COMP90042 Project 2019: Automatic Fact Verification

Peiyi Wang 870963
Lixia Deng 888216

1. Introduction

In this report, a system was established to automatically identify the authenticity of a claim by searching for evidence in a large Wikipedia text corpus. The task can be divided into 3 sub-tasks:
(1) Retrieving documents in the Wikipedia corpus.
(2) Finding the most relevant sentences associated with the claim in the documents.
(3) Using these pieces of evidence to predict the veracity of the claim.

The main techniques used in this project are inverted index, part of speech (POS), BM25, cosine similarity, word2vec, and deep learning (BERT).

2. Literature review

Reddy, Rocha and Esteves (2018) created a system for factual recognition. They explain that the Term Frequency-Inverse Document Frequency (TF-IDF) vectors of the documents and the named entities (NEs) of the document title are both used to determine whether the document contains the evidence of the claim or not.

A powerful language representation model called BERT gets the latest state-of-the-art results in 11 natural language processing tasks (Devlin, et al., 2018). This model is useful in the fact verification system.

3. Fact verification system

The whole system can be separated into 4 sections sequentially.

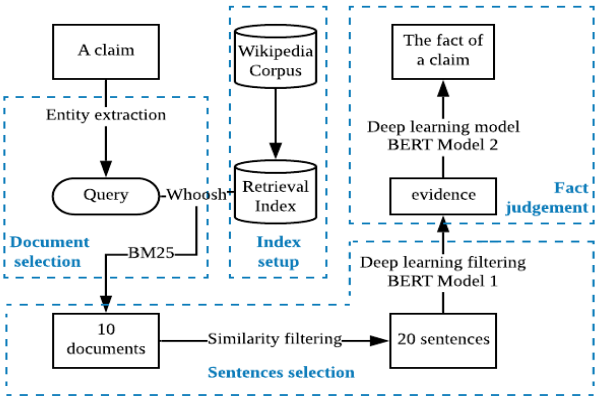

Chart 1: the structure of the system

3.1 Setup retrieval index

The first step is creating an information retrieval system. Compared to forward indexing, an inverted index that maps from content to its locations in a table would be much faster in practice (Song, et al., 2019). In this project, a Python package named Whoosh is adopted for the construction of the inverted index. To be more specific, the rows that share the same Page identifier in the provided wiki texts are considered as a single document to be indexed. Also, the Page identifier itself, and its processed version by replacing the underline with blank are also stored as two attributes of the documents, named "title" and "title_remove_underline".

3.2 Document selection

After the index is set up, the next step is to find the most relevant documents according to claims. To achieve it, two sub-steps are required: transform the claim to the standard query, then search it in the retrieval system.

3.2.1 Entity extraction

In this step, the main goal is to retrieve the right documents as many as possible. Hence, the recall is the only focus. To observe the outcome better, we also revised score.py to extend the calculated identifiers from 5 to 20.

Initially, some Named Entity Recognition tools, such as AllenNLP NER tagger and Spacy NER 101, are applied to identify the main types of entities (Location, Organization, Person). However, these ready-made toolkits have poor performance in recognizing all entities in a claim, which makes the document recall rate very low.

In order to improve document recall, we created some original rules for identifying more entities in a claim: (1) Only proper nouns and nouns begin with uppercase letters should be queried (spacy_pos). (2) Noun chunks without stopwords should be queried (spacy_chunk). Spacy is used to implement these rules.

```
claim : Winnipeg is home to several sports teams.
spacy_ner: []
spacy_chunk : ['Winnipeg', 'home', 'sports teams']
spacy_pos: ['Winnipeg']
```

Image1: Entity extraction

It can be seen that more entity information in a string can be identified using the original rules (spacy_pos and spacy_chunk). To compare their performance, we apply these methods on the same condition (devset.json, search page identifier only, return 10 docs, and one random

sentence for each document). Meanwhile, a group of spacy_pos on different numbers of returned documents are also tested.

