Project Report

Bo Ma, Chenyan Xiong, Troy Hua, and Vinay Vemuri

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
mabodx@gmail.com,cx@cs.cmu.edu,vvvemuri1@gmail.com, troy@cs.cmu.edu

Abstract. Machine-reading task is an interesting problem that given a document, and a set of questions, one need to automatically answer these questions using the information provided in the document. In this work, we complete this task using Configuration Space Exploration (CSE) framework, which is an extension of Unstructured Information Management Architecture (UIMA). In this work, we develop several approaches to the given baseline system, including enriched features and relevance models in candidate selection and answer scoring. Further more, we propose a new answer strategy to deal with 'none of the above' answer and decide not to answer when confidence is limited.

1 Introduction

We start from the baseline system. The baseline system divide the task into two main components:

1. Candidate Sentence Scoring. Given a question, sentences in document are ranked based on their relevance to the question.
2. Answer Scoring. The candidate sentences are treated as evidence in the document to find the correct answer. Thus the relevance between answer and candidate sentences are calculated. The final answer is obtained by aggregate scores across all candidate sentences, or majority voting, where each candidate sentence votes for its favorite answer.

These two components directly follows how we human-beings deal with reading task. We first read the document, then for each question, we find out which part of the document are related to the question (candidate sentence scoring), and then we decide which answer is correct based on the related document part (answer scoring). And the baseline system performs well in the training data set.

In this work, we find that there are (luckily) several parts we could improve.

In candidate sentence scoring, the baseline system uses open source search engine's scoring function to calculate the score between each query-candidate pair. However, the search engine is optimized to rank documents, which is usually longer. It is hard to say whether the ranking function in search engine, I.E. language modeling is suitable for our task. Also, it is an unsupervised approach. It is interesting to see whether using the supervision from training data could improve the performances.

In this work, we proposed two approaches to improve candidate sentence scoring:

1. Richer Features: Calculate the similarity between question and candidate sentence using different kind of similarities, at different expression dimensions.
2. Use supervision: Merge the similarity score using cross-validation on training data set.

In Answer Scoring, only the relation between answers and candidate sentences are considered. However, the answer is usually very short, normally only one word. Such one word similarity is not very reliable in lots of text tasks. In this work, we catenate the question to the answer, and calculate answer score by the similarity between candidate sentence and question-answer together.

In Answer Decision, the baseline system will answer all questions, no matter whether confident or not. However, the C@1 evaluation metric favors the system which knows when not to answer a question when it is not confident. Also, for answer 'none of the above', the baseline system treats it the same with other answers. It hurts the performance cause similarity between 'none of the above' with candidate sentences does not matter at all.

In our work, we use two thresholds to solve these problems. Given scores for answers, we will first decide whether to answer it. If the best score of answers is larger than a threshold, then we will answer it. After that, we decide whether we should choose 'none of the above'. If the best score is below a threshold, we will choose 'none of the above'. The two thresholds are trained in training data using cross-validation.

Question Type. In the baseline system, all questions are treated the same. However, different question types may require different answering strategy. In this work, we also make use of several rules to detect the question type, and assign score to answers based on where they could be suitable answers for that question type.

In the rest of this report, we will first discuss our framework and pipeline using CSE. Then we describe the methods used in our system, in section 3. The last part of this report is about submitted runs and individual contributions.

2 Framework and Pipeline Description

We build our UIMA-based pipeline based on the provided baseline code. The framework of the our pipeline is as follow:

- Annotation
  - Annotation from the baseline code
  - Question Type Annotation
- Answer Ranking
  - Candidate Sentence Independent ranking
    - Answer Type Matching Score
    - Answer Question PMI score
  - Candidate Sentence selection and ranking
  - Final Answer Ranking
- "None of the above" and "not answersed" strategy control

And the complete pipeline looks like this:

Here, we introduce the details of each component.

2.1 Syntax Annotation from the baseline code

In this part, we use mostly the original baseline. There are two major differences. First, we do not use a solr indexer for candidate sentence selection, so we remove the solr indexer component. The other difference is that to handle "None of the above" option, for each question, we add a feature showing whether this question has a "None of the above" option.

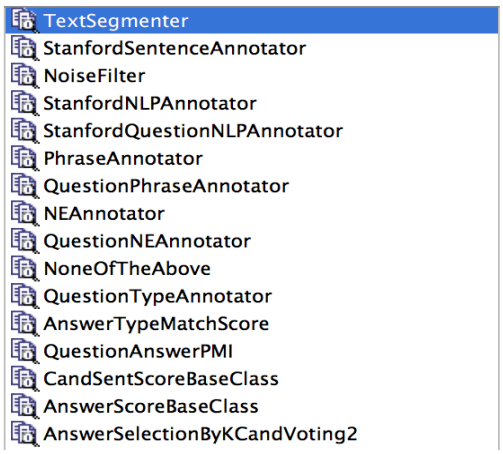

Fig. 1: Complete Pipeline

2.2 Question Type Annotation

We classify all the questions in to 6 categories:

1. QUANTITY: How many/How long...?
2. REASON: Why ...?
3. CHOICE: Name three <proteins> that ...
4. SUBTYPE: What <protein> ...?
5. YES_OR_NO: Is ...? ; Does...?
6. NROMAL: otherwise

We classify the questions for two purposes. First, we have different strategies and scoring weights for different types of questions. Secondly, this component works closely with the **Answer Type matching score** part. For QUANTITY question, we give a higher score for answers with numerical terms. For SUBTYPE and CHOICE questions, we can extract the category of expected answer from the question, like ”protein” in the example, and give a higher score for answers with a matched terms. More details will be discussed in the section of **Answer Type Matching Score**.

