COMP90042 Project Automatic Fact Verification

Ruilin Liu 871076, Jianyu Yan 930562

Abstract

This document will illustrate the implement of automatic fact verification project. There are five main sections in this paper. Firstly, we will introduce the requirements of this project, and the source dataset. Then, this article shows the details of design, development and performance of automatic fact verification system. The comparison and analysis of methods we tried will be discussed in the next section. Finally, a conclusion and possible improvement will be given.

1 Introduction

In the past years, the amount of data increases dramatically and it becomes hard to verify and validate the correctness and quality of those data timely only by manpower. Therefore, it is necessary to verify factual information quickly and automatically in various circumstances. Otherwise, the failure of verification will lead to some undesirable and serious problems such as spreading rumour, loss of profits, the stock market fluctuation and even illegal behavior.

According to the pdf file downloaded from LMS, this project aims to develop an automatic fact verification system. The application is based on information retrieval system, machine learning system and other knowledge technology based model. Firstly, a search engine will be established based on wiki-pages-text data, then we will process claim sentence and use the processed list as query to search top-k relative documents result. After getting top-k documents, we tried two methods to re-rank the results to improve information retrieval performance, TF-IDF and word2vec similarity. On the other hand, this project trains open source neuro-linguistic programming (AllenNLP) model to predict the verification result of dev data.

There are several challenges in this project, such as how to improve information retrieval system performance, how to pre-process data to guarantee its quality, balancing the cost and performance of NLP model training. Besides, we import acl latex template introduced in the project pdf file to write this report.

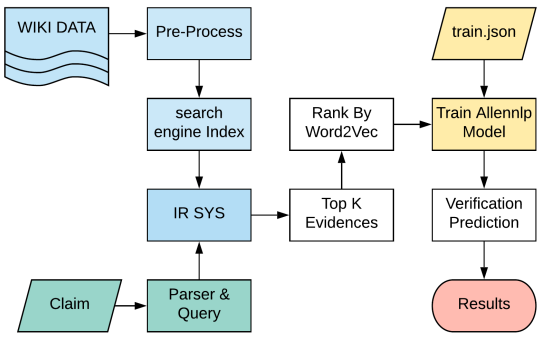

Figure 1: Structure of whole system

2 Relative work

There are various official and personal submissions of solutions of Fact Extraction and Verification Competition (FEVER). According to (Luken et al., 2018), they parse claims to get key words and search sentences containing more than two key words as information retrieval candidates instead of building a search engine. Another team provides another solution, they demonstrates a more complex repeatable verification system and claims that it can be extended on distributed system; but they did not give a measuring matrix and discuss the performance (Chess et al., 2004).

3 Data Set

According to the wsta-project description, the dataset is derived from workshop on Fact Extraction and Verification (FEVER). The zip file

contains 109 text files which are used to store preprocessed data from Wikipedia. In wiki text file, each line represents a sentence in original webpage and can be departed into three pieces, the document identifier, sentence serial number and content of sentence. Here is an example of data in wiki file.

```
1986_NBA_Finals 2 The Celtics defeated the Rockets four games to
two to win their 16th NBA championship .
1986_NBA_Finals 3 The championship would be the Celtics ' last
until the 2008 NBA Finals .
```

Figure 2: line in wiki-xxx.txt

But there are some lines not formulated well in dataset, it may cause variable type error or index out of bounds exception. Here is an example shown in figure 3, the middle term in 32864th line is "Zhongshan" instead of a integer number for sentence id. And, this will cause error while we are trying to change its variable type by int(str) after reading.

```
32863   Yuanshan_Station 0 The Taipei Metro Yuanshan Station is
        located between the Zhongshan and Datong districts in Taipei
        , Taiwan .
32864   Yuanshan_Station Zhongshan Zhongshan District, Taipei
32865   Yuanshan_Station 1 It is a station on the Tamsui Line -LRB-
        Red Line -RRB- .
32866   Yuanshan_Station 2 There was a station of the same name on
```

Figure 3: Error data in wiki-109.txt

The json files are "training.json", "devset.json" and "test-unlabelled.json". Data in json files are formulated as dictionary data structure to show details of claim, label and evidence. Claim is used to show the sentence to be verified. There are three labels, "SUPPORTS", "REFUTES" and "NOT ENOUGH INFO" for no evidence. And evidence section will show detail information for support or not. This project takes advantage of "training.json" to train model and uses "devset.json" for analysis and measuring performance. Json structure is shown as below.

```
"137334": {
  "claim": "Fox 2000 Pictures released the film Soul Food.",
  "label": "SUPPORTS",
  "evidence": [
    [
      "Soul_Food_-LRB-film-RRB-",
      0
    ]
  ]
},
```

Figure 4: json structure

4 Solutions

4.1 Information Retrieval System

For information retrieval task, we have tried two libraries, Pylucene and Whoosh, instead of coding inverted index list by ourselves. However, the performance of whoosh is very terrible, it is almost four or five times lower than Pylucene at testing stage, thus we decided to go into Pylucene. As Pylucene is an python extension of Java Lucene (Foundation, 2019), there is almost no available document or tutorial for it as the api varies between different versions. As a result, we change to use Lupyne. This is a more pythonic search engine library based on Pylucene as well (Coady, 2019).

