COMP90042 Project 2019: Automatic Fact Verification System

Saket Khandelwal
Master of Data Science
saketk@student.unimelb.edu.au
Student ID: 1041999

Thanaboon Muangwong
Master of Data Science
tmuangwong@student.unimelb.edu.au
Student ID: 1049393

Abstract

This project task is to develop a system that verify claim based on extracted evidence from text. The main tasks involved in developing this system are document retrieval, sentence retrieval, recognizing textual entailment. In this report, we explain our approaches and decision-making process during this project.

1 Introduction

Nowadays, there are vast misinformation spread on the internet and spread false news. This project is to verify the information is true or false or not have enough information.

This project purpose is to automate the process of fact verification and create a system that can verify the sentence base on the evidence sentence on Wikipedia. The expected output for each claim is 'SUPPORT', 'REFUTE' and 'NOT ENOUGH INFORMATION'. For 'SUPPORT' and 'REFUTE' output, this system needs to be able to extract the evidence that support or refute the claim. In this project, there are 3 tasks that is needed to solve this problem. First Task is **Document Retrieval,** by using claim, it needs to be able to extract the document that contain information about this claim. Second task is **Sentence Retrieval**, from document retrieval, it needs to extract the sentences that are relevance to the claim. Last step is Textual Entailment, after collect relevance sentences. It will predict that whether the claim is 'SUPPORT', 'REFUTE' or 'NOT ENOUGH INFORMATION'. After the claim classified, the next step is to select the most relevance sentences that related to this claim except in 'NOT ENOUGH INFORMATION' case.

The evaluation of this project is to check that our system is able to predict correctly, and the evidence presented is accurate. The "*devset.json*" has been given so that we can build our system based on result check against the devset.json file. In this paper we describe our approaches for **Document Retrieval, Sentence Retrieval and Claim classification** which are the components of our pipeline. We also provide a thorough analysis based on an evaluation of our system.

2 Background

In this section, the methods/tools below are used to complete this project.

2.1 SQLite

SQLite is a library that implement small, fast and robust SQL database engine. SQLite is use for storing Wikipedia documents.

2.2 Coreference Resolution

Coreference resolution is substituting the pronoun with its entity which it is referring to. We use the NeuralCoref[1] library for this task.

2.3 AllenNLP

AllenNLP[2] is an open source Natural Language processing library. We use the library's Name Entity Recognition, Textual Entailment and Constituency Parsing tool.

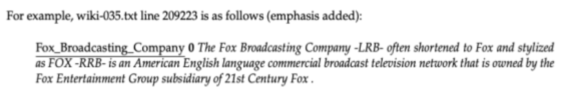
3 Our System

The implementation of our Fact Verification system consists of four main steps. Their implementation is described below.

[1] https://github.com/huggingface/neuralcoref
[2] https://allennlp.org

3.1 Preprocessing

For this project we were provided with wiki-pages where each line consisted of a page identifier, sentence id and the text corresponding to the first two fields. This is illustrated in Figure 1 below.



For example, wiki-035.txt line 209223 is as follows (emphasis added):

Fox_Broadcasting_Company **0** *The Fox Broadcasting Company -LRB- often shortened to Fox and stylized as FOX -RRB- is an American English language commercial broadcast television network that is owned by the Fox Entertainment Group subsidiary of 21st Century Fox .*

Figure 1: Example of corpus text

In order to utilize this corpus for our system we used the page identifiers as documents where each of them could have multiple sentences. We used SQLite to store the documents and its corresponding sentence id and text. We indexed the page identifier column (doc id) for fast querying. Before adding the corresponding text of a document to the database we removed the special characters.

3.2 Document Retrieval

Our document retrieval component involves getting entities of the claims and utilizing the Wikipedia API[3] for searching the top page identifier with that entity and cross checking if that page identifier exists in our corpus or not. One of the reasons of implementing this pipeline is because most of the documents (page identifiers) in our corpus are mostly entities.

