likely be applicable to any task. We initially tried to make specific modifications to the pipeline based upon our primary analysis of the provided system and documents; however these decisions proved to have minimal impact on our system’s performance, due to the fact they were not derived from *error analysis.*

This paper presents an outline of the modifications, reasoning and performance changes we made to the baseline system provided.

2. Previous Work

The following section provides an outline of the components from the baseline system which are included in our system as well without major modification. Please see *"Building an Optimal Question Answering System Automatically using Configuration Space Exploration (CSE) for QA4MRE 2013 Tasks"* (Patel, Yang, Nyberg, Mitamura) for more information concerning these components.

2.1 Data Readers / Cas Consumer's

2.1.1 QA4MRETestDocReader

Reads in documents containing questions and answers.

2.1.2 QA4MREXMITestDocReader

.Reads XMIs into the CAS

2.1.3 QA4MRECasConsumer

Writes the the CAS out to an XMI file.

2.3 Annotators

2.3.1 Text Segmenter

Remove the unnecessary document segments such as "References", "Figure" and "Table" headers.

2.3.2 Sentence Extractor

Extracts sentences from the text using StanfordCoreNLP.

2.3.3 Noise Filter

Removes sentences that probabilistically do not contain relevant information

2.3.4 Named Entity Recognizer

Extracts named entity using the ABNER library.

2.3.5 Noun Phrase Extractor

Provides part of speech tagging on various tokens.

2.4 Document Indexer

Uses a Solr server to store data and construct queries from which to extract candidate sentences.

3. System Architecture

This sections outlines the components added to or significantly modified from the baseline system provided.

3.1 Sentence Expansion

The baseline code only scored answers based on their relevance to individual sentence. However, in many cases answers to sentences are stretched across multiple sentence. Therefore, in our system, we attempted to include contextual named entities and noun phrases from surrounding sentences.

3.2 Answer Pruning

3.2.1 Part of Speech Pruner

Based on the structure of any given sentence, we can rule out certain answers based on their part-of-speech tags. For example, for "How many...?"-type questions, we can rule out all answers that don't correspond to numeric quantities.

3.2.2 Co-Occurrence Pruner

In some cases words in a question have very little relation to the answers. For example, if a question is asking about a "gene", and one of the answers happens to be "cheeseburger", we can almost immediately rule it out, as "gene", and "cheeseburger", have very little semantic similarity.

3.3 Candidate Sentence Search Strategy

The baseline matcher forms a query for each question and picks the top k candidate sentences from the result. The intuition behind the matcher is to capture the notion of multiple-choice answers. This matcher forms the solr query in the form of a question and an answer choice. Therefore, instead of having 1 query for each question, it creates m query for each question where m is the number of multiple-choice answers for that question. As we expected, for some of the questions we get the better candidate sentences which leads to a better result.

3.4 Candidate Answer Scoring

The input of this module is K candidate sentences. For each of the question and its answer set, we calculate the score for each answer in the answer set with respect to each candidate sentence. Different methods can be used in the score calculating. In our pipeline, three strategies were explored:

1. Number of Noun-phrases/Named-entities matched in answer choice and candidate sentence
2. Number of Synonyms matched in answer choice and candidate sentence

3. Point-wise Mutual Information (PMI) score between Noun-phrases/Named-entities in answer choice and candidate sentence based on the background corpus.

3.5 Answer Choice Selection Strategy

Our pipeline uses a single vote voting system to select the best answer given the best K candidate sentence scores for each answer. Each candidate sentence has 1 vote and the answer choice which receives the most votes is selected as the winner. The winning answer must receive a number of votes as least as high as the minimum threshold. Otherwise, no answer will be selected. We added some functionality to the voting logic such that whenever 'None of the above' was an answer choice and the voting system came up with null as answer (our system was not confident enough to provide an answer), we changed the answer to 'None of the above.'