We define a python class to read and process data from wiki text files. This class contains a iterator method to yield data required by search engine. To improve performance, we clean undesirable data like replacing stopwords, punctuation and duplicated space characters. Then, we build a index by Lupyne. For each claim, we analyze it by NLTK Averaged Perceptron Tagger, then form different query statements. By querying each statements, we find the k nearest sentences from the wiki based on BM25 similarity.

Additionally, We use a TF-IDF vector space model to re-rank the top k sentences. In this process, we apply Wu-Palmer Similarity to identify synonyms and treat them as the same word. To improve the sentence precision, We also set thresholds to eliminate sentences with low similarities.

Moreover, we have tried to train a word2vec model and calculate the sentences similarity by mean vector of words. Although it works well in our testing stage, it takes more than ten hours to establish model based on the whole wiki text files. Considering the limit of time and the acceptable performance of TF-IDF re-rank, we drop it into abandoned version.

4.2 Verification Part

We select Textual Entailment model from AllenNLP (Gardner, 2018), which re-implements the decomposable attention model. It is used to predict whether the first sentence implies the facts in the second sentence. The basic idea is to get an

embedded representation by processing each token of the claim and evidence, then align same words and recognize the relationships between other aligned phrases. Finally make a conclusion based on the previous analysis (Parikh et al., 2016).

5 Comparison and Error Analysis

5.1 Index Establishment

As each wiki is separated to several sentences, at the beginning, we plan to merge the sentences from the same wiki term and add them to the index as one document. When we query a claim, it returns the top k wikis, then we split each sentence and use TF-IDF to re-rank them. However, this method has low performance. Alternatively, we change to rebuild the index and add each sentence as a document, then query the top k sentences, use TF-IDF to re-rank and select the top 5. In this case, it will take more time due to increased number of documents, but the performance improve slightly.

By analyzing the output, we found out that the reason is the Lupyne library includes some enhancements during searching, which could be better than pure TF-IDF model we developed. Therefore, as the third attempt, we firstly query the top k wikis from the first method, then build a new in-memory index using the sentences in the k wikis. Next, we query top k sentences from the new index. Although this method has the best performance, it costs more than 6 hours for "devset". Every query gets slower than before even we delete and close every in-memory index after each query. Finally, we abandoned this method due to the time limitation and chose to use the sentence index in the following development.

5.2 Query and Re-Ranking

5.2.1 Query

By analyzing the train set, we realize that most claims are a simple subject–verb–object structure. For example, In "train.json", the first claim is "Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.", the second sentence is "Roman Atwood is a content creator." and the fourth claim is "Adrienne Bailon is an accountant." Apparently, the subject is the most important part of the query and it might be more likely to exist in the wiki's title. In some case, the object might be possible to exist in the title. We import NLTK Averaged Perceptron Tagger to tag each token in the claim, then separate the tokens before the first verb as "Subject". We also get the last few tokens which has tags start with "NN" as "Object". In order to apply to sentences with other structures, we extract all "NN" tokens as the "NNs query". When we query a claim, we firstly query 2 sentences using "Subject" by title field, then query 2 sentences using "Object" by title field, finally query 4 sentences using "NNs query" and the original claim by text field. The proportion is decided by experiments with "devset".

devset	Doc Index (%)	Sentence Index (%)	Doc + in-memory index (%)
Sentence Precision	16.14	17.12	17.90
Sentence Recall	60.49	67.12	69.69
Sentence F1	25.48	27.29	28.48
Document Precision	41.52	29.81	36.23
Document Recall	76.20	79.89	81.21
Document F1	53.75	43.42	50.10

Table 1: Index performance measuring matrix

Subject Subject	Verb	Object
Nikolaj Coster-Waldau	work	Fox Broadcasting Company
Roman Atwood	is	content creator

Table 2: Subject-Verb-Object structure

5.2.2 Re-ranking

We implemented a TF-IDF model to re-rank the 8 sentences which are returned by Lupyne. We import the Wu-Palmer Similarity to identify synonyms. Specifically, if the similarity of two terms is larger than 0.8 (which decided by multiple tests), the system will treat them as the same word. In the train set and dev set, most evidences are less than five sentences. To improve the sentence precision, the system will eliminate the sentences with scores t points less than the first ranked sentence. We manually binary search the threshold

and set it to 0.15. However, it could have negative influence on recall. By applying improvements above, the information retrieval system shows an acceptable performance on devset.

