DeanQA : An Advanced Question Answering System

Yann Le Gall
University of Pittsburgh
Department of Computer Science
6503 Sennot Square
`ylegall@cs.pitt.edu`

Eric Heim
University of Pittsburgh
Department of Computer Science
5324 Sennot Square
`eth13@cs.pitt.edu`

Alex Conrad
University of Pittsburgh
Department of Computer Science
5422 Sennot Square
`conrada@cs.pitt.edu`

1 Introduction

In this paper, we design, implement, and evaluate a question answering (QA) system, *DeanQA* . In our approach, we use several different strategies from different domains to select potential answers. Then, we employ a majority voting scheme to combine the results.

To train and test our QA system, we used the "CBC Reading Comprehension Corpus". This corpus is composed of 125 news stories, each accompanied by a set of 6-10 factoid questions (e.g. questions that begin with "Who", "When", "Where", etc.).

All stories in the dataset were been split into sentences (one sentence per line) using the MXTERMINATOR sentence splitter developed by Adwait Ratnaparkhi. Paragraphs from the original story are separated by an empty line.

The rest of this paper is organized as follows: in §2 we describe the design and implementation of our QA system and each of its sub-components. Next, in §3 we evaluate our system on the test dataset and present the performance results. Finally, we conclude in §4.

2 Implementation

In this section we discuss the design and implementation of *DeanQA* . First, we explain the general, overall structure. Then we give a detailed description of the sub-components (*AnswerFinders*) of the system. Finally, we present our technique for combining the potential answers from each sub-component.

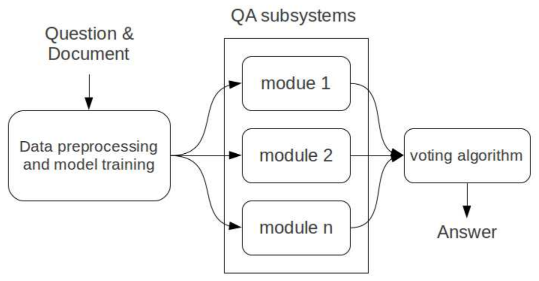

Figure 1: The *DeanQA* system model.

Figure 1 shows the general design of *DeanQA* . The system takes processes a document and a set of questions as input. At this point, several preprocessing steps might be performed, including the removal of stop words, coreference resolution, etc. Next, the processed document and the current question are passed to each of the AnswerFinders. Each of these AnswerFinders implements a different strategy for finding answers to the question, and they each return a collection of `<answer,confidence>` pairs. Finally, the potential answers are combineded in a voting algorithm that gives different weights to the potential answers based on the confidence and the question type.

In the next few subsections, we give more detailed descriptions of the implementation of the different AnswerFinders and the strategies that they use.

2.1 Bag-of-Words

The Bag-of-words (BOW) model is a simple model that provides relatively good performance in our system. In our domain, BOW treats each question as a set of words. Then, for each sentence in the document, we count the number of matching words. The probability that a sentence is the correct answer is estimated as the ratio of the number of matching words to the total possible number of matches.

2.2 Bag-of-NGrams

Building on the idea of Bag-of-words, the Bag-of-NGrams (BONGrams) is a similar model which treats each question not only as a set of words (unigrams), but as a set of all possible n-grams, $1 \leq n \leq k$, where k is the number of words in the question. The probability that a sentence is a correct answer is estimated as the ratio of the number of matching ngrams to the total number of ngrams generated from the question.

2.3 Linguistic Rule-based Strategies

We also implemented a rule-based AnswerFinder based on the previous work of (Riloff and Thelen, 2000). This system applies different rules based on the question type. A preliminary score is calculated for each line based on the BOW model. Then, different amounts of points are awarded for each satisfied rule. For example, if the question type is WHEN, then a few points are awarded to sentences that contain related words such as “time”, “during”, “before”, “after”, etc. More points would be awarded if the sentence contains a month, day, year, or parts of a date, such as “January 10^{th}, 1985.” Finally, the sentence with the best score is elected to be the most likely answer to the question.

2.4 SVM Classification

The next component of the *DeanQA* system takes a Machine Learning approach that barrows heavily from the work done by (Ng et al., 2000). In their paper, the authors extracted features from a training set of news documents and corresponding questions about the documents. Then, they trained a C5 decision tree classifier, which is a more recent version of the C4.5 (Quinlan, 1993) classifier, in order to predict which sentence answers a question. Each data point represents a sentence-question pair with a label being positive (1) for sentences that answer the paired question or negative (-1) for sentences that do not.