4. Submission Runs

Through our runs, we used both PMI and similarity scoring, a variety of K values for K candidates throughout our pipeline as well as different levels of pruning and different scoring mechanisms. The mechanisms used in the indexing and search phase for each run are outlined below:

4.1 Run 1

Search Phase:

- QuestionAnswerMoreCandSentSimilarityMatcher
- AnswerChoiceMoreCandAnsPMIScorer
- AnswerSelectionPMIByKCandVoting

4.2 Run 2

Indexing Phase:

- SentenceCombiner

Search Phase:

- POSPruner
- QuestionAnswerCandSentSimilarityMatcher
- AnswerChoiceCandAnsSimilarityScorer
- AnswerSelectionByKCandVoting

4.3 Run 3

Indexing Phase:

- SentenceCombiner

Search Phase:

- POSPruner
- CooccurrencePruner
- QuestionAnswerCandSentSimilarityMatcher
- AnswerChoiceCandAnsSimilarityScorer
- AnswerSelectionByKCandVoting

4.4 Run 4

Indexing Phase:

- SentenceCombiner

Search Phase:

- QuestionAnswerMoreCandSentSimilarityMatcher
- AnswerChoiceMoreCandAnsPMIScorer
- AnswerSelectionPMIByKCandVoting

4.5 Run 5

Indexing Phase:

- Sentence Combiner

Search Phase:

- POSPruner
- CoocurrencePruner
- QuestionAnswerMoreCandSentSimilarityMatcher
- AnswerChoiceCandAnsPMIScorer
- AnswerSelectionPMIByKCandVoting

5. Results and Error Analysis

With the original test data, our 5 submissions received a maximum average c@1 score of 0.4375. On the blind set, our system received a maximum average c@1 score of 0.41, based on our groups own annotations.

Our logic for choosing the ‘None of the above’ option did not have sufficient testing time and thus needs more tuning so that the option is chosen when we intended it to be. Having optimized ‘None of the above’ logic could make a large difference in the actual c@1 scores for our submissions on the blind set considering the number of questions with a ‘None of the above’ answer.

There are also many common errors in our system which we were already aware of not being handled. For example yes/no/maybe questions, in additions to questions that need numerical answers would require some additional logic or annotations, however our system does not provide these features. These are features that would provide a small, but definite improvement in system performance however due to time limitations were ruled out of our plan of action.

6. Conclusion and Future Work

The change to our pipeline with the biggest individual impact was modifying the Solr query construction for retrieving candidate sentences to include both the question and the answer. After doing this, we were able to make improvement to the other areas of the system much more quickly as the information presented was much more relevant.

Through the course of building our system, we have discovered a few limitations by using a Solr server to retrieve candidate sentences from. One of the observations we made when evaluating question was the number of distinct question types. For example, in some cases we encountered “Yes/No/Maybe” type questions. In these cases, the approach to answering must be different, as the answer text itself provide little to no information when measured against the original article. The same is true for the “True/False” questions.

Another type of question that we encountered that we could not address was the question that requires additional reasoning, such as addition and subtraction. For example, when a question asks “how many examples of… were discussed above?”, we would need to count the number of statements that constitute an example.

Additional expansion techniques would also be welcomed. One such example is identifying common acronyms in a given domain, and expanding based on those. Additionally, we believe more advanced heuristically could help improve the system. Dependency Parsing could be better incorporated, likely using representation in a graph database.

Finally, we believe that would improve system is being able normalize different scoring strategies. One of the largest drawbacks of the pipeline is each component’s relative dependence on the other. For example, PMI scoring scores documents on a scale much different that a typical zero to one similarity. Because of this, the voting process must be adjusted (especially if thresholds are involved). By normalizing scoring, it would be possible to build ensembles of scoring systems, providing the weighted benefits of different scoring strategies.

We would like to have more time to analyze the system, the QA system landscape and the available services and annotators to identify Alzheimer task specific features to our system. In our limited amount of time, it was difficult to identify errors related to the Alzheimer task in specific.

Acknowledgements

We would like to thank Professor Nyberg, as well as the 11-791 TA's for not only providing us with the opportunity to work on this project, but also to the learning tools and resources for successfully building a system for the QA4MRE task.

References

Alkesh Patel, Zi Yang, Eric Nyberg, Teruko Mitamura . "Building an Optimal Question Answering System Automatically using Configuration Space Exploration (CSE) for QA4MRE 2013 Tasks". 2013.