Sentence Precision	41.53%
Sentence Recall	52.44%
Sentence F1	46.35%
Document Precision	55.10%
Document Recall	68.25%
Document F1	60.97%

Table 3: Measuring matrix

As mentioned in the solutions section, we have tried word2vec model as re-rank method. We takes advantage of gensim library to train model. Doc2vec can be regarded as an unsupervised advanced word vector. It adds another feature vector, the document id vector. Also, we could provide more features like tags. Then, it is available to check the similarity of every unique document to every tag and predict tag with highest similarity to document. In testing stage, we tried to separate document id (wiki term) into tag list after removing stopwords and punctuation.

5.3 Textual Entailment

In the first version, we use the evidences in train set to train the Textual Entailment model. For all "NOT ENOUGH INFO" instances, we randomly choose a sentence from the wiki. However, there are a large number of disorganized sentences in the wiki dataset. An example in the "wiki-001.txt" is:

```
1981–82_French_nuclear_tests 4 –RCB– 5 m +
1981–82_French_nuclear_tests 6 | underground shaft , weapons development
1981–82_French_nuclear_tests 8 |
1981–82_French_nuclear_tests 10 | style = `` text-align : center ; '' |
1981–82_French_nuclear_tests 12 |
1981–82_French_nuclear_tests 14 |
1981–82_French_nuclear_tests 16 |
1981–82_French_nuclear_tests 18 | –
1981–82_French_nuclear_tests 20 !
1981–82_French_nuclear_tests 21 Tyro
1981–82_French_nuclear_tests 23 | 17:27:00
1981–82_French_nuclear_tests 25 | style = `` text-align : center ; '' | TAHT
Rim zone , Areas 1–2 , Moruroa Atoll : Dahlia6
1981–82_French_nuclear_tests 27 | 5 m +
1981–82_French_nuclear_tests 29 | underground shaft , weapons development
1981–82_French_nuclear_tests 31 |
1981–82_French_nuclear_tests 33 | style = `` text-align : center ; '' |
1981–82_French_nuclear_tests 35 |
1981–82_French_nuclear_tests 37 |
1981–82_French_nuclear_tests 39 |
1981–82_French_nuclear_tests 41 | –
1981–82_French_nuclear_tests 43 !
```

Figure 5: wiki example

Most random evidences are meaningless, and it cause no instance in the test dataset could be labeled as "NOT ENOUGH INFO", because all evidences provided by the IR system above are totally different from the random evidences. A solution is using IR system to query the nearest sentences as evidences. At first, for each claim, we concatenate all evidences of each claim as a single string. In order to ensure the "consistent style" of evidences, we use our IR system to search evidences for train set and dev set, then use them to train the Textual Entailment model.

However, the IR system doesn't have high accuracy, which could influence the training. To reduce the impact, we try to use the original evidences to train the model by single sentence, and predict the label for each evidence of a claim. Finally, we label the claim by the most common label among multiple evidences, then abandon the sentences which has a different label with the final label.

	Retrieved Evidences	Default Evidences
Epoch	40/140	45/140
Training During	2:42:41	4:08:45
Training Accuracy	0.725	0.894
Training Loss	0.632	0.268
Validation Accuracy	0.492	0.556
Validation Loss	1.207	1.242

Table 4: Training result comparison

5.4 Final result

After analysing final result's output, we figure out that TF-IDF will allocate high weight for some short sentences containing key words. However, actually those sentences can not be regarded as evidence. From our point of view, it may be a good idea to take sentences' length into consideration to solve this problem.

```
('Label Accuracy', '\t\t47.43%')
('Sentence Precision', '\t41.21%')
('Sentence Recall', '\t51.65%')
('Sentence F1', '\t\t45.84%')
('Document Precision', '\t54.59%')
('Document Recall', '\t67.12%')
('Document F1', '\t\t60.21%')
```

Figure 6: Final Result

6 Conclusion

To sum up, this project is a automatic fact verification system based on pre-process approach information retrieval system, re-ranking method and neural network model for prediction. We have tried various methods or system structures after reading associated paper, the final version is the result of balancing performance improvement and cost of time and space. Without the limitation, it may be better by some approaches like modifying Pylucene search engine functions, do more iterations during allennlp model training stage or increasing dimension size of word vectors.

References

Chess, D., Krasikov, S., Morar, J., and Segal, A. (2004). Fact verification system. US Patent App. 10/324,723.

Coady, A. (2019). Lupyne's documentation. Retrieved from https://pythonhosted.org/lupyne/.

Foundation, A. S. (2019). Welcome to pylucene. Retrieved from http://lucene.apache.org/pylucene/.

Gardner, M. (2018). Textual entailment model. Retrieved from https://allennlp.org/models.

Luken, J., Jiang, N., and de Marneffe, M.-C. (2018). Qed: A fact verification system for the fever shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 156–160.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.