Automated Fact Checking for Climate Science Claims

Zhaoning Lu

Abstract

This research focuses on implementing an automated pipeline of fact-checking in the climate science domain. The main methodologies used in this project include Sentence-Transformers, SBERT, and deep neural networks. The highest Evidence Retrieval F-score obtained is 0.14 and the highest claim classification score obtained is 0.54. Based on the result, the pipeline is proven to be feasible.

1 Introduction

Climate change is a highly concerning topic for the past decades. In recent years, the problem of misleading public opinion with false information or unverified statement brings a lot of trouble. Such a phenomenon has a detrimental impact on the society. It is necessary to ensure accuracy and build trust in scientific areas. Based on prior research by [1], indicating that user sharing on social media is the major source of fake news. One promising strategy is to implement automatic fact-checking tools to label information convey to the audience.

The fact-checking pipeline implemented in this paper can be split into two main blocks: evidence retrieval block and claim classification block. The evidence retrieval block will be fed with raw claim texts, and then output relevant retrieved evidence for each of the claims. The retrieved evidence at this step only needs to be in the same topic of the claim, it does not need to be classified into facts like 'supporting', 'disputed', etc. The classification is done in the second block, which uses different models from the first block. The input to this model is a claim along with all retrieved evidence, which will be passed into a BERT network and output a label. Detailed model implementation and structures will be provided.

2 Literature Review

2.1 Word2Vec

Word2Vec is first introduced by Tomas Mikolov et al. in 2013 [2], it has been widely used in the natural language processing domain. The model has the ability to represent words in a dense vector space. The cosine similarity of similar words should be located closer in the space. One key assumption for the Word2Vec model is that words in similar contexts are highly probably to have similar meanings.

2.2 BERT

BERT (Bidirectional Encoder Representation from Transformers) is a state-of-art model used in natural language processing. It is widely used in processing sequential data. As the name suggests, BERT learns bidirectional contextual information. This learning methodology can capture deeper relationships between words. In practice, BERT has been proven to have high performance in text classification, sentiment analysis, question answering, and language translation [3].

2.3 Sentence-Transformers & SBERT

In comparison with BERT, Sentence-Transformers focus on generating more accurate sentence embeddings. BERT focuses more on capturing the deep understanding of individual words in a sentence, while Sentence-Transformers aims to capture the meaning of the whole sentence and produce its vectorized embeddings. Sentence-Transformers are primarily designed for information retrieval, semantic similarity, etc.

SBERT (Sentence-Bert) utilized a very similar structure as Sentence-Transformers. In this project, the Sentence-Transformers is built based on a BERT model, so that it can be reused as a component in the SBERT model, more on this will be illustrated in the methodology section. Based on previous research by [4], SBERT has shown promising results in creating high-quality sentence embeddings.

3 Evaluation Metrics

The evaluation method for the evidence retrieval part is based on F-score, which is commonly used in classification tasks to evaluate the model's performance. The F-score is a harmonic mean of precision and recall, which produces a measure that reflects both precision and recall with equal weight.

$$Precision\ (P) = \frac{TP}{TP + FP}$$

$$Recall\ (R) = \frac{TP}{TP + FN}$$

$$F\ Score\ (F) = 2\frac{P\ R}{P + R}$$

In the above formulas, TP denotes for the number of true positive predictions. FP denotes the number of false positive predictions. FN denotes the number of false negative predictions.

4 Data Analysis

The training data used in this project are claim-evidences paired with their labels {SUPPORTS, REFUTES, NOT_ENOUGH_INFO, DISPUTED}, where the claim and evidence are raw sentences. The training set contains 1,228 instances, while there is a development set that has been separated containing 154 instances. The evidence dataset contains 1,208,827 instances. Handling this huge dataset is a key challenge.

After performing a short statistical analysis on the training dataset, several findings are captured: the minimum number of retrieved evidence for one claim text is 1, the maximum retrieved evidence for one claim text is 5, and the average is 3.4. A similar trend appears in the development set. In terms of sentence length for claims and evidence, the average length is quite close for all datasets (around 20), however, it is noticed that there exist a few numbers of long evidence (around 479 words).

The histogram below shows the sentence length trend on training and testing sets.

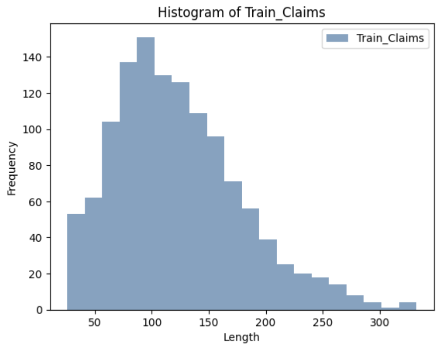

Fig [1]. Sentence Length on Training Set

The result shows the trend of sentence length, most of the claim text falls in the 60-130 range.

