Building a Question Answering System for the QA4MRE 2013 Alzheimer Task

Kartik Goyal, Haoyu Wang, Wenlong Tu, Lars Mahler, Kuo Liu
Carnegie Mellon University, Pittsburgh, PA, 15213
{Kartikgo, haoyuw, wenlongt, lmahler, kuol}@andrew.cmu.edu

Abstract

As an important branch of natural language processing, question answering systems have been discussed more and more frequently in recent years. Interest in question answering systems is increasing because of their potential to aid research and commercial projects. In this paper, we use the UIMA framework for building our question answering system for biomedical purposes and applied our system on the Alzheimer task of QA4MRE at CLEF 2013. Since this is a blind test, we make use of the corresponding test data for 2012 to test our system. Experiment results show that our system can achieve c@1 of 42.5% on the 12-test-alzheimer dataset. Hopefully we can do a pretty good job on the blind test dataset.

Keywords: Biomedical question answer, UIMA

1. Introduction

Text processing components like parsing, part of speech tagging etc., are improving every year, which in turn leads to better QA systems. These systems can either be domain specific or general in nature. General QA systems are far more difficult to build than the domain specific ones because of the inherent ambiguity and flexibility of natural languages. As mentioned earlier, we are focusing on building a domain specific system with a closed world knowledge-base/background corpus. Moreover, seeing the success of IBM Watson, a general framework for the data process pipeline called UIMA gives us a good choice to build a QA system in limited time with a lot of contributors.

In this paper, we extend the initial framework of the given baseline to get an improvement on the Alzheimer task. To achieve better results, we decided to use several existing open source Natural Language Processing (NLP) tools and combine them in an efficient manner in our pipeline. The crux of our approach is to identify the tools that would help us get a good performance and then integrate them in an intelligent manner for optimum performance.

The outline of this paper is as follows. In Section 2, we review the baseline pipeline which was provided at the beginning of the project. In Section 3, we introduce the framework of our system, focusing on the new elements we introduce. In Section 4, we show the experimental results of our system on the 12-test-alzheimer dataset (Since the final test set is blind, we have no idea about the performance of our system on the final dataset). In Section 5, we conclude our work and provide a general idea about future work directions.

2. Previous Work

For the convenience and time saving purpose, a baseline QA system which is also focused on the Alzheimer task is given[1]. This baseline pipeline is the implementation of one published paper on CLEF 2013. Since our pipeline is based on this framework, we will give a general review about it before stepping into the next part.

