Automatic Fact verification

Yinghao Zhu 887402
Zhaopeng Xie 807605

1 Introduction

In a data explosion decade, information is generated every second among the world. Therefore, it becomes more challenging and consuming to judge facts manually. In the light of this, letting machines verify facts becomes a trend and a hot topic for years.

Accordingly, we conduct this project is to build an automatic fact verification system which can predict whether a claim is true or false. To achieve reasonably accurate predictions, Information Retrieval, Semantic Analysis and Machine Learning technologies are applied in this project constituting a pipeline.

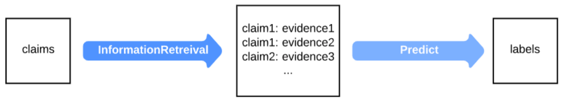

Figure 1: System pipeline

This report will introduce these methodologies and how they are implemented followed by results representation. After that, we conduct analysis on the result drawing a conclusion

2 Dataset

In this section, we present given dataset and how they are used in our project. All these will be pre-processed by operations includes lowercase, stemming, remove stop words and punctuation before using.

- "wiki-text.zip": a collection of Wikipedia documents, the resource where we trace evidences according to a claim.
- "training.json": a collection of segments which contain a claim, a label and a list of evidences (one evidence is referenced via an article name and sentence index).
- "devset.json": the same format with "training.json".
- a collection of segments which contain only a claim.

During developing, we used "training.json" to train our conducted model, and take advantage of "devset.json" to improve it. Finally, "test-unlabelled.json" is the target we are predicting with. For one claim, our system generates an output ranging from "SUPPORTED" indicating all relevant evidence consistent with the claim; "REFUTED", indicating the claim is opposite from traced evidences; and "NOT ENOUGHI NFO" showing that our system cannot retrieval any evidences.

3 Methodology

In this section, we will compare underlying technologies and explain how we apply and improve them to suit this project.

3.1 Information Retrieval

Naturally, TF-IDF is ideally the easiest method to achieve information retrieval, while due to the large amount of data are given, it may be not effective enough for this project. Thus, we purposed our own searching system. This search engine include Name Entity Recognition, Post of Speech, N-gram similarity and so on.

Our Information retrieval system is stem from inverted index and B-Tree(figure 2). The first part is a term-frequency dictionary. Instead of using contents of wiki to build this list, we built this by using all titles. Following part is a title list that contains all titles that include a specific word. The next is a title-document map, which indicates a title and which article it is and the start and end pointer of its contents. When a claim is given, we first do pre-processing

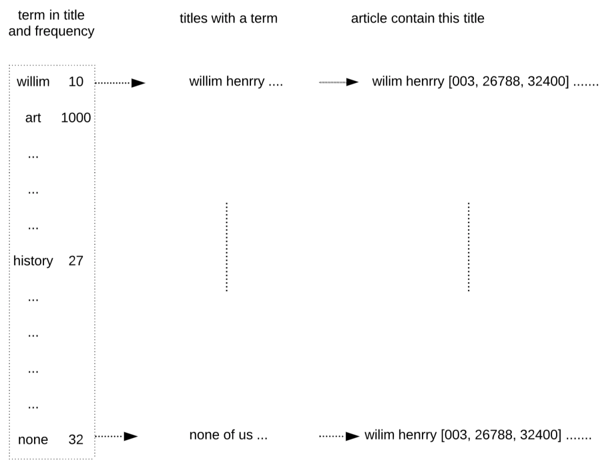
term in title and frequency
titles with a term
article contain this title
willim 10
willim henrry ....
wilim henrry [003, 26788, 32400] ....
art 1000
...
...
...
history 27
...
...
...
...
none 32
none of us ...
wilim henrry [003, 26788, 32400] ....

Figure 2: The search engine structure

which includes lowercasing, stemming, removing stop words and punctuation. After that, we apply NER of spaCy to identify all entities, as well as the combination of NER and POS identifying subjects to ensure futher comprehensiveness, because NER itself cannot find out entities such as "history of art", but POS can do that. We also have redundancy removal, for example NER system may return "Willim Henrry", while POS return "Henrry", it is removed for improving efficiency.

After entities are addressed, we start from the sparsest word to find titles contain this word where 1-gram similarity and 2-gram similarity are implemented. 1-gram is response to the situation that entities just have exactly one word, for example "cat", while 2-gram considers two continuous words co-occurrences in entities. Then the title with the highest similarity will be returned. Besides, this title will be used to identify wiki documents that include this title followed by applying the wiki-document number and two pointers to gather all related contents for our next module, semantic analysis.

