UW Question Answering 2013

|  |  |  |
| --- | --- | --- |
| **Marie-Renée Arend** | **Josh Cason** | **Anthony Gentile** |
| University of Washington | University of Washington | University of Washington |
| rcarend@uw.edu | casonj@uw.edu | agentile@uw.edu |

Abstract

stub

1 Introduction

As a baseline, we developed a redundancy based QA system which draws on web results from the Bing search engine. We then built upon that baseline to treat the question-answering process as a classification task, using Mallet to learn features and weights. In addition, we examined the effect of changes such as query word sense disambiguation.

[anything else to add here?]

2 Approach

The three main components of our redundancy baseline are query processing, ngram generation and filtering, and answer projection.

[do we want to put a pipeline diagram or something here?]

2.1 Query Modification

Often when searching the web or a document collection, it is difficult to get the correct results because of mismatch between the wording of a user’s question and the wording of potential answers. To alleviate this to some degree, we incorporate a query processing unit. In our baseline iteration of our system, this module only performed the most basic processing tasks, such as stemming, removing stopwords and adding the target if not already present in the question.

In our current iteration, we attempted to use our own version of the modified Lesk algorithm as described by Huang et al. (2008). Instead of just disambiguating the head word in the query, we chose to disambiguate every word in the query that can be found in WordNet. Once the best sense for these words was generated by the algorithm, we wanted to use those best senses to find synonyms to add to the original query.

The word sense disambiguation and synonym-generating code did not work as intended for this version of our QA system. Perhaps the most troublesome problem with the code’s results was that the words were not properly disambiguated, leading to a large number of both relevant and irrelevant synonyms being added to the reformed query. This created too much noise in the query and, as a result, many questions that were answered in our baseline system were not answered this time around. A second major problem is that the disambiguation algorithm takes too much time to process each query.

To address the issues that arose with this latest version of our query expansion module, we are working on introducing some major improvements, such as increasing the efficiency and accuracy of our version of the modified Lesk algorithm and reducing the number of synonyms added to the final reformulated query to cut down on the amount of noise generated. If we are unable to improve the Lesk algorithm to a satisfactory extent, we plan to try using a version of Resnik’s word sense disambiguation algorithm (Resnik, 1995).

2.2 N-gram Generation and Filtering

The next module generates ngrams from the web results and filters them following some of the techniques mentioned in (Lin, 2007). One hundred results are gleamed from the Bing API search, tokenized, and, from the title and snippet, unigrams, bigrams, trigrams, and quadrigrams are counted. Multi-token ngrams are then filtered according to the following criteria. An ngram is removed from the ranking if the ngram begins or ends with a stopword, contains words from the question, or has other problems like if there is punctuation leftover from tokenizing.

Ngrams are then reranked with the following formula.

The score, *Sc*, of a candidate answer in increased by the sum of the scores of its unigram tokens. This combats the tendency to reward shorter answers due to their natural higher frequency.

2.4 Answer Projection

The top twenty answer candidates are chosen and are used to project the answer onto the AQUAINT corpus (Graff, 2002). AQUAINT is used for evaluating QA systems, and it is necessary to find a document in it to support our web retrieved answer. We use a standard IR system Lucene (Hatcher et al., 2004) in order to retrieve the most relevant documents.

Several subtasks in these modules are supported by the open-source Natural Language Toolkit (Bird, et al., 2009).

3 Results

|  |  |
| --- | --- |
| Lenient MRR | 0.0455 |
| Strict MRR | 0.0039 |

Table 1: MRR scores

Our results were not very encouraging. However, given further refinements, we can see it improving. A major practical improvement we hope to make is to implement some caching. Currently testing code is time consuming given network latency and the massive size of the AQUAINT corpus. Caching results will make it faster for us to measure the effectiveness of alternative configurations.

Another option is to add *idf* scoring of answer candidates as discussed in (Lin, 2007).

4 Conclusion

stub

References

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. O'Reilly Media.

Graff, D. (Ed.). (2002). *The AQUAINT corpus of English news text*. Linguistic Data Consortium.

Lin, J. (2007). An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems (TOIS)*,*25*(2), 6.

Hatcher, E., Gospodnetic, O., & McCandless, M. (2004). Lucene in action.

Resnik, Philip. (1995). Disambiguating Noun Groupings with Respect to WordNet Senses. *Third Workshop on Very Large Corpora*. Retrieved from

<http://acl.ldc.upenn.edu/W/W95/W95-0105.pdf>