Question Answering

Student Name: Zhenxiang Wang Student Number: 879694
Kaggle Team Name: zhenxiangw Username: zhenxiangw

1. Introduction

An automatic Question Answering system is developed to answer questions given the corresponding document. This paper will firstly introduce the basic question answering system as baseline, then make error analysis and explain the enhancements methods and reasons behind the choices, and finally discuss the future development of the system.

2. Basic Question Answer System

The basic question answer system contains 3 steps:

2.1. Sentence Retrieval

The first step is using standard information retrieval techniques to identify the sentence in the document which is most likely to contain the answer to the question. To achieve this goal, multiple preprocessing methods is applied to this situation, including tokenizing, stemming, lemmatizing, removing non-words and removing stop words. Then TF-IDF is applied to the term weighting. Finally, cosine similarity is used to find the most relevance sentence.

2.2. Entity Extraction and Question Classification

The second step is extracting the name entity in the sentences and classifying the question into different types. The basic QA system extract name entity by running the Spacy Name Entity Recognition (NER) system to each sentence and combining some similar name entities. There are 11 name entity types, such as "GPE", "LOC", "PERSON", and so on. The question type set is the same as the name entity type set. Question classification is implemented by defining some rules using regular expression.

2.3. Answer Ranking

The final step is ranking answer. First, name entities that match the question type will be ranked higher. Second, name entities which appear in the question will be ranked lower. Thirdly, sort the answers based on their distance from the open-class word from the question. Finally, the first answer from the sorted list will be selected and then processed to match the expected format.

3. Error Analysis and Enhancements

After implementing the basic QA system, error analysis on the development dataset is applied to make enhancement to the system.

3.1. Sentence Retrieval Analysis

Precision@k is used as the evaluation metric for the sentence retrieval task. Different retrieval methods are tested to find better strategy to enhance the accuracy of this step. The results are shown in Chart-1.

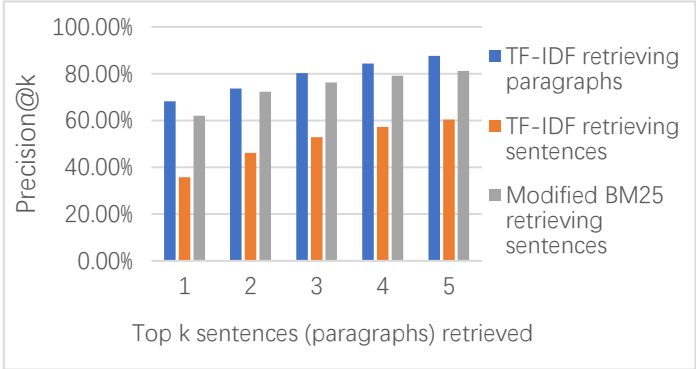

Chart-1 Precision@k for different sentence (paragraph) retrieval methods

3.1.1. Errors

As shown in Chart-1, the original TF-IDF term weighting performs well in retrieving paragraphs, however, its performance becomes very poor when retrieving sentences. The reason behind this is related to our data set and the specific application scenario. The TF-IDF algorithm is based on such a hypothesis: the most meaningful terms for different documents should be terms with high frequency in the document and less frequency in the other documents of the whole document set. This simple hypothesis cannot effectively reflect the importance of the term and the distribution of the feature terms especially when dealing with document with small size like sentence (Xia, T., & Chai, Y., 2011).

3.1.2. Modified BM 25

To solve this problem, a modified version of BM 25 (Robertson, & Zaragoza, 2009) is applied to do the term weighting. In the original BM 25, when the number of documents containing a term exceeds a certain percent of the total number of documents, IDF becomes negative, which is not reasonable in this case because although some keywords in the question appear in many sentences, they are still important features to retrieve sentence. Therefore,

we reduce the impact of IDF and make it never become negative. Other parameters are also tuned to increase the precision. In addition, 5 sentences are retrieved to increase the total precision of sentence retrieval task. Finally, as shown in the Chart-1, the average precision of sentence retrieval has been increased by about 20%.

3.2. Question Classification Analysis

To analysis the error occurs in question classification task, for training and dev data, we use NER tag of the answers as the type of the questions. The question types predicted by the system are then compared with the labels mentioned above. After doing these, two main errors have been found.

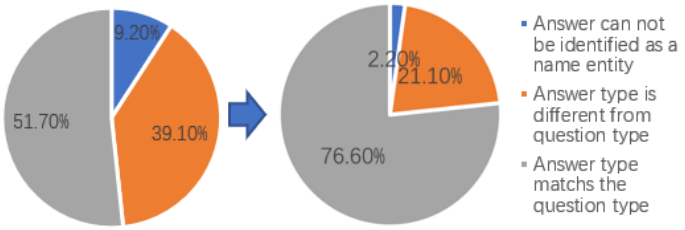

Chart-2 Performance of Question Classification task

3.2.1. Errors

As shown in the left pie chart of chart 2, firstly, a large part of the questions in the dev data is not correctly classified. The reason is that the rules for question classification cannot cover many special questions. Secondly, some answer words cannot be identified as a name entity.

3.2.2. LSTM Question Classifier

The second issue is handled by pos tagging the sentences and extract the noun word if there is no name entity identified. However, for the first issue, adding more and more special rules is not a good solution, since it is not generalized, and it is easy to overfit the training data. Therefore, a question classifier based on Long Short-Term Memory (LSTM) is built to do this task. LSTM is a kind of time recurrent neural network, which is suitable for dealing with important events with relatively long interval and delay in time series (Graves, & Schmidhuber, 2005). It meets the need of question classification in this scenario.

3.2.2.1. Preprocessing

The model is implemented using Keras package. After processing the data string, sentences are tokenized, turned to vectors and then padded to the same length. Then word embedding is then applied to turn the word vectors into dense vectors of fixed size as the input of the neural network. The question type labels are turned to one-hot vectors as the labels of the neural network.

3.2.2.2. Network Structure

The network has 3 layers. The first layer is a LSTM layer with 60 nodes. The max pooling is applied on it to reduce the dimension. The second layer is a densely-connected layer with 50 nodes, using rectifier linear unit as the activation function. The output layer is a densely-connected layer with 11 nodes, using softmax as the activation function to output the probability distribution of question types. In addition, dropout is applied to each layer to disable 10% percent of the nodes when training to prevent overfitting.

3.2.2.3. Training and Testing

We use training data to train the model and dev data to test the performance. The results are shown at the right pie chart of Chart-2. The precision of question classification has been raised by 24.9%.

3.3. Performance of each Enhancement

The macro-averaged F1 score is used as the metric to evaluate the final performance of each enhancement to the system on the dev data. The relative performance on the Kaggle leaderboard is also given as a reference. The result shows that two enhancements have improved the performance of the system.

QA system	F1 Score on Dev data	Mean F-score on Kaggle
Basic QA	12.8%	13.5%
After Sentence Retrieval Enhancement	19.7%	20.6%
After Question Classification Enhancement	23.9%	24.5%

Table-1 Performance of each Enhancement

4. Future Work

In the future, an end-to-end machine reading and question answering system will be implemented. The system will try to build a model like QANet proposed by Google Brain (2018), which combines local convolution with global self-attention to increase the performance of the system.

5. Conclusion

In this paper, we propose an automatic question answering system. The system is analyzed and then enhanced by applying a modified BM 25 to retrieve sentences and building a question classifier based on LSTM to classify questions. These enhancements all improve the performance of the system.

6. Reference

Graves, A., & Schmidhuber, J. (2005, July). Framewise phoneme classification with bidirectional LSTM networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on* (Vol. 4, pp. 2047-2052). IEEE.

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, *3*(4), 333-389.

Xia, T., & Chai, Y. (2011). An improvement to TF-IDF: Term Distribution based Term Weight Algorithm. *JSW*, *6*(3), 413-420.

Yu, A. W., Dohan, D., Luong, M. T., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *arXiv preprint arXiv:1804.09541*.