In Order to extract entities, we employ the Named Entity Recognition Tool provided by AllenNLP. AllenNLP's NER tool extracts various type of entity tags which is particularly useful in our case. It can properly recognize main type of entities like Location, Organization and Person with addition of recognizing titles of songs, movies, etc. Many of the documents in the corpus are titles of movies and songs. The NER tool outputs entities and their corresponding NER tags. Document names in our corpus were joined together, for example "*Emma Watson"* but the output provided by the NER tool separates each word and tag and returns them which leaves us with a multi-word entity. In order to solve this, we had to merge the words together for each of them. For example, consider a claim consisting of the name *"Emma Watson"*, the NER tool's output for this claim is *'tags': ['B-PERSON', 'L-PERSON'], 'words': ['Emma', 'Watson']*. In order to make this compatible with our data we had to merge this to return the entity *"Emma Watson"*. In addition to this our system also adds tokens to entities depending on the NER tag. Many of the documents in our corpus consist of identifiers like *"Lost (TV series)"*, the NER tool does recognize *"Lost"* as a *"WORK_OF_ART"* tag which is correct but it still would not match our target, to handle this we append the token *"series"* to the word to use for the Wikipedia search. We look for mention of series, film, etc. in the claim to append the appropriate token.

Even though the NER tool is able to capture many types of entities it still is not able to recognize entities in various cases. For example, consider the claim *"Sully is a form of media"*, the NER tool is not able to recognize any entities in this claim. In order to tackle entity disambiguation, we extract the noun phrase (NP) preceding the first verb phrase (VP) in the claim to use it for the Wikipedia search. To extract the NP in a claim, we use the constituency parser provided by AllenNLP which is an implementation based on Minimal Span Based Constituency Parser (Stern et al, 2017) which uses ELMo embeddings (Peter et al,2018). We also append tokens like *'film'* to the NP where the VP include the usage of certain verbs like *"directed by"* which is certainly referring to a noun which is a film.

From the entities extracted from the combination of methods mentioned above we receive a list of candidate entities which we then use the Wikipedia search function to search in order to acquire the top Wikipedia document. The result is then filtered by comparing with the Wikipedia documents in our corpus and the final output of the document retrieval component is generated which are the list of candidate documents.

3.3 Sentence Selection

Our sentence selection component consists of 3 main steps. Firstly, we query all the sentences based on the candidate document list produced by the document retrieval component. Then, to select sentences we use unigram and bigram TF-IDF. Before creating the TF-IDF matrix we first split the sentences in tokens, removed stop words, lower case and lemmatize them. In this abstraction a concatenation of page identifier and its sentence id are represented as documents. Lastly, we extract the top 3 documents which are

[3] https://pypi.org/project/wikipedia/

most similar to the claim by using TF-IDF cosine vector similarity. This candidate list of sentences is then passed onto the Textual Entailment component.

3.4 Textual Entailment

Recognizing Textual Entailment is the process of determining if a certain text (Hypothesis) can be inferred from a provided text (premise). The final process in our pipeline is recognizing textual entailment. In this process the system has to classify a claim as 3 possible labels which are *SUPPORTS, REFUTES or NOT ENOUGH INFO*.

We used the textual entailment (TE) component provided by the AllenNLP toolkit which is a re-implementation of the decomposable attention model (Parikh et al, 2016) which uses pre-trained ELMo embeddings. Our first approach was to train our own neural network based on (Conneau et al, 2018). The idea behind this architecture is to input two sentences and embedded it with encoder. After that it be using fully connect layer to recognize the relation between claim and evidence. However, the result check against development set is poor. The second approach was performing coreference resolution using NeuralCoref on each of the candidate sentences provided by our sentence retrieval module and providing each sentence and claim pair as the input to the TE module, which classifies each pair based on the labels mentioned above. Our system then takes the label with the highest count for each set of sentences and a claim as the final label. We noticed that it did not perform well on the development set. Therefore, we changed our approach where we concatenated the sentences provided by the sentence retrieval module without coreference resolution and used this as an input to the textual entailment component as the premise and the claim as our hypothesis. The system then returns the label classified by the TE component with evidence.

4 Results and Error Analysis

Table 1 shows the results of our system performance on the development dataset and the codalab dataset. As we can observe our system's sentence recall is decent, but the precision is low for both data sets which is the reason for the low sentence F1 score. Our document F1 was high compared to other components score. Our system achieved scores higher than that of the baseline.

Type of metric	Development (score %)	Codalab (score %)
Document F1	74.3	72
Sentence Recall	65.3	62
Sentence F1	42.83	40
Sentence Precision	31.8	29.7
Label Accuracy	45.3	43.8

Table 1: System performance with scores on both datasets

As observed the reason behind our sentence precision being low is because our system returns the top 3 sentences as evidence without taking into account which sentences actually contribute to the classification of the label. This incurs a huge penalty on claims which have only 1 actual evidence which supports the claim. We discuss each component in our pipeline below.

