Automatic Fact Verification with Machine Learning

Qijie Li (927249), David Watson(759612)

Abstract

In this paper, we attempt to provide an alternative technique and add research to the Fact VERification (FEVER) shared task. This task is centred around understanding how to validate and provide evidence for fact based claims- which has become a recent global issue. Our approach uses a pipeline system to complete the task encompassing both pre-trained and trained models at different stages. Through various iterations of the model we optimize the parameters- allowing for a system that run efficiently and surpasses the baseline requirements. Using different types of model allows us to conclude that pre-trained models can be used in varying situations, therefore, they can be scaled and hopefully widely used in future language processing tasks.

1 Introduction

Presently, as the level of information being creating has increased exponentially, so has our ability to access and interpret it. However, one perpetual concern is understanding whether the information we consume is fact-based or simply conjecture **(Hanselowski and Gurevych, 2018)**. Therefore, confirming the validity of information is an important challenge, which cannot be attempted manually due to the incredible rate of data creation **(Thorne and Mittal, 2018)**. An alternative approach is to use Machine Learning Fact-verification tools to quickly process data and assess its validity. In order to help stimulate progress **(Thorne and Mittal, 2018)**, the Fact Extraction and VERification (FEVER) dataset was created, which encompasses 185,445 labelled claims based off Wikipedia sentence extracts. Each of these claims contains a set of evidence, which are used to justify the label attached to the claim.

The challenge laid out by the FEVER dataset development team was to design a system that works in a three step process to predict the label and provide the supporting evidence for unlabelled claims. This process involves:

- *Document Retrieval*- Finding the relevant documents to the claim
- *Sentence Retrieval*- Finding the relevant sentences to the claim
- *Verification*- Assigning the correct label for the claim based on the evidence provided.

The effectiveness of a system is critiqued on these three steps, hence, it must retrieve the correct documents, find the correct sentences and label the claims without losing precision throughout.

For our system, we decided to use pipeline approach centred around a pre-trained model, so that we could transfer its knowledge to a new scenario (Jain and Learned-Miller, 2011). The intuition behind this is to understand the scalability of pre-trained models and whether they can be part of an overall solution to verification problems, or if they only work in their prescribed space. Overall, we create an efficiency based pipeline system to break each of the tasks into their specific sub-tasks and provide a scalable design.

2 Pipeline System

The pipeline system, as shown in Figure 1, is broken into three functional stages to address the three stage FEVER task.

2.1 Document Retrieval

In the initial stage of the pipeline, we leveraged an entity-linking technique- tasked with matching the ID's of the Documents to each of the claims **(Hanselowski and Gurevych, 2018)**. The reasoning behind this approach is twofold- it reduces the computational demand by only iterating through entity names, and has proven