Info extraction method	Document Recall	Time cost
spacy_ner	58.80%	625s
spacy_chunk	71.92%	1301s
spacy_pos	**70.53%**	**917s**

Table 1: Recall comparison

It can be seen from the table that NER performs the worst. Chunk is slightly better than POS but costs much more time. Hence, we use spacy_pos method to transfer a claim to standard queries.

3.2.2 Candidate search by the query

With the query keywords, the index made by Whoosh offers 3 ways to search documents: page identifier only, the content of its sentences only, or a combination of them.

Search Type	Doc Recall	Time cost
title only	**70.53%**	**917s**
content only	45.84%	5078s
combination	71.03%	6458s

Table 2: Whoosh search

From the table, a conclusion can be drawn that page identifier, which is the title of a document, is most relevant to the claim. Since searching for contents is time-consuming, we only match the query in the page identifier, then retrieve the whole document including sentences where BM25 (B=0.75, K1=1.2) is adopted as the search method. The model of BM25 is a modification of the TF-IDF with the prior relevance document information addition (Soboleva and Vorontsov, 2019).

Number of returned docs	Document Recall	Time cost
5	66.21%	832s
10	**70.53%**	**917s**
20	73.12%	969s

Table 3: Number of return comparison

Though returning 20 has the best recall rate, it implies much more time to process the sentences inside the document in the next step. Therefore, we only return the first 10 documents as the outcome.

3.3 Sentence selection

In the last part, each query will return 10 documents, which contain hundreds of sentences in total. In this section, we use similarity comparison to choose 20 most relevant sentences for each claim first, and then put them in BERT to train a model that helps to judge whether a sentence could be evidence for a claim.

3.3.1 Similarity filtering

In this step, various methods for calculating sentence similarity are used to identify sentences that support the claims, including the LSI model based on TFIDF, cosine similarity based on word2vec, and Spacy similarity toolkit.

The Gensim library is useful in building TF-IDF or LSI models and implementing word2vec embedding. Besides, Spacy computes cosine similarity based on 300-dimensional GloVe vectors. Both these two libraries are used in this section.

Similarity method	Sentence Recall
gensim_lsi TF-IDF	28.52%
cosine by word2vec	50.75%
spacy similarity	**62.07%**

Table 4: Recall comparison

Comparing with the other two methods, the Spacy similarity has the best performance on sentences recall. Therefore, it is applied to filter out 20 best candidates from all the returned sentences.

3.3.2 Deep learning filtering (BERT)

In order to narrow the scope of returned sentences from 20 to a reasonable number, we utilize BERT to train a model. There are two stages to use BERT: pre-training and fine-tuning (Devlin, et al., 2018). In this report, the system runs fine-tuned training with the BERT-Base pre-trained models.

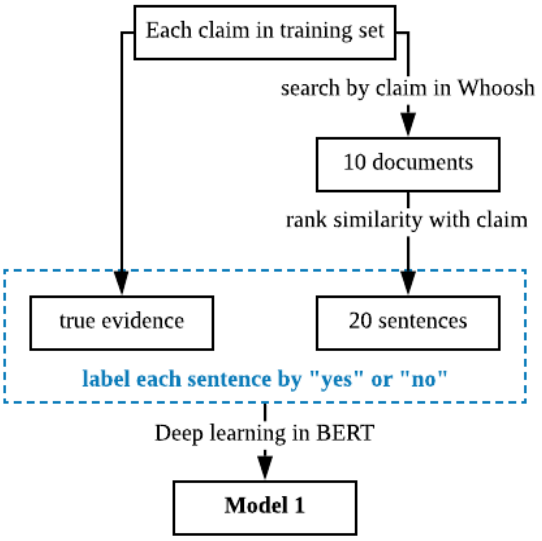

Chart 2: Model 1 for evidence selection

In the training dataset, each sentence presents with its corresponding claim and a label "yes" or "no", which means whether it is true evidence or not. We pick out 15,000 claims in the train.json with near 300,000 sentences returned from the above steps, used as train data. In addition, the devset is used as evaluation data.

```
INFO:tensorflow:***** Eval results *****
INFO:tensorflow:  eval_accuracy = 0.9730543
INFO:tensorflow:  eval_loss = 0.15844645
INFO:tensorflow:  global_step = 27280
INFO:tensorflow:  loss = 0.15843466
```

Image 2: Sentence Selection Training Output

The evaluation on devset shows high accuracy of this model, rendering it feasible to filter useful evidence in returned 20 sentences.