Each feature represents some characteristic of the sentence, the question, or both. For example, the Diff-from-Max-Word-Match (DMWM) gives a score based on how many lemmatized words are in both the sentence and question. This is very similar to the bag of words component. Other features take advantage of named entity recognition, whether the sentence is the title or dateline, or keywords that were learned to be commonly present in sentences that answer specific question types. All of the features from (Ng et al., 2000) were used except for the “keywords in questions” features, as well as the coreference information to boost the named entity features. For a more detailed explanation of these features, see their work. We used the Stanford CoreNLP library for part of speech tagging (Toutanova and Manning, 2000) and named entity recognition (Finkel et al., 2005) in the feature extraction process. In addition to the features mentioned previously, we added binary features to describe the question type. More specifically, six binary features were added indicating whether the question for the data point was a “Who”, “What”, “Where”, “When”, “Why”, or “How” question. If none of the features were “true” then the question type could be interpreted as “Other”. In total, 24 features were extracted.

The main difference this component has from (Ng et al., 2000) is that instead of using the C5 classifier, we trained a Support Vector Machine (SVM). We used the WEKA 3.6 (Hall et al.,) Java wrapper class for the LibSVM 3.11 (Chang and Lin, 2001) SVM library to implement this SVM. The radial basis kernel was used with a cost parameter of 4.5 and a gamma parameter of 1/24 (1 over the number of features) after cross validation. This model was trained on the “input” dataset, which had 37 documents and 324 questions. On the “input-test1” set (36 documents, 310 questions) we found that using the WEKA implementation of the C4.5 classifier (called J48, parameters M = 7, N = 17) resulted in having a lower prediction accuracy (131 out of 310) than the SVM model (148 out of 310). This was done by simply picking the sentence-question pair that the models deem as the most likely sentence that answers the question.

2.5 Discourse Component

In addition to the primary AnswerFinders above, we also implemented several secondary AnswerFinders. These secondary AnswerFinders were not intended to be used on their own, but rather to be used to augment the predictions of a primary AnswerFinder(s) by boosting the probabilities of sentences which contained particular kinds of features. This secondary AnswerFinder, the discourse-based answerer, seeks to learn correlations between types of discourse relations and types of questions. For example, a "Synchronous" or "Asynchronous" relation may frequently be part of an answer to a "When" question. Thus, the goal of this module is to boost or diminish the probability that a sentence answers a question given the type of question and the discourse relations in which the sentence participates.

To automatically identify discourse role information, we utilized a discourse parser trained using the Penn Discourse Treebank (PDTB) tagset (Lin et al., 2010). Unlike RST, PDTB annotations are not hierarchically organized throughout the document; rather, PDTB annotations are relatively flat, linking pairs of text spans using local relations. For example, in the following sentence:

"*The Texas oilman has acquired a 26.2% stake valued at more than $1.2 billion in an automotive-lighting company, Koito Manufacturing Co.* But **he has failed to gain any influence at the company.**"

a "Concession" relation would be identified between the italicized and boldfaced portions. The italicized portion creates an expectation which the boldfaced portion denies, while the underlined portion serves as the explicit signal of the concession (Prasad et al., 2008).

A second reason for using this discourse parser is that it has been shown to produce reliable results. Previous work successfully utilized this parser to improve text coherency prediction (Lin et al., 2011).

We utilized the training data to learn correlations between question type and discourse roles in which the answer sentence participated. Because of space constraints, we cannot list these correlation values here. Using these scores, this module adjusts the probabilities of all sentences as answers, giving a large boost if the sentence participates in multiple roles which are strongly correlated with the question type, and a small boost if few or weakly correlated roles are present. For this task, "no-relation" was also considered as a relation type, to capture the correlation between particular question types and answer sentences which participated in no identifiable discourse relations[1].

2.6 Named Entity Recognition Component

The secondary named entity recognition AnswerFinder adjusts sentence probabilities based on the question type and the number of each kind of named entity present within the sentence. For example, when answering a "Where" question, sentences which contain multiple "Location" entities are given a boost in probability. We used the Stanford named entity recognizer library with the pre-trained 7-class model. The seven classes of entities which this model identified were "Time", "Location", "Organization", "Person", "Money", "Percent", and "Date".

As described above, using basic counts of each type of named entity resulted in marginal improvements in accuracy for several question types. We also investigated resolving pronominalized mentions of named entities, but we were not able to use this knowledge to improve performance. Using Stanford CoreNLP, we automatically built coreference chains for each document and replaced each mention of an entity with the most specific mention identified for the entity[2].

