Fact Verification and Extraction of Climate-Related Claims

Damian Curran

Abstract

Fact verification of claims is a highly studied topic. It is particularly important in the realm of climate change. In this paper, I show that a pipeline approach using evidence shortlisting, ranking and classification can obtain high ranking fact verification performance on the custom climate change claims dataset.

1 Introduction

The ability to verify the accuracy of a claim is a highly sought after functionality in an age of rising online misinformation. In this project, a custom dataset has been produced for analysis which contains climate change claims which can be verified by reference to a large corpus of evidences. In this report, I detail my approach to solving this problem using a pipeline approach.

2 Related Work

This challenging task of fact verification is often conducted by reference to Fact Extraction and VERification (FEVER) labelled datasets. The original FEVER dataset was designed to test methods for fact extraction and verification of claims (Thorne et al. 2018). Other datasets in different domains but with a similar structure have been released since, including SciFact (Wadden et al. 2020), Fever Symmetric (Schuster et al. 2019) and Climate Fever (Diggelmann et al. 2020).

3 Method

3.1 Pipeline Method Overview

It is common to approach FEVER tasks as a three-step pipeline problem. Those steps typically are document retrieval, sentence retrieval and claim verification (Bekoulis, Papagiannopoulou, and Deligiannis 2020).

However, this exercise differs from other FEVER problems because the 'knowledge base' does not consist of entire articles (which must then be separately parsed to identify the evidence), but rather a single collection of short 'evidences'. Therefore, the actual steps adopted in this report are slightly different from the standard FEVER approach in order to account for the amended dataset. Each step is discussed in the sections below.

3.2 Development Metrics

During development and fine tuning of the pipeline, I trained on the 'train' set and analysed performance on the 'dev' set. The dominant performance metric I sought to maximise in the first two steps (Shortlisting and Evidence Ranking) was macro recall of evidence. This was the appropriate metric because any failure to recall evidence in an early step will necessarily reduce performance in a subsequent step because of pipeline error propagation. Label accuracy and confusion matrices were used to assess performance in the final step of Claim Classification.

3.3 Shortlisting

The first step I refer to as "shortlisting". The aim of this step is to obtain a 'shortlist' of evidence candidates for each claim. This is done by reducing the evidence corpus (containing over 1 million instances) to a workable pool of likely candidates for each claim. This list is then passed forward for evidence ranking.

This shortlisting is a practical necessity because of the excessive computational resources and time that would otherwise be required to analyse all evidence-claim pairs across the knowledge base during evidence ranking. I also found during testing that top-5 recall in the subsequent stage is

greatly improved if the evidence candidate pool has been reduced by this step.

I used two separate feature engineering approaches for this task and then combined the results to obtain the final shortlist for each claim. Both approaches assume that evidence-claim pairs have high similarity across a range of features.

The first approach used sentence embeddings. Sentence embeddings were obtained for each evidence text with the HuggingFace `'distilbert-base-cased'` model. [1] This model type was selected because it is designed to be fast and lightweight. These are desirable qualities when processing a very large initial set but where embedding richness is not a priority. I found that the cased model produced higher recall than the uncased model.

Sentence embeddings of 768 dimensions were obtained for each evidence and claim text separately. Each evidence-claim pair was then ranked by cosine similarity and the top-10,000 evidences for each claim, in similarity order, were stored for further processing.

The second approach used a bespoke noun & entity retrieval method. For each evidence text, I obtained a set of all nouns, pronouns, and certain named-entities[2] in the string. I made a judgment call on those entity types that I thought would be relevant to climate facts, such as geographies and organisations. Standard `SpaCy` tools were used for the extraction. Similarity between each evidence-claim pair was measured by the length of the set overlap. The top-10,000 evidences for each claim, in similarity order, were stored.

The two ranked top-10,000 lists were then combined into a single 'consolidated' list by alternating between each list and popping the top item from each, skipping over duplicates which had already been selected.

Similarity Method	recall (n=500)	recall (n=10000)
'distilbert' embeddings	0.454	0.861
noun_entity overlap	0.511	0.812
consolidated	0.621	0.928

Table 1: Recall of correct evidences in the dev set within the top-n items in shortlists.

As can be seen in Table 1, the consolidated list produced higher recall on the dev set than either of the individual methods alone. This suggests that they are identifying different sentence characteristics which are relevant to similarity. I hypothesise that the embedding list better identifies contextual or thematic similarities across the entire string, but may not recall all exact string matches on key terms, whereas the noun_entity list has opposite emphasis. The combined list then takes the best matches from each approach.

