COMP90042 Project 2 Fact Verification System Report

931031 Fan Yang 951247 Xinhao Zhang

Abstract

This paper proposes to tackle the problems of validating whether a claim is true, false or not supported based on a Wikipedia text corpus. This task combines the challenges of document retrieval with comparing semantics between two sentences by machine learning method. In our project, we first utilize TF-IDF and bigram to match claim with related text. After generating claim-evidence pair csv file, we then train an Enhanced Long Short-Term Memory (ESIM) model with this file. The model classifies and labels the sentences pair as SUPPORTS, REFUTES and NOT ENOUGH INFO. Our experience indicates that word embedding and ESIM are quite effective to learn the semantics between sentences as our best label accuracy is 47.7%, while the Sentence Selection F1 is 25.3% on the Codalab competition.

1 Introduction

This paper considers the problem of automatic fact verification and the unique resources of answers is Wikipedia. When a claim comes, we will search related sentences in the Wikipedia text corpus and by comparing their semantics, the claim will be labelled as SUPPORTS, REFUTES and NOT ENOUGH INFO.

In this project, the data set we use contains a large number of claims and evidence. Thus, we have enough training data set to train our model. Meanwhile, to reduce the time cost of training, we do not consider all data in the given training data set, which will be explained in the training section.

2 Document Retriever

In this paper, how to search evidence by analysing the given claim will be the first task. According to DrQA (Danqi et al., 2017), we tried to utilize inverted index first to retrieve related documents of claims. However, because of a great number of documents and terms, the inverted index will have lower performance. Therefore, we modified the retriever designed by Danqi et al. and use TF-IDF and bigram to retrieve documents.

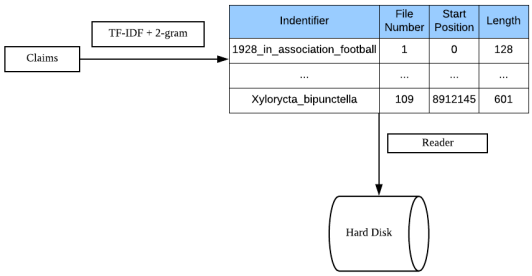

Figure 1: Data Retriever Overview.

The architecture of the document retriever is shown in figure above. Firstly, we build an index file stored in memory to indicate which file the page identifier is in and the offsets and length of the page identifier in the file. Thus, each row of the index file contains the number of files, the start position in the file and the length of the related documents. Therefore, with this index file, we can search the document according to the page identifier more conveniently.

The most important part of the document retriever is the searching model based on TF-IDF. We consider the page identifier as documents in TF-IDF as loading whole text files in memory will cost lots of space. Before tokenize the identifier, we preprocess the text by replacing underline with space and remove the signs, namely -LRB-, -LSB-, -RRB-, -RSB- and other punctuations. Then, with help of Name Entity Recognition, which will be described below, we can search the identifier by the name entities in the claims, which would increase the accuracy of searching related documents. Besides, n-gram is also added in the model, because some combined terms will have different

searching result from each term alone. For example, "NBA Finals" will have different results from "final". Therefore, a TF-IDF model with 2-gram is constructed and, in this model, the page identifiers are seen as documents and the name entity in claims are seen as query. In this way, the document retriever is built.

In this part, we utilize Name Entity Recognition provided by Spacy and the pretrained model "en_core_web_sm" is used. Actually, the model "xx_ent_wiki_sm" is more useful to recognize name entity in Wikipedia texts corpus, and it supports multiple languages. However, since claims in data set are all in English except some name entity in other languages, such as some French names, we finally select the model "en_core_web_sm" which will find out name entity in English texts. To increase the accuracy of Name Entity Recognition, we also consider the pos of tag in the sentence, i.e. the subject of sentence is also seen as name entity as the subject of the sentence is the most importance clue to search documents in our daily life.

After that, when the TF-IDF model search the identifiers, they are used as a key in the index file to read documents. As explained above, the index file indicates the location of documents by keys, and finally the file on disk can be read.

3 Sentence Pairs Classification

The second part of our fact verification system is the sentence pairs classification model. The document retriever will find all the satisfied documents and we pairs each of them, which means that a claim may combine with several evidences. Those evidences may be either of the three labels: SUPPORTS, REFUTES and NOT ENOUGH INFO. Therefore, our model needs to classify the sentence pairs into three tags. At the early stage, we tried the Siamese Long Short-Term Memory (LSTM) Network (Mueller et al., 2016). It contains two LSTMs. One is used to process the claim and the other is for the evidence. We calculated the distance between two outputs and applied a simple classification layer based on the distance. However, this model did not work well, so we turned to use the Enhanced LSTM (ESIM) model (Chen et al., 2017).

3.1 ESIM Model

The ESIM model is a Natural Language Inference (NLI) model, which has been applied on the Stanford Natural Language Inference (SNLI) dataset and has good performance. The target for NLI is to distinguish whether sentence pairs, one is "premise" and the other is "hypothesis", have the label of "entailment", "contradiction" and "neutral". Those three labels are closely connected with our labels.