4.1 Document Selection

Many of the claims were classified wrong by our system because the document retrieval component returned a set of documents which were related to **all** the entities in the claim which added a lot of extra noise when inputting sentences to the TE module. As illustrated in Table 2, our system retrieved *"Himalayas"* in addition to "*Bermuda Triangle"* which added extra noise to both the sentence retrieval and TE module which resulted in our classifier to label it as "SUPPORTS" where as it should have been "REFUTES".

Claim: 'Bermuda Triangle is in the western part of the Himalayas.	
Our System: Bermuda Triangle and Himalayas	Actual: Bermuda Triangle

Table 2: Example of Document Retrieval for a claim.

Other cases include our document retrieval component not able to retrieve candidate documents which were not mentioned in the claim. As shown in the Table 3 below, relevant sentences for this claim were associated with the document '*James Taylor'* but our system could not retrieve as it was not an entity mentioned in the claim.

Claim: ‘Hourglass is performed by a Russian singer-songwriter.	
Our System: Russian and Hourglass (Squeeze song)	Actual: James Taylor and Hourglass (James...)

Table 3: Example of document retrieval for a claim not mentioning required entity.

The mistakes by our system for document retrieval could be solved by using a dual approach where we use both the entity extraction method and the TF-IDF method combined to retrieve the top documents relating to the claim.

4.2 Sentence Retrieval

Our sentence retrieval component is also not really reliable as text in the candidate documents contain a lot of extra information which is not relevant the claim but adds extra noise which results in retrieving wrong sentences which are close to the claim as the words mentioned in the claim are also mentioned in the sentences but have no relevance whatsoever. Eg. “*Murda Beatz’s real name is Donald Trump.”*. For this claim our sentence retrieval returned sentences consisting of *‘Donald Trump’* as they were mentioned more in the sentences which resulted in our component ranking *“Donald Trump”* sentences higher. To improve our sentence retrieval component, we could also consider the usage of semantic similarity between the sentences by using word embeddings.

4.3 Textual Entailment

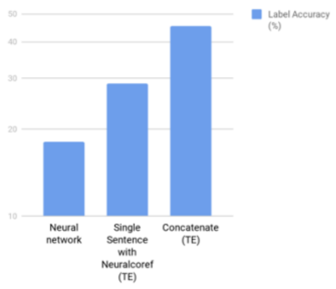


Figure 2: Label Accuracy for each TE approach.

Figure 2 shows the label accuracy for each of our approach mentioned in section 3.4 to recognize textual entailment. We can observe that our final approach yields the highest label accuracy on the development set.

# of sentences	Label accuracy	Sentence F1
5	41	32.7
4	43.5	38.8
3	45.3	42.83

Table 4: Performance of TE component with various number of sentences.

We also tested our TE component by using multiple number of sentences as input. The results illustrated in Table 4 show that our TE component does better as the number of sentences decrease which makes sense as the higher number of sentences add unnecessary noise.

We also observed that for a lot of claims (Eg. “*Justine Bateman is an artist”*) which were labelled as “NOT ENOUGH INFO” in the development set, were classified incorrectly by our system. This is because our sentences retrieval component returned many sentences based on the entities but none of them had relevant information regarding the subject of the claim (Eg. *“artist”*) resulting in our classifier either labelling them as “SUPPORTS” or “REFUTES”.

For further improvement, we could use pretrain AllenNLP model weight to train against our dataset because SNLI comprise of short sentences while our evidence training set is quite long and it should improve accuracy.

5 Conclusion

In this report, we describe our system for fact verification task. The tasks were divided into three major tasks which is given claim find the most relevance documents, given claim find the most relevance sentences from the first part and then classify the claim against those sentences and give an output. We use SQLite to store documents and using NER to extract the most relevance document. For sentence selection, we use cosine similarities take between claim and sentences. After that we concatenate those sentences and using AllenNLP to recognize textual entailment and classify the claim and extract sentences as evidence. Our system performs very well and above given baseline from our requirement.

References

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault and Antonine Bordes. 2018. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *arXiv:1705.02364v5*

Ankur P. Parikh, Oscar Täckström, Dipanjan Das and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. *arXiv:1606.01933v2*

Mitchell Stern, Jacob Andreas and Dan Klein. 2017. A minimal Span-Based Neural Constituency Parser. *arXiv:1705.03919v1*

Matthew E. Peters, Mark Neumann, Mohit Lyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer. 2018. Deep Contextualized word representations. *arXiv:1802.05365v2*