UW Question Answering 2013

Marie-Renee Arend
University of Washington
rcarend@uw.edu

Josh Cason
University of Washington
casonj@uw.edu

Anthony Gentile
University of Washington
agentile@uw.edu

Abstract

stub

1 Introduction

stub

2 System Overview

stub

3 Approach

As a baseline, we developed a redundancy based QA system which draws on web results from the Bing search engine.

3.1 Redundancy QA

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

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. Currently, the module only removes stopwords and stems the tokens. However, in the next iteration we may include an expansion of query terms to include, for example, synonyms.

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.

$$S_c = S_c + \sum_{t \in c} S_t$$

The score, S_c, 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.

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).

4 Results

Lenient MRR	0.0455
Strict MRR	0.0039

5 Discussion

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).

6 Conclusion

stub

Acknowledgments

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.