3.4 Fact judgment by deep learning

In this section, BERT is applied to predict the fact of a claim with the obtained evidence corresponding to the claim.

In this model, the training data contains true evidence in the train.json, which is tagged as "SUPPORTS" or "REFUTES". For the claim with "NOT ENOUGH INFO" in the train.json, we applied model 1 to find evidence, making model 2 as a triple classifier model. 30,000 claims from train data are taken as the training set. The devset is used as evaluation data.

```
INFO:tensorflow:***** Eval results *****
INFO:tensorflow:  eval_accuracy = 0.8213141
INFO:tensorflow:  eval_loss = 0.9010432
INFO:tensorflow:  global_step = 2290
INFO:tensorflow:  loss = 0.9010432
```

Image 3: Label Prediction Training Output

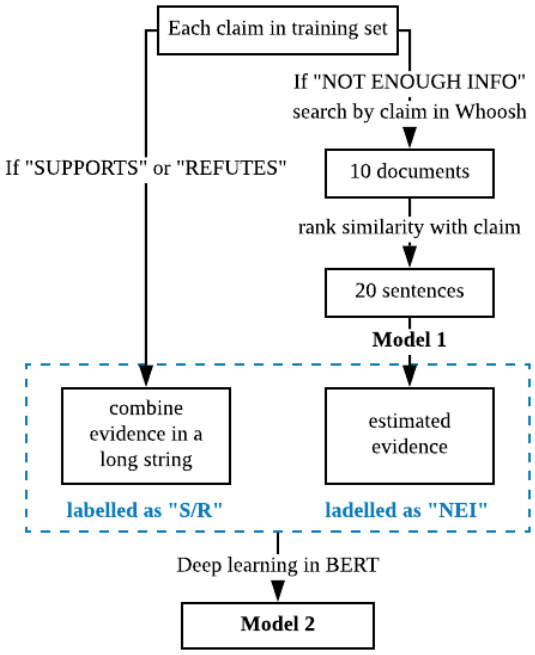

Chart 3: Model 2 for label judgement

After the model 2 is built, we get the fact verification of claims. The whole system is set up.

4. Results

Document			Sentence			Label
Precision	Recall	F1	Precision	Recall	F1	Accuracy
89.27	44.93	59.77	81.38	39.58	53.26	56.66

Table 5: Scores (%) of the final test

From the result, it can be seen that the precision of sentence and document is relatively high. But the recall is unsatisfying. That is, most of the sentences we got after all the filtering are the true evidence. However, over half of the true evidence was not recognized or kept at last by the system. Therefore, it is necessary to analyze possible reasons to improve our system.

5. Error analysis

5.1 Document retrieval

In the document selection part, we use spacy_pos to transfer a claim to query, then search the top 10 documents by title matching. The document recall rate in this step is only 70.53%, which means approximately 30% true evidence for “supports” and “refutes” is lost in this section. To improve it, extending the number of returns could be useful but the time cost will dramatically increase. The best way may be finding an efficient and accurate method to extract really useful information in the claim. For example, taking advantage of a certain corpus

that targets a wiki-style document can be considered if possible.

5.2 Sentence selection

As explained in previous sections, the sentence selection was done by two steps. In the first step, we filtered 10 documents to 20 sentences by spacy_similarity. Before this step, the document recall rate is 70.53%, while it slightly decreases to 70.18% after this step, but the sentence recall is 62.07%. It can be speculated that almost all useful documents were kept, but part of right sentences among them were lost during similarity filtering.