Figure 1 shows the number of evidences in the dev set recalled within each 500-wide bin by each method. The consolidated method clearly has superior performance. Notably, the vast majority of evidences are recalled in the top-500 items, with the number of recalled item dropping significantly thereafter. This observation is relevant to the 'cut-off' point, *N*, used in the subsequent evidence ranking step.

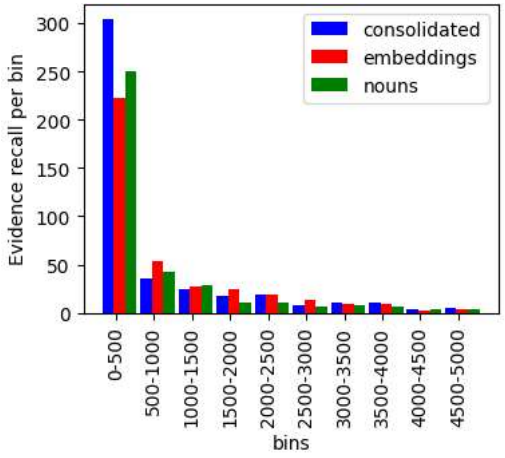

Figure 1: Recall count of evidences in dev set in bins (w=500) for each shortlist method (where dev set had total n=491 evidences).

3.4 Evidence Ranking

The next step is to select the top 5 most relevant evidences from the top-N ranked evidence from the shortlist for each claim. Five evidences are selected at this stage because it is the maximum number of evidences that are present for each claim in the training and dev sets.

This is treated as a binary classification problem for each evidence-claim pair, being 'relevant' or

[1] `https://huggingface.co/transformers/v3.3.1/pretrained_models.html`

[2] `"DATE", "EVENT", "FAC", "GPE", "LAW", "LOC", "NORP", "ORG", "PERSON"`

'not relevant'. I fine-tuned a HuggingFace `'roberta-large'` model with a binary classification head for this task.[3] During training, concatenated evidence-claim pairs were encoded and the `'CLS'` token embedding fed into the classifier. Every evidence-claim pair in the training set was labelled 1 ('relevant'). Irrelevant evidences, labelled 0, were also sampled for *each claim* as follows:

i. Two "hard negative" samples. These were randomly selected from the 500th to 1500th position in the shortlist ranking from the first stage for that claim;
ii. Two random samples from the evidence corpus.

The above 'pointwise' approach has been shown to be effective in similar FEVER tasks, as has hard negative mining (Soleimani, Monz, and Worring 2019). The hard negative mining aims to train the model to separate highly relevant evidence-claim pairs from less relevant (but still related) pairs. During development, I tested the recall of this trained model on several combinations of 'hard' and 'random' negative training pairs. I found that recall was greatest with 2-2 respectively. I also found that 'harder' pairs (ie. those higher up the shortlist ranking) produced better model recall (although that increased the risk that I might erroneously sample a relevant evidence during training).

The final model was trained for 2 epochs with a learning rate of 1e-05. I found that training for additional epochs reduced performance on the dev set. At inference, each claim-evidence pair was run through the model and ranked in descending order of confidence of label 1 (relevance). The top-5 were selected for each claim and stored for processing in the subsequent step.

Only the top *N* evidences from the shortlist were paired and tested in this way at inference. Although I had stored a shortlist of the top-10,000, these were not all used during inference. As can be seen in Figure 1, the recall density is much greater at the top of the list (0-500) than end. The selection of the appropriate N value was therefore a trade-off between the number of evidences (high-N) and density of evidences (low-N). After testing various values on the development set, the 'N' threshold with the best performance was found to be 500.

Hyperparameter	Experiments	Final Selection
Negative samples per positive sample	1, 2, 5	2
Negative sample index range from shortlist	500-1500, 5k-10k	500-1500
Training Epochs	1,2,5	2
N	400, 500, 1000, 2000	500

Table 2: Hyperparameters tested on dev set during development of Evidence Ranking model, and final selection of optimum parameters.

Table 2 shows the hyperparameter ranges tested during the development of the Evidence Ranking model, and the final selection which produced best performance measured as highest recall of evidence for each claim in the dev set (being a dev set best macro recall of 0.281). This metric is subject to fluctuate with each training run, as the negative samples are randomly selected during training.

3.5 Claim Classification

The final step is to take the top-5 ranked evidence samples for each claim and, from them, assign a label to the claim and select the final evidences. This process involves two sub-tasks.

Individual labels applied to the top-5 evidences for each claim	Final claim label given to the claim	Final evidence selected for the claim
S and (opt) NEI	S	S (only)
R and (opt) NEI	R	R (only)
NEI (only)	NEI	All 5x NEI
S (>1) and R (>1) and (opt) NEI	D	S and R (but no NEI)

Table 3: Assumptions made during Claim Classification as to label of evidence-claim pairs and overall label.