3.2 Label Prediction

For this stage, we aim at one claim and relevant evidences by processing each claim - evidence pair from our search engine to see whether they are semantically consistent and then determine which label it belongs to.

Among current text analysis methods, TF-IDF is again the most intuitive one. This method counts term frequency (TF) and inversed document frequency (IDF) of all different terms among all documents to conduct a map between TF-IDF value and label. However, sequences in this model is ignored, while bag of words hardly contributes to semantics analysis. Besides, POS Tagging is also a potential solution. This method tries to tag each word with a part of speech and extract a rule that may help in predicting label. Although there are several ways to handle sequence issues, same sequences express different ideas, for example, I like it and I dislike it. Therefore, we need to find out a better way for this project to analysis semantics.

As researching, existing model made by Aditya and Jonas help to contribute to our project(Mueller and Thyagarajan, 2016). In their model, they apply vectors to represent each word and calculate through LSTM model to conduct a vector which can represent one sentence. Accordingly, two sentences are represented by two vectors. After receiving these two vectors, they calculate Manhattan distance to determine whether how likely they are presenting the same idea. Two sentences are transformed and being presented by two vectors, and Manhattan distance is calculated by them.

Apparently, according to their model, the output is just a value ranging from 0 to 1. We have to design a classifier to determine which value represent "SUPPORTED" and which value belongs to "REFUTED". Nevertheless, label cannot be determined simply by the distance only, because the sentences themselves also affect classifier, namely, two vectors. Therefore, basing on Aditya and Jonas s model, we introduce more layers taking two vectors and Manhattan distance as import to predict label.

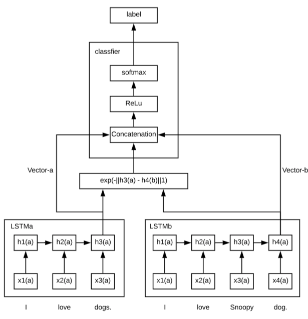

Figure 3: The search engine structure

As shown in the figure above, concatenation layer combine two embedded vectors and distance into one more complex vector and send it to one dense layers and a classfier. Finally, a label is output. Then, we use the lable of these sentences to label all claims. If there is a “supports” or “refuted” sentence of a claim, this claim will be labeled as this evidence, or it will be labeled as “not enough info”.

We have also considered to calculate semantic TF-IDF, due to claims and evidences are both short sentences, these counts may make less senses. Finally, we consider neural network models built with semantics. Because language and meanings are sequences sensitive, a model with lower vanishing gradient and exploding gradient. Hence, LSTM is addressed in our system.

4 Discussion

4.1 Results

The table below shows the results of validation set and test set.

dataset	Article Acc	Label Acc	Sentence F1 Score
dev	0.480	0.442	0.254
test	0.479	0.470	0.233

Table 1: The results of devset and testset

As can be seen for it, the Article Accuracy for development data set is 0.480. The label accuracy is 0.442, while the sentence selection F1 score is 0.254.

4.2 Error Analysis

In development set, Label Accuracy loss is due to the NER process and title finding process. This will lead to a Lable Accuracy loss. For example, the claim “Damon Albarn’s debut album was released in 2011.” is supported by sentence 17 in “Damon_Albarn”, but our system failed to find this article. Instead, this claim is labeled as “NOT ENOUGH INFO”.

Except searing error, the reason why F1 score is just 0.254 also include that our neural network are not good enough to distinguish not related information from refuted and supported sentences. To be more detail, an example is given below.

“$claim$” : ”$Fox2000Picturesreleased thefilmSoulFood.$”

For this claim in development set, its label is ”SUPPORT” but just one sentence to support this claim while our system returns 5 sentences which includes 4 sentences that not not related in the “Soul_Food_-LRB-film-RRB-” article. This lead to a low F1 score. According to our observation, this is the main reason for F1 score lose.

Apart from searching errors, there are errors in label prediction system as well. Basing on algorithm principle analysis, errors occur when evidences are linked. We cannot correctly label this claim-evidence pairs shown in table below.

claim	if A is true then C is true.
evidence1	if A is true then B is true.
evidence2	if B is true then C is true.

Table 2: Evidence chain

Apparently, given these two evidences, claim should be labelled as SUPPORTED, however, our system only check whether single claim and evidence are expressing the same idea. Unfortunately, we cannot find a counter example because of the extremely large dataset.

5 Conclusion

Our work can be improved from two aspects. First, train a NER model according to train set to improve searching model. Second, using more deep neural network to train a classifier, such as Bert. Though we do not achieve a good performance, this project allows us to understand web search and text analysis topics better, which is impressed.

References

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2786–2792. AAAI Press.