5 Methodology

5.1 Evidence Retrieval

There are several methods have been implemented and tested for the evidence retrieval pipeline.

BERT: The first method implemented for evidence retrieval is only based on a pre-trained BERT model. The pre-trained weight used is 'Bert-base-uncased'. This method aims to use BERT to embed raw sentences (claim-evidence pair) into 768-dimensional vectors and forward them to a fully connected layer with dimension (768,1) so that each claim-evidence pair can be predicted to be matched (1) or not (0). To feed raw text into BERT, all the sentences will need to be tokenized and added with <CLS> special token at the start of the token list to indicate the start of a sequence. Correspondingly, <SEP> special token needs to be added to indicate the end of the sequence. To reduce the computational time, all the sentences would be truncated to 40 (if less than 40, it would need to be padded using <PAD>). A standard input to the network is <CLS> + tokens for claim + <SEP> + tokens for evidence + <SEP>. A Data Loader is implemented to process data in a batch. The labelling mechanism is simply to set all positive claim-evidence pairs in the train set to be 1, and randomly sample the same amount of negative pairs for each claim and set it to 0. The embedded 768-dimensional vector would be forwarded to the fully connected layer and compute the loss, finally, backpropagate the whole model. At the prediction step, we can let the model infer unlabeled claim-sentence pairs.

Sentence-Transformers & SBERT: Another method implemented and tested in the evidence retrieval pipeline is based on Sentence-Transformers and SBERT. Since SBERT implemented in this project utilized Sentence-Transformers as a building block, we will first introduce the Sentence-Transformers.

The structure of a Sentence-Transformer is shown below.

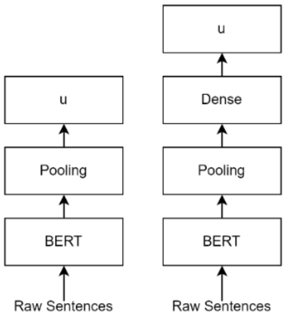

Fig [2]. Structure of Sentence Transformers

As shown in Fig [2], the difference between Sentence Transformer and the BERT method we introduced before is that the Sentence Transformer attached a pooling layer above the BERT layer instead of using a fully connected layer. The output from the Sentence Transformer is u, which is fixed-length sentence embedding. The structure at right shows another possible implementation, which adds a dense layer on top of the pooling layer to reduce output dimensions. The pooling layer plays a critical role here, it helps to capture the relevant information from the word level and forms a larger understanding at the sentence level. Several pooling methods can be selected: {mean pooling, max pooling, cls pooling, etc.}, in this project, we used mean pooling. The weight for the BERT model is 'bert-base-uncased'. At the first training step, we set all positive pairs in the training set with label 1 indicating they are related. And randomly generate a slightly higher number of negative pairs with label 0 for each of the claims. After the first training step, we calculate the cosine similarity and collect all the top k wrongly classified evidence for each claim based on the similarity score. Those collected top k negative samples will be saved and used in the next training step. At this time, we will use the collected top k negative samples along with other newly generated random negative samples to be the negative set. This process can be repeated multiple times. The negative sampling strategy is a key step to ensure the Sentence Transformer produces high-quality sentence embeddings. After repeating this process, we will do a prediction on the test set and save the top 4 pieces of evidence for each of the claims based on the cosine similarity score.

SBERT utilized most of the same ideas as Sentence Transformers, the following figure shows the structure of SBERT.

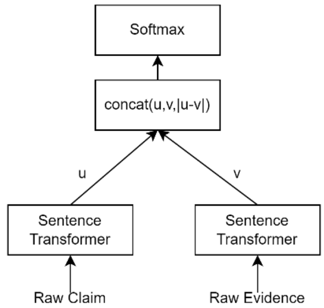

Fig [3]. Structure of SBERT

The main difference between SBERT and Sentence Transformer is SBERT concatenates embedding of the two sentences and the absolute difference between them together and then forwards to a SoftMax layer. The training strategy is the same as Sentence Transformer, where we utilized the same idea of negative sampling.

5.2 Claim Classification

The method used for the claim classification block is by attaching several fully connected layers on top of the BERT tokenizer layer. We label {SUPPORTS, REFUTES, NOT_ENOUGH_INFO, DISPUTED} to {0, 1, 2, 3} respectively. During training in the tokenizer layer, we first attach each of the evidence to the claim text, then separately tokenize each of the text connected with <SEP> special token. After that, we obtained fixed-length input token sequences by padding or truncating. The sequence will be fed into the BERT layer and converted to a 768-dimensional vector. The vector is then passed into a (768,768) linear layer and a (768,4) linear layer, with Tanh nonlinear activation function in between. We also used warmup steps to facilitate training. The loss function used is Cross-Entropy, and the optimizer is Adam.