The first sub-task is to label each claim individually. This was treated as a ternary classification problem with three categories: SUPPORTS ('S'), REFUTES ('R') and NOT_ENOUGH_INFO ('NEI'). The training data only provides only one label per claim. Therefore,

[3] `https://huggingface.co/transformers/v3.3.1/pretrained_models.html`

certain assumptions needed to be made about the structure of the training data before individual evidence-claim pairs could be labelled for training. The assumptions I made are set out in Table 3.

I fine-tuned a HuggingFace `'roberta-large-mnli'` model with a ternary classification head for this task.[4] During training, each top-5 evidence-claim pair was fed into the model and, with the exception of the DISPUTED claims, given the same label as the claim. The DISPUTED claims were omitted altogether from the training set for this step because, as shown in Table 3, I have assumed that DISPUTED claims contain both a SUPPORTED and REFUTES evidence-claim pair. As there was no way to readily distinguish these, they were omitted from the training set.

	D	NEI	R	S
D	7	2	1	9
NEI	5	10	6	20
R	9	4	9	5
S	8	1	2	57

Table 4: Confusion matrix for optimum parameters on dev set

I used confusion matrices to examine the classification results. Table 4 shows the final confusion matrix on the dev set using optimum parameters. It shows adequate performance. I initially found that at inference, the model commonly misclassified REFUTES claims. This is likely due to this class being under-represented in the training data compared to the other classes. I experimented with adding multiple copies of the REFUTES claims into the training dataset to address this. I also experimented with bootstrapping the DISPUTED claims by first training a preliminary model and then classifying the DISPUTED claims with pseudo-SUPPORTS or REFUTES labels, and then re-training on the expanded dataset. However, neither alternate technique notably improved classification accuracy on the dev set. The best performance and that of the final model was obtained by simple 2 epochs with a learning rate of 1e-05.

At inference time, each evidence claim pair in the test set was classified as one of the three labels based on the class with the maximum prediction score. The overall claim label was then determined by majority vote and the final evidence pool selected according to the assumptions in Table 3.

[4] `https://huggingface.co/transformers/v3.3.1/pretrained_models.html`

4 Results

The results of my pipeline applied to the dev set (fine tuned on the train set only) and the test set (fine tuned on both the dev and train set) are shown in Table 5.

	dev set	test set
Evidence Retrieval F-score (F)	0.225	0.135
Claim Classification Accuracy (A)	0.538	0.493
Harmonic Mean of F and A	0.317	0.212

Table 5: Metrics for final model performance on dev and test sets (on Codalab).

I ranked 83 (of 367) on the final Codalab rankings. My comparative results indicate that the pipeline works well and that the methods I adopted were sound. However, there was a notable drop in performance from the dev set, to the initial ranking (~21st), and to the final test set ranking. I only submitted 1 final entry for the Codalab testing (of 4 total), so this drop is not due to overfitting on the test set. I hypothesize that the performance drop is due to randomization in the training methods, and in particular the random negative sampling in Evidence Ranking step. Future improvements in my pipeline would involve repeated training of models and selection of the best performing model for a given set of hyperparameters, a reduction of the scope for discrepant performance due to randomization (such as reducing the size of the window of hard negative samples), and testing on a wider range of hyperparameters.

5 Conclusion

I have shown that the pipeline method performs well when each step optimized to its specific task. In the shortlisting step, I used a combination of entity recognition and the light weight `distilbert` embeddings, which provided high recall. The evidence retrieval step used an optimized `roberta-large` model in a binary classification task on the top 500 evidences for each claim, greatly improved by hard negative mining. The final step was claim classification, which involved fine tuning a `roberta-large-mnli` model with a ternary classifier.

References

Bekoulis, Giannis, Christina Papagiannopoulou, and Nikos Deligiannis. 2020. “A Review on Fact Extraction and Verification.” doi: 10.48550/ARXIV.2010.03001.

Diggelmann, Thomas, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. “CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims.” doi: 10.48550/ARXIV.2012.00614.

Schuster, Tal, Darsh J. Shah, Yun Jie Serene Yeo, Daniel Filizzola, Enrico Santus, and Regina Barzilay. 2019. “Towards Debiasing Fact Verification Models.” doi: 10.48550/ARXIV.1908.05267.

Soleimani, Amir, Christof Monz, and Marcel Worring. 2019. “BERT for Evidence Retrieval and Claim Verification.” doi: 10.48550/ARXIV.1910.02655.

Thorne, James, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. “FEVER: A Large-Scale Dataset for Fact Extraction and VERification.” Pp. 809–19 in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics.

Wadden, David, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine Van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. “Fact or Fiction: Verifying Scientific Claims.” Pp. 7534–50 in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics.