COMP90042 Web Search and Text Analysis
QA System

Team name: yundongm
Member: Mao Yundong (yundongm@student.unimelb.edu.au)

1. Introduction

This project aims to extract the answer to the question from the corresponded document.

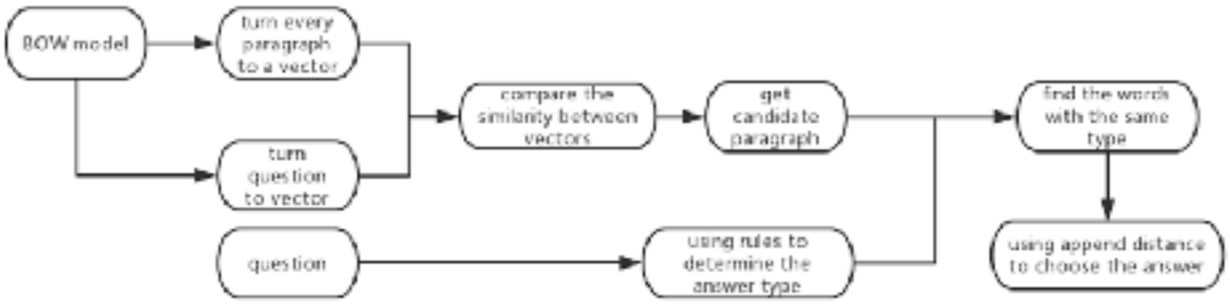
2. Data Set

The data set contains four JSON files: documents.json, training.json, devel.json, testing.json.
documents.json: contains about 440 documents, and our answers to questions are all extracted from these documents.

training.json and devel.json: contains questions and answers.

testing.json: this file just contains the questions and the docid for each question. And the aim of this project is to extract the answer from selected document.

3. Method


BOW model
turn every paragraph to a vector
turn question to vector
compare the similarity between vectors
get candidate paragraph
find the words with the same type
question
using rules to determine the answer type
using append distance to choose the answer

Data preprocess

First using BOW model to collect all the words from the dataset. And regard each word as a feature. And then for every paragraph can turn into a vector.

And during this process, we will remove all stop words which have no means. Then turning the question and all paragraphs to the vectors (easy to be handled).

cos_similarity

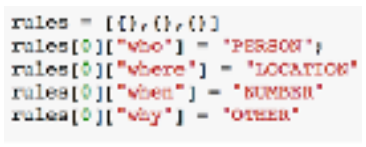
And for this part we using cos value between the vectors to represent the similarity between the question and each paragraph in one document.
Tools: from sklearn.metrics.pairwise import cosine_similarity

Rules For Answer Type

I use a rule that created by myself to determine the answer type of the question.
All answer can be divided into four types: LOCATION, NUMBER, PERSON, OTHER.

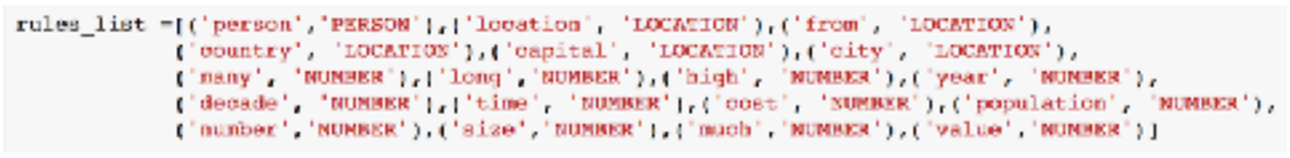
Rules:

First priority:



```
rules = [{},{},{}]
rules[0]["who"] = 'PERSON';
rules[0]["where"] = 'LOCATION'
rules[0]["when"] = 'NUMBER'
rules[0]["why"] = 'OTHER'
```

Second priority:



```
rules_list =[('person','PERSON'),('location', 'LOCATION'),('from', 'LOCATION'),
             ('country', 'LOCATION'),('capital', 'LOCATION'),('city', 'LOCATION'),
             ('many', 'NUMBER'),('long','NUMBER'),('high', 'NUMBER'),('year', 'NUMBER'),
             ('decade', 'NUMBER'),('time', 'NUMBER'),('cost', 'NUMBER'),('population', 'NUMBER'),
             ('number','NUMBER'),('size','NUMBER'),('much','NUMBER'),('value','NUMBER')]
```

If the question contains any word in the first priority list, we directly using corresponded type. And if not in the first priority we check the second priority list and use the corresponded type if it in the second priority list. And if the rules can't be satisfied. We mark it as OTHER type.

Append Distance

Then in the corresponding paragraph, we can get some candidates and in order to choose one from candidates, we calculate the append distance for every candidate. I map all the word except the stop words to the position in the sentence. And then sum the distance of one candidate index to all other keywords. And the smallest distance will be our ultra answer.

4. Result Analysis And Improvement

Result Analysis:

I finally get the result of 18.02% F1 score on the kaggle. Then I try the develop data and get a 19% F1 score. I find that there are three main reasons which cause false predictions. First, the append distance method often mistakes because many right answers are not the closest ones. The second one is that number of answer types is not enough. The OTHER type contains around 40% prediction. And because the OTHER type contains all the named entities. So the prediction on the OTHER type is the lowest. And the last is the chunk partition, the spacy can't actually help us find the real noun chunk. So our prediction often contains the extra word.

Future Improvement

1. Replace the append distance with a better method to high the accuracy when choosing words in the same type.
2. Add more types for the answer type.
3. Using a better method to do the noun chunk partition such as some deep learning method.s

Conclusion

At first, I try to use the deep learning method to do this project but failed. Although there are some papers which are not very long and get a really high accuracy. I still can't construct the neural network correctly. And the GPU Compute Capability is really the limit for me. I try to construct a simple CNN to predict the answer type but the training process is really slow using CPU.

From kaggle we can see that there are many amazing predictions, And I guess it's what deep learning can do, really more advanced than Tradition Method.

Reference

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. CoRR, abs/1611.01603, 2016. URL http://arxiv.org/ abs/ 1611.01603.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, Quoc V. Le. QANET: COMBINING LOCAL CONVOLUTION WITH GLOBAL SELF-ATTENTION FOR READING COMPRE- HENSION. Published as a conference paper at ICLR 2018.