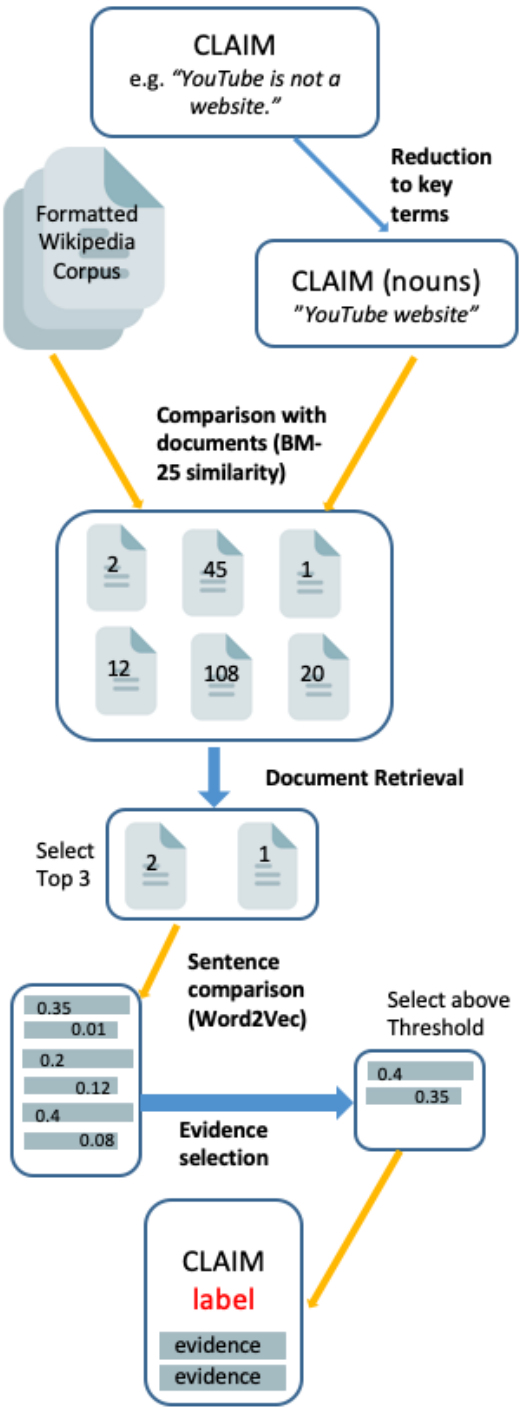

Figure 1: Claim Verification Pipeline

to be an effective trimming technique for large corpus **(Liu and Lu, 2013)**. Additionally, we pre-processed each claim to its nouns- as this provided a better representation of the claim's sentiment. Furthermore, the BM25 similarity index is used to rank the documents, due to its high conversion rate in short-sentence comparisons **(Lv and Zhai, 2011)**. The output of this sub-section is a list of the Top 3 documents that relate to each claim.

2.2 Sentence Retrieval and Evidence Selection

In the second stage, we combined the sentence retrieval and evidence selection tasks in an efficient process. The reasoning behind this approach was intuitive as selecting relevant sentences based on a claim, and assessing whether sentences support or reject a claim are inherently interlinked. Combining this step allowed us to eliminate iterating through all the claims an extra time- thus making the overall pipeline more scalable as a long-term solution.

The focus algorithm of this section is the word2Vec algorithm. Funadentally, Word2Vec takes in an input of words and creates a vector-space for each group-putting those which are similar to each other closer together **(Goldberg and Levy, 2014)**. To apply this model to our scenario we created a dictionary of all the words in the wiki document and ran the claims over this- creating the linking patterns. This model was chosen for it's state-of-the-art word embedding performance **(Goldberg and Levy, 2014)**, and it added diversity to the pipeline in the form of a trained model. The output of this sub-section is a list of sentences which meet a threshold requirement of closeness, and they are considered as evidence for labelling the claim

2.3 Claim Verification

Finishing the pipeline, we use a pre-trained model called AlleNLP textual entailment to interpret our evidence and prescribe a label to the claim. The AlleNLP model works by using local text alignment to understand the relationship between sentences and predict how they relate to each other **(Gardner et al., 2017)**. More specifically, the model will create a vector-matrix of each word in the claim with each word in each tested sentence to understand their relative relationship to each other. AlleNLP then outputs a table such as Table 3- which we use to predict the label of the sentence based on the highest attributed category. Additionally, if there is no evidence moved to this stage of the pipeline, we conclude that there is "NOT ENOUGH INFO" in the corpus, and assign this label to the claim.

The choice to use this algorithm was based on its theoretical high performance **(Parikh and Uszkoreit, 2016)**, and the added diversity it brings to the pipeline- being an untrained

model. Moreover, we confirmed this theoretical high performance level in 2 by using the correct evidences to see if it could provide the correct label for the claim.

3 Results

The results of our system are divided into three parts, document retrieval accuracy, sentence recall and label accuracy. To formalize, our final results are compared to baseline levels and previous iterations.

3.1 Evaluation Metrics

- Document Accuracy (DA)- The percentage of documents that were correctly identified in the evidence.
- *Evidence Recall (F1)*- The percentage of labelled claims that had the correct evidence attached to them.
- *Label Accuracy (LA)*- The percentage of claims that had the correct label attached to them.

	DA (%)	F1 (%)	LA (%)
Random	0	0	33
Iteration 1	38	14	36
Iteration 2	48	20	37
Baseline	N/A	20	40
Iteration 3	52	25	40
Final Pipeline	56	27	44

Table 1: Pipeline Development Results

	LA(%)
Random	33.33
AlleNLP	79.18

Table 2: Performace on AlleNLP

	Probability(%)
Entailment	45.7
Contradiction	4.1
Neutral	50.2

Table 3: AlleNLP Example Output

While our approach to this task was intuitive, our comparative results were not as competitive as we had hoped. Overall we ranked in the top 60% of FEVER Scores and top 50% for Labelling in the competition.

4 Error Analysis

Though theoretically sound, the first iteration of our pipeline failed to meet the baseline requirements. The following is the Error analysis of each section which attributed to increase in performance.

4.1 Document Retrieval

One reoccurring problem with minimising the claim to just nouns is the potential loss of important information. For example:

- *Claim*- "A nomination has been given to A River Runs Through It."
- *Iteration 1*- "nomination River."

As this stage of the pipeline is the most demanding computationally, we could not make any changes that would drastically reduce the efficiency. Therefore, we focused on the most simple, yet widely effective changes that we could, which involved the implementation of additional pre-prossessing steps.

4.1.1 Token selection

Firstly, we noticed that often the key element of the sentence were capitalised in the claims. To exploit this we implemented a capital letter checker. However, this would mean that the first word in the text would always be captured in the formatted claim. Therefore, added a Proper noun tagger to the system, meaning that words which were not nouns or Proper nouns and at the start of the claim were not used in comparisons. For the above example, our Iteration 2 model would pre-process the claim as:

- *Iteration 2*- "nomination A River Runs Through It".