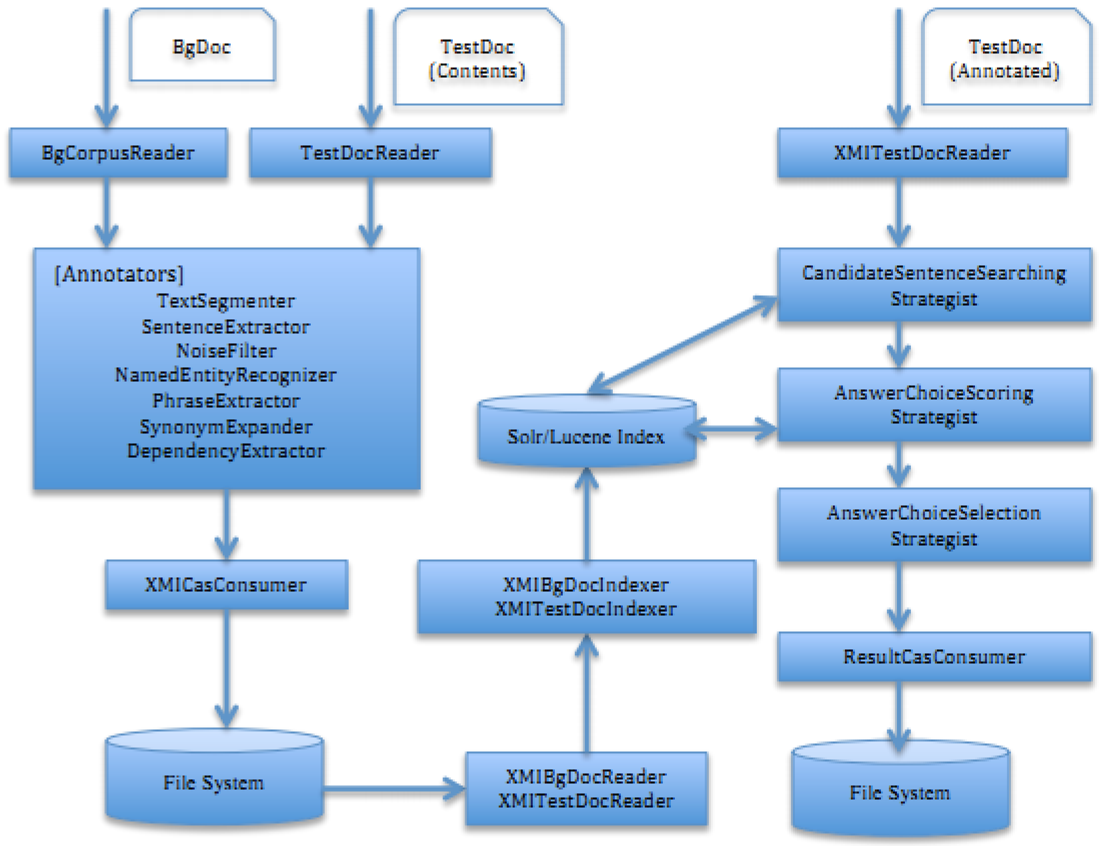

Fig 1. Baseline pipeline

The baseline system can be generally separated into two major parts. The first part is all about making useful annotations, like named entity, phrase annotations etc., on the elements in the document containing information, question and answers. In order to index the annotated elements faster, a popular search engine called Solr is used. The second part is mainly focused on processing the test documents to calculate score for fitness of all the candidate answers to the corresponding questions. In this part, some basic methods such as PMI and Voting are used.

Generally speaking, the baseline pipeline is quite intuitive to understand, and it is similar to traditional information retrieval systems.

3. System Architecture

As discussed earlier, our system is largely based upon the baseline system[1], but we incorporated several changes which we believed would improve performance. In this part, we will mainly focus on the major changes we made on the original system.

The original system extracted key information in the sentences such as named entities and noun phrases, and then use these to compute overlap scores or PMI based scores for different candidate answers. We made the following enhancements by adding more features to improve the scoring of candidate answers.

The steps on the pipeline are:

1) Index the background corpus using Solr.

2) Annotate the input document with several NLP features including tokenization, lemmatization, part of speech tagging, finding Noun Phrases and Named Entities for the input text, questions and candidate answers.

3) **Enhancement:** Perform **entity coreference** on the information in the input document, and build coreference chains. This procedure involves making a chain for each Noun Phrase/Named Entity with its references in the input passage being the chain's nodes. This is important as we will obtain more information/sentences for each noun phrase in contention. Consider following sentence: "Testosterone is a hormone found in males. It is responsible for various masculine features." With coreference information, we know that 'it' in the second sentence refers to testosterone. Hence, while considering candidate answers for the question "Which hormone is responsible for masculine features?", our system can be more confident about the answer 'testosterone'.

4) **Enhancement:** Perform **symbol expansion** and **acronym expansion**. Symbol expansion replaces non-English characters with their English equivalents (for example: "Aß" expands to "Abeta"), enabling symbols and their English counterparts to be used in finding candidate sentences. Acronym expansion replaces acronyms with their full text (ex: "AD" expands to "Alzheimer's Disease"), enabling acronyms and their expansions to be used in finding candidate sentences.

5) **Enhancement:** Updated Solr indices and queries. The baseline system was indexed/queried using Named Entities and Noun Phrases. We added other fields which are raw text ('text'), coreference, and synonyms. The fields are intuitive and our major addition was the 'coreference' field for each sentence which contains a list of coreference phrases in the whole document for each 'phrase' in the corresponding sentence.

6) **Enhancement:** Since we added quite a lot of additional information to our Solr queries, our recall increased, but our precision decreased due to noise (less-important features drowning out important features). As a result, we tuned our Solr queries to keep all of the new sources mentioned above, but to give higher weight to Named Entities and Noun Phrases. We performed this tuning using the 12-test-alzheimer dataset.

7) **Enhancement:** Find candidate sentences using 3 different Solr query types: queries based on **question** text (similar to baseline implementation), queries based on **answer** text, and queries based on **question+answer** combinations*. Hence, we will get 3 sets of candidate sentences. The strategy for constructing our queries is based on different fields and the weights for different fields are different (i.e. we give more weights to fields which are important such as named entity and less weights to fields which are less important such as coreference).
*Note: for queries based on question+answer combinations, we replace the target in the question (the noun phrase attached to the WH-word in the dependency parse) with the candidate answer. For example, for the question 'Which hormone is responsible for masculine features?' and candidate answer 'testosterone', we issue the query based on the text and noun phrases in "Testosterone is responsible for masculine features"