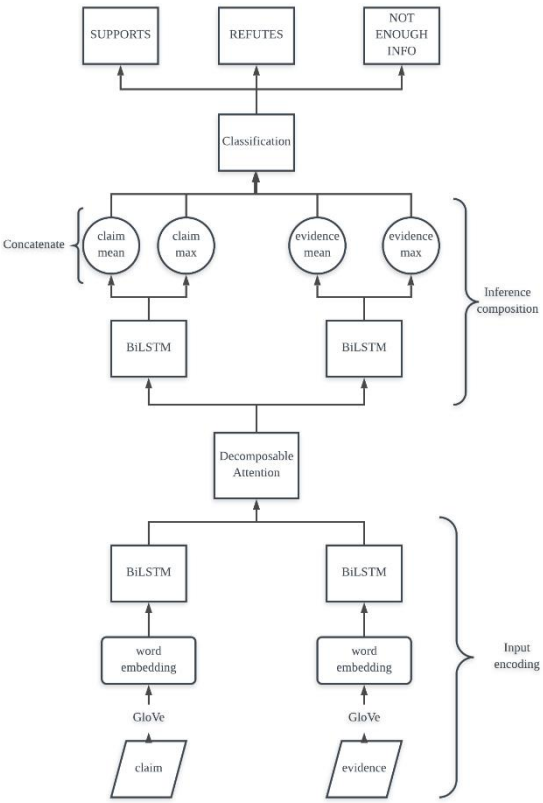

Figure 2; the overview of our ESIM model

The ESIM contains three parts: input encoding, local inference modelling and inference composition. In our application, we used the pre-trained GloVe word embedding. The model is trained by corpus in the Wikipedia with 300 dimensions, so it is relevant to our documents. After embedding the words in the claim and the evidence, they are input into a bidirectional LSTM with 100 hidden layers to get the final encoding. With the encoded claim and evidence, a decomposable attention mechanism is applied to recalculated their vectors. Then, our model will calculate the subtract and multiply value between the first encodings and the recalculated encodings, which will be concatenated together with both

encodings as an input to another bidirectional LSTM to get two new encodings. The max and mean values of the encodings will be calculated and concatenated together as the input of the last classification layer. The final outputs are the scores that the sentence pairs belong to the certain labels.

3.2 Training Data

In order to train our ESIM model, we need to generate a training set. The given training data is used to create a excel file, which contains over 140000 "SUPPORTS" data and 50000 "REFUTES" data. However, we have three labels, the given set can not meet the demand of our model. Random generator is used to retrieve the documents which is totally different from the claim. A more than 2000000 "NOT ENOUGH INFO" data is created by the random generator. Moreover, the training set needs to be balanced to avoid the bias. We randomly select about 50000 data from each label. The component of our training data is shown in the table

SUPPORTS	REFUTES	NEI	total
50000	50000	50000	150000

Table 2: the component of training data. (NEI: NOT ENOUGH INFO)

The string labels are all substituted by the number labels. 0 represents the "SUPPORTS", 1 for the "REFUTES" and 2 for the "NOT ENOUGH INFO"

3.3 Output Process

After training the ESIM model, we need to predict labels for the claims in the test file. The file has been input into the TF-IDF and 2-gram model. It comes out with over 400000 sentence pairs which combine with their titles, pages and keys. The output from our classifier will give each sentence pair a number label. Most of claims will have multiple labels at that time, because there should be non-related evidence which is found by the document retriever. We reform our output into a dictionary. The keys of the dictionary are the claim keys and the values are claims and evidences. We remove all the evidence with the label "NOT ENOUGH INFO" and then count the length of evidence. The left evidence may have two labels, both "SUPPORTS" and "REFUTES". In this situation, we assign the label "NOT ENOUGH INFO" to the claim.

4 Result and Analysis

label acc.	sentence F1	document F1
53.57%	52.57%	90.44%

Table 2: the result of the development set

The development set consists of 5001 claims and the number of each labels is 1667. The result from our system is decent. The label accuracy and sentence F1 score are better than the baseline which is 40% for label accuracy and 20% for sentence F1 score and they are provided by the project creator. The document F1 is the highest score, because our document retriever gets most right titles for the claim. Sentence F1 score is relatively lower than document F1, which may indicate that the number of the evidences which are retrieved from the document may be too large.

	SUPPORTS	REFUTES	NEI
predict	1767	2358	876
real	1667	1667	1667

Table 3: the predict number and real number of each labels for the development set.

The table 3 shows the number of all labels in real development set and the predict one. Judging from the table, we can find that our model tends to classify the evidence into "REFUTES" label. From our point of view, sentence pairs which have "REFUTES" labels are more based on the semantics, while the "NOT ENOUGH INFO" labels are more based on the lexicon. The model will mistake the "NOT ENOUGH INFO" to "REFUTES". It might be caused by following reasons:

- **Word embedding model**: in our ESIM, we applied GloVe pre-trained model as our embedding approach. The model may not contain some words (Obama) and it also has close embeddings for numbers.
- **Training data bias**: some of the evidences in the data do not contain subjects. For example," He remained the team 's starting quarterback …", it might be non-related evidence with the claim "Colin Kaepernick became a starting quarterback …". This kind of data will cause the model not classify the evidence into "NOT ENOUGH INFO" label.
- **Less training depth**: our model is more complicated than the Siamese LSTM. It

consumes a lot of resources. Therefore, in our application, we only train it for four epochs and the hidden layer of LSTM is also set as 100 than 300. This may cause the classification problem.

5 Future work

Our fact verification system can be further improved from two aspects. For the data retriever, adding a filter function in it or return the top k similar documents. These can limit the number of documents being retrieved and can improve the sentence precision. For the model, training data might be slightly changed or checked to avoid those evidences with no subjects. It might be a good way to add the titles before embedding the evidence because it contains more information.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1870–1879. https://doi.org/10.18653/v1/P17-1171 American Psychological Association. 1983. *Publications Manual.* American Psychological Association, Washington, DC.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., & Inkpen, D. 2016. *Enhanced lstm for natural language* inference. arXiv preprint arXiv:1609. 06038.Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503-512.

Mueller, J., & Thyagarajan, A. 2016. *Siamese recurrent architectures for learning sentence similarity.* In Thirtieth AAAI Conference on Artificial Intelligence. Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.