2.3 Candidate Sentence Independent ranking

Our answer ranking has two parts. One depends on the candidate sentences selected for this question and answer pair. The other ranking is independent on candidate sentences and is only based on the question and answer. We have two scores in the second category.

Answer Type Matching Score We handle three kinds of Question Type in this component. For YES_OR_NO question, we give a higher score for answers with ”yes” or ”no” terms. For QUANTITY question, we give a higher score for answers with numerical terms. We also handle the numerical terms in English, like ”five days later”. For SUBTYPE question, we have extracted the category terms in Question Type Annotator component and we will look at a taxonomy dictionary we created off line to rank the answer higher with the terms in the corresponding category. For example, if the question ask for ”what protein...” and we have a table with many protein names, we will give the answer with a higher score if the answer contains a protein name. Of course, it is impossible to deal with all the possible categories. We only deal with some frequent categories.

Question Answer PMI In the original baseline system, PMI is calculated based on the answer and candidate sentence. But we find that the correlation between the answers and the question itself is very informative so we add a PMI between the question and answer options.

2.4 Candidate Sentence Selection

In the original baseline, candidate sentence is selected based on question only. In our system, we also consider the answer. So instead of having a fixed set of candidate sentence for each question, we have ranked a list of candidate sentences for each question and answer pair. Figure 2 and Figure 3 illustrate the old pipeline and our new pipeline, respectively.

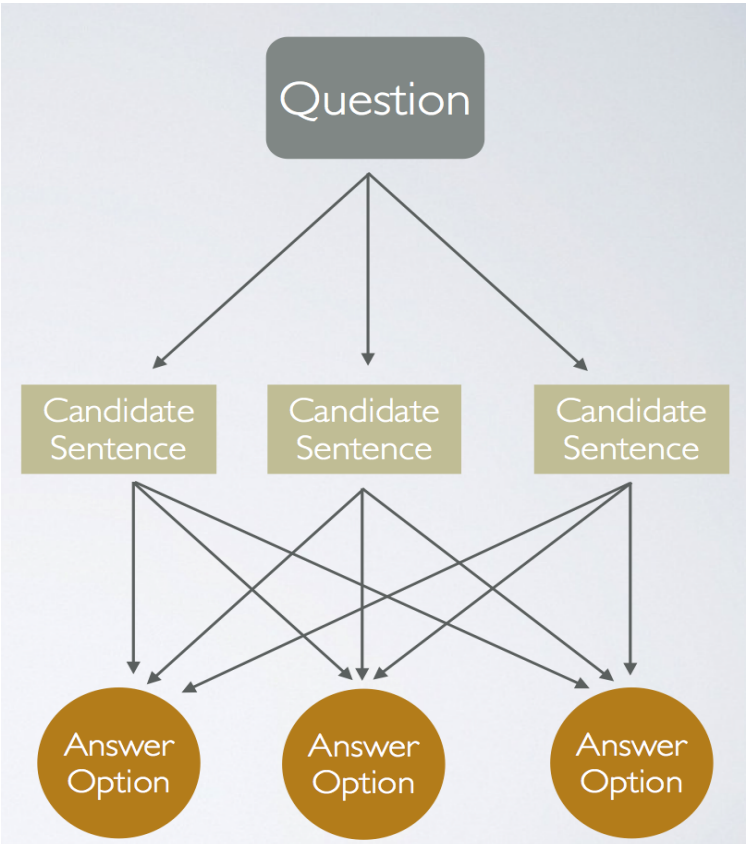

Fig. 2: Original Ranking Framework

2.5 Final Answer Ranking

We combine the scores from all the previous parts. To tune the weights, we use cross validation on the training set to optimized the performance.

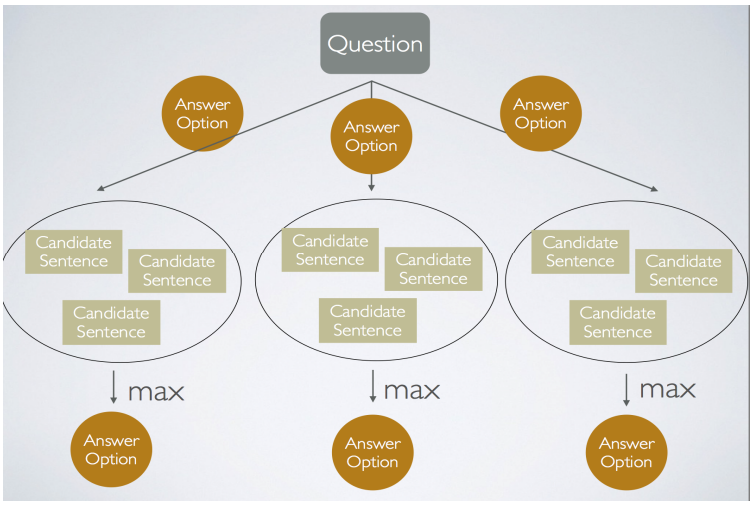

Fig. 3: New Ranking Framework

2.6 ”None of the above” and ”not answered” strategy control

For the question with ”None of the above” option, we have two thresholds, a and b. If the best score is below a, we will choose ”None of the above”. If the best score is between a and b, we select not to answer. If the score is above b, we choose the answer with the best score.