8) **Enhancement:** Next, we score each candidate answer based upon the candidate sentences we obtained from the queries, and other useful information about the answer itself. As mentioned above, we had 3 sets of candidate sentences from 3 different queries: the **question** query, the **answer** query, and the **question+answer** query. For each set of candidate sentences, we compute the similarity scores (PMI of NER, PMI of NP, PMI of Coreference, NER and NP overlap, and Q&A overlap) between each possible answer and the set of candidate sentences. As a result, there are 3 x 5 = 15 possible scores that we were able to use to score each candidate answer, however some of them took too long to compute, and so we ended up with a final set of 5 scores that we used. These are:

- PMI of NP: Question Candidate Sentences + Answers
- PMI of NER: Question Candidate Sentences + Answers
- PMI of Coreference: Question Candidate Sentences + Answers
- Overlap of NP, NER: Question Candidate Sentences + Answers
- PMI of NP, NER: Question + Answers

9) **Enhancement:** In addition, for each candidate answer, we added 2 additional flags to check that the question cardinality matched the answer cardinality (i.e. if the question is asking about an entity that is singular, is the answer also singular?), and to check that the question target entity type matches the answer entity type (i.e. if the question appears to be asking for something that is quantifiable / integral, does the answer contain an integer?). These flags were used in the resulting scoring function: if either flag was a mismatch (i.e. question != answer), then we penalized the candidate answer.

10) **Enhancement:** From the above enhancements, we ended up with many individual scores (5 similarity scores + 2 flags), which we then combined into a final score for each candidate answer. These similarity scores were combined by a linear function, and the flags were used to penalize in the case of a mismatch. We tuned the weights using the 12-test-alzheimer data set.

As a result of these changes, we made several modifications to the original (baseline) type system. For the sake of space, we will not show the entire type system, but the changes can be seen in the following figures:

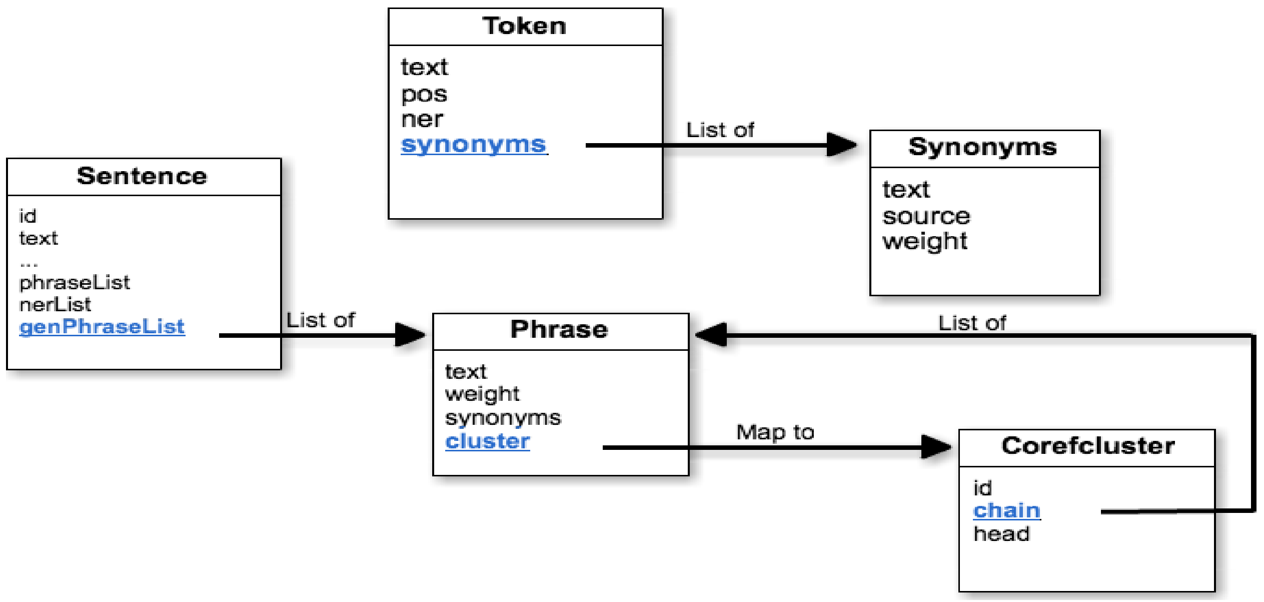

Fig 1. Changes supporting: symbol expansion, synonym expansion, addition of coreferences