2.7 Query Expansion Component via WordNet

Although this did not make it into our final system, we also experimented with question expansion via WordNet synsets. Given all of the words in a question, we performed a WordNet lookup of each word and concatenated onto the question all terms appearing in any of the synsets of the word. We did not make use of part-of-speech information, but rather considered all four parts of speech for which WordNet contains entries for each word in the question. Because our initial investigation of question expansion using WordNet was significantly negative, we did not further explore this path or consider more refined techniques for incorporating WordNet information, though this would be an area for future

[1] discourse code in *cs2731.discourse* package

[2] coreference code in *cs2731.Preprocessor*

work[3].

2.8 Combining Answers with Majority Voting

Previous work by Rotaru and Litman demonstrates that combining the outputs of multiple QA systems can achieve better results than the individual systems alone (Rotaru and Litman, 2005). We incorporate this idea into the design of our QA system by combining the outputs from each of the subsystems described above.

Since some models performed better or worse depending on the question type, we decided to use a different weighting of models for each question type. Since model execution is prohibitively slow, we manually assigned approximate weightings using a limited number of executions guided by our intuitions and empirical insights. For each of the question types, we first investigated each of the primary AnswerFinders (BOW, BONGrams, rule, SVM) in isolation and combination. We also investigated whether each of the secondary AnswerFinders (disc[ourse], ner) and preprocessing techniques (WordNet expansion, pronominal reference resolution) led to improvement or decline when combined with a baseline BOW. Promising techniques were combined for further experimentation. When combining techniques, we leaned towards including as many modules as possible, eliminating modules only if they hurt performance. As mentioned earlier, while our WordNet expansion and pronominal reference resolution techniques were not helpful to our system, all other modules proved beneficial (or at least, not harmful) on one or more question type. We performed this weighting optimization step using the "Test1" dataset. Because the SVM and discourse modules were trained using the "Training" dataset, we feared that optimizing module weightings using this dataset would be too biased towards these modules. These weights are omitted due to space concerns.

3 Evaluation

Tables 1 and 2 show the number of questions the *DeanQA* system answered correctly out of the total number of questions in the first and second test sets respectively. In both cases our system answered well over half of the questions correctly. "When" questions seem to have been the easiest to find the correct answer for and "Other" questions seemed to be the hardest. This can be understood by the fact that "When" questions have answers that include very specific clues to some kind of time or date, but "Other" questions have a much wider array of possible forms to their answers.

Type	# Correct	Percentage
WHEN	25 out of 34	73.53%
WHERE	23 out of 36	63.89%
WHAT	42 out of 80	52.50%
WHY	26 out of 44	59.09%
WHO	26 out of 41	63.41%
HOW	36 out of 68	52.94%
OTHER	3 out of 7	42.86%
TOTAL	181 out of 310	58.39%
NORMALIZED		60.89%

Table 1: Statistics for Test Set #1 by question type.

Type	# Correct	Percentage
WHEN	39 out of 56	69.64%
WHERE	39 out of 64	60.94%
WHAT	87 out of 142	61.27%
WHY	38 out of 66	57.58%
WHO	38 out of 59	64.41%
HOW	56 out of 81	69.14%
OTHER	5 out of 11	45.45%
TOTAL	302 out of 479	63.05%
NORMALIZED		63.83%

Table 2: Statistics for Test Set # 2 by question type.

4 Conclusions

Question-answering is complex task and a cutting-edge research area in natural language processing. Our project attempts to use BOW and SVM backbone models augmented with additional knowledge obtained from discourse parsing, WordNet synsets, and intelligent coreference resolution. More sophicsticated techniques for incorporating this additional knowledge could be used to improve performance in future work.

[3]query expansion code in *cs2731.QuestionExpander*

References

Chih-Chung Chang and Chih-Jen Lin. 2001. Libsvm: a library for support vector machines.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL*, pages 363–370.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A pdtb-styled end-to-end discourse parser. *CoRR*, abs/1011.0835.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. *ACL*.

Hwee Tou Ng, Leong Hwee Teo, Jennifer Lai, and Jennifer Lai Pheng Kwan. 2000. A machine learning approach to answering questions for reading comprehension tests. In *In Proceedings of EMNLP/VLC-2000 at ACL-2000*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. *In Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC). Marrakech, Morocco*.

J. Ross Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems - Volume 6*, ANLP/NAACL-ReadingComp ’00, pages 13–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Rotaru and D.J. Litman. 2005. Improving Question Answering for Reading Comprehension Tests by Combining Multiple Systems. In *Proceedings of the American Association for Artificial Intelligence: Workshop on Question Answering in Restricted Domains*.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *In EMNLP/VLC 2000*, pages 63–70.