For the question without ”None of the above” option, we only have one threshold. If the score is above the theshold, we will choose the best answer, otherwise, we choose not to answer this question.

3 Methods

In this section, we present our methods developed for this project, in candidate sentence scoring and Answer scoring.

3.1 Candidate Sentence Scoring

In this component, we aim to find related sentences for a given question, and assign a score to each sentence and (question-answer) pair. Formally, let $q - a$ be the given query and answer pair, $s_1, ...s_i, ..., s_n$ be the sentences in the document, our target is a score function $f(q - a, s_i)$ that assign higher score to relevant sentences. Their are two parts in model f: feature extraction and similarity models.

Features The features express a question-answer or sentence using lexical information. There are three features:

1. unigram: unigram of q, s, very basic lexical features.
2. bigram: bigram of q, s, higher order ngrams are believed to be useful in QA tasks.
3. named entity/noun phrase: named entity and noun phrase carry important information of a question or sentence.

Similarity Between Answer-Question and Candidate Sentence Given two set of features from question-answer and sentence, we use several We use the following three methods to calculate the answer-question and candidate sentence similarity. Formally, let v_i, v_j be two sets of features, we evaluate several different ways of calculating the similarity of them using $f(v_i, v_j)$.

Cosine Similarity

$$f(v_i, v_k) = \frac{v_i v_k}{|v_i||v_k|}.$$

Dice Coefficient

$$f(v_i, v_k) = \frac{2|v_i \cap v_j|}{|v_i| + |v_j|}$$

Jaccard Similarity

$$f(v_i, v_k) = \frac{|v_i \cap v_j|}{|v_i \cup v_j|}.$$

To sum up, for candidate sentence scoring, we have three type of features and three type of scoring functions. We use CSE framework to evaluate the performance of all combination of them in training data, and select the best combination as our final submission.

3.2 Answer Scoring

Our final answer score comes from two parts: candidate sentence dependent score, as describe previously, and independent answer score.

From Candidate Sentence Score to Answer Score Using the models in section 3.1, for each query-answer pair, we will have a ranked list of candidate sentences. We choose the maximum score in the ranked list as the score for the answer we used to generate the sentence ranking:

$$S(q, a) = \max_{s_i} f(q - a, s_i).$$

Independent Question Answer Score As described in section 2.3, we have two scores generated directly from question and answer: answer type matching $t(q, a)$ and query answer PMI $p(q, a)$.

Ranking Model Given the three scores, we use a linear model to merge them for a final ranking score of answers:

$$w_1 S(q, a) + w_2 t(q, a) + (1 - w_1 - w_2) p(q, a) \quad (1)$$

$$\text{s.t. } w_1, w_2, (1 - w_1 - w_2) \geq 0 \quad . \quad (2)$$

As there is only two parameters in our model, it could be easily trained on cross-validation in our training data:

$$w_1^*, w_2^* = \text{argmax}_{w_1, w_2} \text{C@1 in CV}(w_1, w_2, \text{training data}).$$

4 Submitted Run

We only have one run of `output.xml`. Because we select parameters based on the best performance we have on the training data.

5 Individual Contribution

1. Bo Ma: Implementation of Jaccard similarity, error analysis on answer scoring, and answer strategy for 'none of the above' answers. Final report write up for corresponding parts.
2. Vinay Vemuri: Implementation of Cosine Similarity and Dice Similarity, error analysis on answer scoring and answer strategy for not to answer. Final report write up for corresponding parts.
3. Chenyan Xiong: Strategy and model design in candidate sentence selection, answer scoring, and 'none of the above' plus 'not to answer' choices. Implementation of final answer score merging weights training. Report write up for milestone 1, corresponding part in final report, and merging the final report.
4. Troy Hua: Design and build the full pipeline, including final parameter/method selection using CSE. Design and implementation of answer independent scoring. Presentation for milestone 2. Final report write up for corresponding parts.

6 Conclusion

Our contributions in this work are:

1. We experimented with richer type of features (unigram, bigram and noun phrase/name entity) and similarity functions (Cosine, Dicc and Jaccard) in candidate sentence scoring.
2. We modified the answer scoring pipeline and consider answer's similarity with candidate sentence together with question's similarity by concatenate question and answer together as a new 'question'.
3. We developed a new rule based score method that directly assign score to answers based on its type matching with question type. As a result, different question types are treated differently in our system.
4. We use training data to provide supervision in combining scores from our different component together for a final answer score.