As seen in Table 1,this specific change improved the document retrieval accuracy by 20% leading which translated to small increases in the Sentence Recall and Label Accuracy.

4.2 Sentence and Evidence selection

While there is an efficiency benefit of combining these two stages, there is an accompanying effectiveness cost. This is the result of using all highly correlated sentences, as found by Word2Vec, as evidence for the verification stage without reaffirming them with a second approach. However, we decided not to change

this step, as we wanted to maintain the core design of our pipeline, while improving the performance through parameter optimistaion. Therefore, we enhanced this sub-tasks performance in two areas:

4.2.1 Boundary optimisation

To make the decision as to whether a sentence is used as evidence, there needs to be a boundary point. Our initial reasoning was to use 2 Standard deviations from the mean as this boundary point, therefore only capturing the most relevant and statistically significant sentences. However, this level was too limiting- and led to a poor sentence retrieval score. Through testing on the development set, we optimised this parameter to 1.5 standard deviations, as seen in Table 4.

Boundary point	F1 Score (%)
1.25	16
1.5	19
1.75	17
2	14

Table 4: Boundary optimisation

The overall improvement that this step had on the model can be seen in Iteration 2 in Table 1.

4.2.2 TF-IDF implementation

One reoccurring problem with using a trained version of Word2Vec is its inability to properly interpret words that it has not been trained on. To compensate for this, we added a TF-IDF dictionary which leveraged the sklean package (Pedregosa et al., 2011) to interpret unseen words. While the majority of sentences went through the word2vec model, those that had unseen words now went through a pre-learned model- allowing for a more accurate assessment. This improvement that this had on our system can be seen in Iteration 3 in Table 1.

4.3 Claim Verification

As the final stage of the pipeline is the pre-trained model, and part of our analysis is to assess the ability of these models to adapt to different contexts, we aimed at minimal interference with the underlying structure. Overall, the reoccurring flaw of this sub-system was that it classified too many instances as "NOT ENOUGH INFO". To address this, we used a narrowing selection technique.

4.3.1 Narrowing selection

Originally, when evidence is present in the system, the model uses sentiment analysis to predict the most appropriate bucket for the word. To improve labelling, we introduced a narrowing effect, where, when evidence was presented, the system always had to choose a label of either "REFUTES" or "SUPPORTS". This catered for cases where the model found it difficult to predict the sentiment as non-neutral, but definitely disregarded one of the sides (such as the example in Table 3). The result of this change saw the model surpassing the Baseline in all measurable metrics and completed our final version of the model.

5 Future Development

Overall, we believe that the intuition behind out pipeline model is accurate, and has been proven to work in other examples (Jain and Learned-Miller, 2011). Therefore, to further develop our model in terms of effectiveness, we sacrifice some of the efficient goals. Particularly, we would add a two stage Document Retrieval pipeline- based on the sentence sentiment and the claim sentiment. While this is not plausible for our model, as the run-time would render the pipeline unscalable, it should theoretically dramatically improve our document retrieval process.

Furthermore, for sentence retrieval, we word optimally train the Word2Vec on a larger corpus and use the TF-IDF model in parallel to reaffirm the conclusion on which pieces of evidence were most relevant.

Lastly, we would re-code the AlleNLP text entailment model for our specific and neglect the neutral component. Additionally we would have it iterate over each individual piece of evidence to gauge its validity, rather than the combination of all sentences.

6 Conclusion

In this paper, we have presented a pipeline model to advance the level of understanding surrounding the FEVER task and its boarded implications on fact verification. Though the model was not competitive in the competition, it was ran efficiently and added validity to the idea that combining trained and pre-trained models can provide synergy to a pipeline in a scalable way.

References

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Goldberg and Levy. 2014. word2vec explained: Deriving mikolov et al.s negative-sampling word-embedding method. *arXiv:1402.3722v1*, pages 1–5.

Sorokin Schiller Schulz Hanselowski, Zhang and I. Gurevych. 2018. Multi-sentence textual entailment for claim verification.

Jain and Learned-Miller. 2011. Online domain adaptation of a pre-trained cascade of classifiers. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,*, 1(1):1–8.

Wu Zhong Wei Liu, Li and Lu. 2013. Entity linking for tweets. *eprint arXiv:1803.05355*, 1(51):1304–1311.

Lv and Zhai. 2011. When documents are very long, bm25 fails! 1:1–2.

Das Parikh, Tckstrm and Uszkoreit. 2016. A decomposable attention model for natural language inference. *EMNLP*, 1:1–7.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Christodoulopoulos Thorne, Valchos and A. Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *eprint arXiv:1803.05355*, 1(1):1–20.