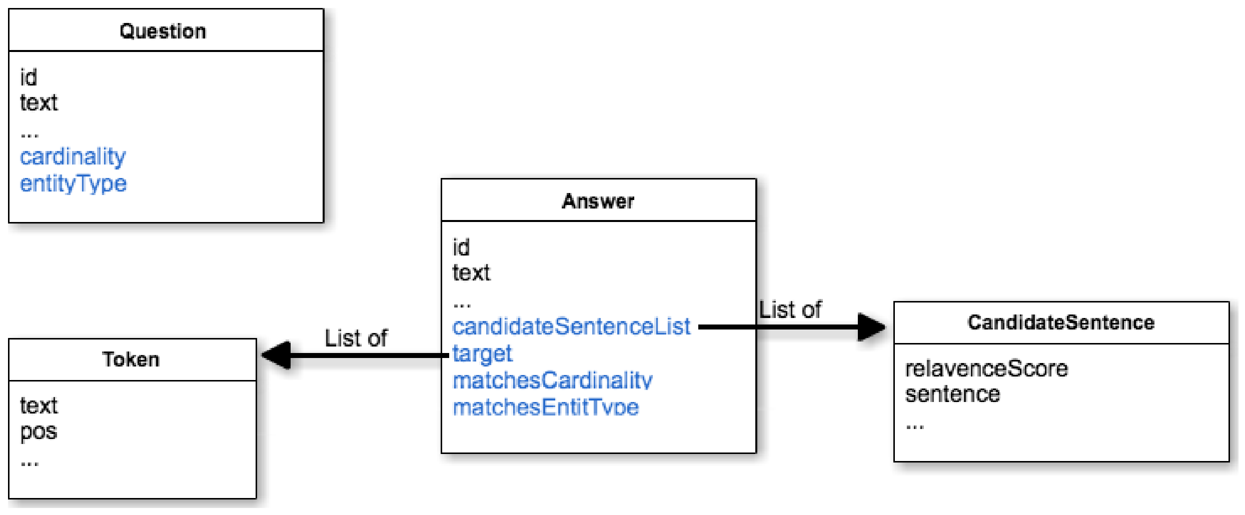

Fig 2. Changes supporting: symbol expansion, synonym expansion, addition of coreferences

4. Experiments & Results

Our experiments were based on the 12-test-alzheimer dataset, and results for milestones M1, M2, and M3 are listed as follows. Since the analysis on the results for milestones M1 and M2 have been explained in detail in class, we omit them here to avoid redundancy. Error analyses for M1 and M2 can be found in the "documentation" folder in our project repository.

Version	doc1	doc2	doc3	doc4	mean
M1	0.4	0.5	0.4	0.2	.375
M2-v2	0.4	0.6	0.4	0.3	.425
M3-before tuning	0.4	0.4	0.1	0.6	.375
M3-after tuning-v1	0.2	0.7	0.3	0.3	.375
M3-after tuning-v2	0.3	0.3	0.7	0.3	.400
M3-after tuning-v3	0.2	0.7	0.3	0.3	.375

Table 1. c@1 for all three milestones on 12-test-alzheimer

As described above, we built a system that had many individual scores - 5 similarity scores which we combined in a linear function, and 2 flags which were used to penalize in the case of a mismatch. For M3, we built a script that looped through 10^5 different parameter settings to try to find the best combination of weights. This script identified several different parameter configurations (see A, B, and C below) that we expected to have strong performance on the 12-test-alzheimer dataset. However, the script was separate from the SimpleQuestionRunCPE pipeline, and when used these settings in the SimpleQuestionRunCPE pipeline, we found that our results were slightly worse than M2. As a result, our submission contains 7 output files: 3 output files associated with M3 configurations (A, B, and C below), and 1 output file associated with our best M2 configuration (D below), and 3 more output files associated with M3 configurations, but NO penalties for question / answer mismatches (A', B', and C' - no results shown below).

Param Setting	**Results on 12-test-alzheimer dataset**	**Weight: PMI Score for QC+A (NP)**	**Weight: PMI Score for QC+A (NER)**	**Weight: PMI Score for QC+A (Coref)**	**Weight: NP & NE Overlap**	**Weight: PMI Score for Q+A (NP,NER)**	**Penalty: Card. mismatch**	**Penalty: Entity Type mismatch**
A	.375	5	1	6	7	0	.8	.8
B	.400	4	2	5	5	0	1.0	1.0
C	.375	5	0	6	6	0	.8	.8
D	.425	/	/	/	/	/	/	/

Table 2. Parameter settings and the corresponding results (/ denotes unused parameters since they're about M2)

Even though configurations A, B, and C do not achieve results that are as good as M2, they may work on the blind test set. As a result, we still keep these settings of parameters to see if they will work on the new test dataset.

5. Conclusion & Future Work

In this paper, we provide a framework for building a QA system aimed at Alzheimer task. Experiment results show that we can get c@1 of over 42.5% on the 12-test-alzheimer test dataset.

Future work includes code reconstruction, trying other directions to improve the performance and beat over-fitting.

Acknowledgements

We thank the TAs who gave us much necessary guidance. We also thank our respectable Prof. Eric Nyberg who gave us excellent lectures and knowledge to build large scale systems.

References

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