The second step, in this section, is judging whether a sentence belongs to true evidence by the deep learning model in BERT. Although the accuracy shown in the output (see image 2) is 97.3% over the evaluation dataset, the true sentence recall, observed from the final result, decreases to 39.58%, which means around one-third of evidence is labelled as 'no' in the process of BERT. At first glance, these two ratios seem unreasonable. However, 97.3% is the accuracy of labelling 20 sentences. By observing the training data, it can be detected that a claim has 1 to 2 sentences as evidence by average. If the error rate of 2.7% is all from labelling true evidence as 'no', the number of wrongly labelled evidence sentences is approximate 0.54 (=20*2.7%). Divided over 2 sentences, the error rate of labelling true as false is about 27%, which explains the one-third loss of sentence recall. Besides, from the final result, the sentence precision holds a relatively high level (81.38%). Hence, it can be inferred that the model built by BERT is too strict for labelling a piece of evidence as 'yes'. The reason for it could be that sometimes one single evidence sentence cannot support the claim. Instead, a combination of several pieces of evidence supports or refutes the claim together. In this situation, each sentence in the evidence list may not have enough similarity with one claim so that the BERT model will not label them as evidence since the model can only judge a single sentence at a time.

To improve the first step, one way might be training the custom word vectors combined with the pre-trained vectors. For the second step, a potential solution is to take all sentences after document retrieval as the input of the deep learning model when training it, and let the model select useful sentences among them. But it needs a more comprehensive machine learning tool since BERT can only do classifier modelling.

5.3 Fact judgement

In this section, the model 2 built in BERT will label claims which have evidence from the output of model 1. Those who are labelled as "NOT ENOUGH INFO" would eliminate their evidence. But from the test data, it can be found almost no claim is labelled as "NEI" in the model 2, proving that all the sentences conveyed by model 1 is highly relevant to the claim. However, the label accuracy is only 56.66%, mainly due to the low recall. We also noticed that after model 1, 59.6% of claim is labelled as "NEI", while the true value for it should be 33.3% calculated by a dumb test. That means too many claims has no evidence detected by the system. In other words, at least 26.3% mistake of label classification is because of documents selection and model 1. And the rest of accuracy loss should be owing to system error in model 2 itself.

5.4 Summary and enhancement

Above all, the final result implies a precise but not comprehensive choice of sentence made by the system. The most apparent part that needs improvement is the recall of docs and sentences, whose loss can be summarized in the following flow chart.

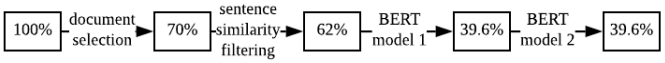

chart 4: The loss of sentence recall

In this flow chart, the largest loss happened in document selection and BERT model 1. We only tried to improve the first section due to the time limitation. As the method of spacy_pos we defined could not extract all the necessary information as needed, we extended the rule to extract pronouns, interjections, adjectives and adverbs in uppercase also, making the doc recall increase from 70.53% to 76.39%. Keeping other parts remaining the same, we applied this method to select documents for the same final test set, all the recall scores increased by about 3%.

Document			**Sentence**			**Label**
Precision	Recall	F1	Precision	Recall	F1	Accuracy
88.74	47.91	62.23	80.76	42.21	55.45	57.54

Table 6: Scores (%) of final result by enhanced method

6. Conclusion

In summary, the Automatic Fact Verification System made in this project has substantial improvements beyond baseline. However, there is still much room for adjustment in the future. The potential solutions to improve the system include: (1) wiki-targeted information extraction, (2) custom pre-trained vectors for sentence similarity, (3) more robust deep learning tool that can deal with evidence selection.

References

Reddy, A. J., Rocha, G., & Esteves, D. (2018). DeFactoNLP: Fact Verification using Entity Recognition, TFIDF Vector Comparison and Decomposable Attention. Retrieved from https://search-ebscohost-com.ezp.lib.unimelb.edu.au/login.aspx?direct=true&db=edsarx&AN=edsarx.1809.00509&site=eds-live&scope=site

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR, abs/1810.04805*.

Song, X., Yang, Y., Jiang, Y., & Jiang, K. (2019). Optimizing partitioning strategies for faster inverted index compression. *Frontiers of Computer Science*, (2), 343. https://doi-org.ezp.lib.unimelb.edu.au/10.1007/s11704-016-6252-5

Soboleva, D., & Vorontsov, K. (2019). Three-stage question answering system with sentence ranking. *EPiC Series in Language and Linguistics*, 4, 18-25.