6 Results & Discussion

6.1 Evidence Retrieval Results & Discussion

During execution, it turns out the evidence retrieval block is more challenging compared to the claim classification block. The main reason is memory usage and time consumption. For the BERT method we introduced in section 5.1, the F-score on the development set is 0.07, which is quite low.

By analyzing the similarity score we obtained for claim-evidence pairs, it turns out most of the score is located close to 0.5. Intuitively, this fact means the model very lacks confidence when making predictions. Also, it is noticed some completely irrelevant pairs could also result in a very large similarity score. The possible reason behind this may be because directly using a (768,1) layer to connect the BERT layer cannot gain a proper sentence-level understanding.

The results we obtained from the Sentence Transformer method introduced in section 5.1 is much more promising. On the development set, the F-score is 0.16, and 0.14 on the public Coda Lab test set. The improvement could come from two aspects: Firstly, in the Sentence Transformer method, we used a mean pooling layer to facilitate the model gaining deeper sentence-level information. Another reason is the negative sampling strategy. During execution, after adding the generated negative samples and continuing fine-tuning the model, the increasing trend of the F-score on the development set is observed clearly.

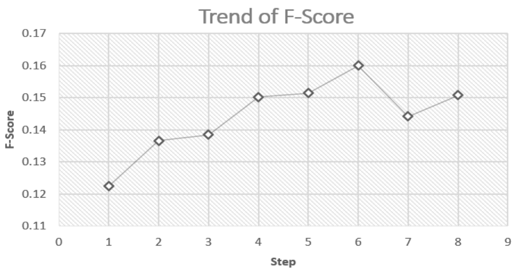

Fig [4]. Effect of Adding Negative Samples

Figure 4 illustrates the effect of the negative sampling method. Note that the 'step' is a different concept from the 'epoch', where 'step' denotes the whole process of adding newly generated negative samples and retraining the model. The intuition behind this is that we want the model to learn from a 'harder' domain and get some new 'knowledge'.

In terms of time consumption, training the model for 1 epoch on GPU would normally take 2 minutes, encoding all the sentences including claims and evidence would normally take 20 minutes, and computing similarity on each of the claim-evidence pairs would take hours.

Due to time limitations, the SBERT model is treated as future work.

Table [1]. Evidence Retrieval F-Score

Metrics	BERT with FC	Sentence Transformer
F1 (Dev Set)	0.07	0.16
F1 (Test Set)	0.05	0.14

Table 1 illustrated F-Score we obtained based on the two methods. It is obvious that the Sentence Transformer has a significant improvement.

6.2 Claim Classification Results & Discussion

Based on the development dataset, the accuracy of the model introduced in section 5.2 achieved 83% accuracy. This promising result validates the correctness of the model. During implementation, we also tried different activation functions. Based on experimental data, the Tanh normally achieves the highest accuracy, while ReLU is the fastest with a little drop in accuracy. The accuracy from the test set shows a gap to the development set (~50% accuracy on the test set). The main reason behind this should be we would expect to have wrongly retrieved evidence (noise) in the data pairs, while we did not hold the same condition in the development set. One possible way to improve the model is by introducing noise while training, which can be future work.

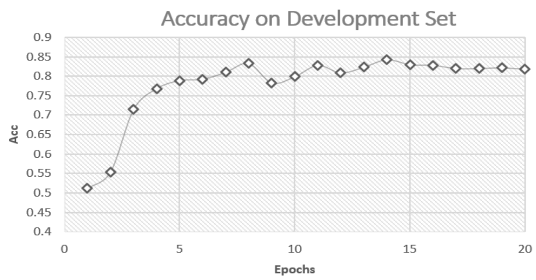

Fig [5]. Claim Classification Accuracy

As shown in the above figure, the accuracy of the development set converged in 20 epochs.

In conclusion, the introduced pipeline to perform automated fact-checking has been proven to be feasible based on experimental data. There are also several possible future works with great potential been illustrated.

References

[1] Koch, T. K., Frischlich, L., & Lermer, E. Effects of fact-checking warning labels and social endorsement cues on climate change fake news credibility and engagement on social media. *Journal of Applied Social Psychology*. https://doi.org/10.1111/jasp.12959

[2] Mikolov, T. et al. (2013) Efficient estimation of word representations in vector space, arXiv.org. Available at: https://arxiv.org/abs/1301.3781

[3] Devlin, J. et al. (2019) Bert: Pre-training of deep bidirectional Transformers for language understanding, arXiv.org. Available at: https://arxiv.org/abs/1810.04805

[4] Reimers, N. and Gurevych, I. (2019) Sentence-bert: Sentence embeddings using Siamese Bert-Networks, arXiv.org. Available at: https://arxiv.org/abs/1908.10084