COMP90042 Project 2023: Automated Fact Checking For Climate Science Claims

1144831

Abstract

Guided by the strong performance of BERT (Bidirectional Encoder Representations from Transformers) models, this paper tailors the pre-trained language models to form the crux of an automated claim verification tool. We opted for an unsupervised evidence retrieval algorithm using embedded vectors of evidences, and a supervised approach to adapt the pre-trained BERT model to our task-specific classification of claims. Additionally, this work also includes an extensive comparative analysis using multiple evaluation metrics to determine an optimum model.

1 Introduction

Social media has enabled people to post information, facts and public data on the internet with ease, without any regulation or verification (Bakshy et al., 2012). However, misinformation in topics like climate change, scientific claims and global news can create havoc and panic in the society (Lazer et al., 2018). The volume of claims that is generated on the internet calls for the need of a verification system in place to cross-check data regarding sensitive matters. One plausible solution is an automated fact checking model that has been developed to retrieve evidence(s) for claims made on climate change, and use the evidence to verify the claims. How can we leverage pre-trained language models (Min et al., 2021) to build a fact-checking system that can capture legitimate evidences to accurately verify any claim related to climate change? - this question is what drives the discussion of this paper.

2 Literature Review

In 2018, (Devlin et al., 2018) introduced the pre-trained language model, Bidirectional Encoder Representations from Transformers (BERT). BERT has been intensively investigated in Natural Language processing (NLP) recently, and has proven to perform significantly well (Tenney et al., 2019; Gao et al., 2019). This provided a strong motivation to explore BERT to solve our problem statement.

The usage of TF-IDF (Term frequency - Inverse Document Frequency) feature vectors is a significantly common approach for sentence retrieval algorithms. However, the information theory of BM25 (Kadhim, 2019) and the usage of other embedding vectors (Zoupanos et al., 2022) have been vigorously investigated in recent years by a number of writers.

The scoring function by BM25 (Kadhim, 2019) for a potential evidence E, given a query C (a climate science claim, in our context), is defined as:

$$score(E, C) = \sum_{i=1}^{n} IDF(c_i) \frac{f(c_i, E)(k_1 + 1)}{f(c_i, E) + k_1(1 - b + b\frac{|E|}{avedl})} \quad (1)$$

Equation (1) has been leveraged in the comparative assessment of this work after thorough analysis of its performance as compared to TF-IDF feature vectors.

The analysis in a closely related topic by (Soleimani et al., 2020) provides a valuable insight for our research topic. (Soleimani et al., 2020) model a two-layer BERT pipeline (Figure 1) where the first BERT layer takes documents in the input and retrieves sentences in the output that act as the evidence(s) for a given claim. The second BERT layer takes these evidences as the input along with the claim, and classifies the claim-evidence pair. The final label for a claim is drawn by comparing the independent decisions for each evidence linked to the claim. Drawing inspiration from their work, one of our proposed solutions also uses a stacked BERT architecture (further explained in Section 4.1).

3 Method

To design a fact checking model, we finalized a system consisting of two key components. An

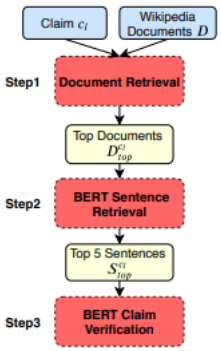

Figure 1: Three-step pipeline evidence extraction and claim verification by (Soleimani et al., 2020)

evidence retrieval component that retrieves the top 5 evidences for any given test claim, and a *claim-evidence verification classifier* that classifies a given claim-evidence pair into one of the four categories : SUPPORTS, REFUTES, NOT_ENOUGH_INFO and DISPUTED.
We implemented our research methodology in Python, and deployed the code-base on Google Colab to leverage the GPUs for data preprocessing.

3.1 Data Description

The dataset provided for the scope of this project consists of 1228 labelled claims as training data, 154 labelled claims as validation (or development) data and 153 unlabelled claims for testing. Each labelled claim has the following format:

```
{
    <claim-id>: {
    "claim-text" : <claim text>
    "claim_label" : <claim_label>
    "evidences" : [<id1>, <id2> ..]
    }
}
```

The list of IDs under evidences are the identifiers for evidences provided as the knowledge source (evidence.json) of this project.

Since each claim entity currently has a list of evidences , we "exploded" the *evidences* column, so that each row consists of a claim (id and text), and an evidence ID, using the DataFrame.explode() function from *pandas* (an open source, Python BSD-licensed library).

3.1.1 Evidence Retrieval

For a given unlabelled (each claim-id only has a claim-text) test claim, the first aim was to generate a list of evidences. For this, we implemented a semantic search using *Facebook AI Similarity Search (FAISS)* (Johnson et al., 2019) on embedding vectors of the evidences provided in the knowledge source. Using this embedding corpus of evidences, we extracted the top 5 evidences for each claim. This led to the generation of a new test data frame, consisting of 5 rows for each test claim (one row for each claim - potential evidence pair).

The embeddings dataset was created using the *AutoTokenizer* tokenizer with a pre-trained AutoModel transformer. The model uses "bert-base-uncased" for semantic search. We opted for the popular approach of using the last hidden state of the [CLS][1] token to create embeddings for each evidence. The embeddings of the evidence texts are indexed using Dataset.add_faiss_index() function, which is essentially used to retrieve the top 5 evidences for each claim (Refer the code block in Figure 2).

```
claim_embedding = get_embeddings([claim]).cpu().detach().numpy()
scores, samples = embeddings_dataset.get_nearest_examples(
  "embeddings", claim_embedding, k=7
)
samples_df = pd.DataFrame.from_dict(samples)
samples_df["scores"] = scores
samples_df.sort_values("scores", ascending=False, inplace=True)
for _, row in samples_df.iterrows():
  evidence_id = row.evidence_id
  new_row = {'claim_id':claim_id,'claim_text': claim ,'evidences':evidence_id}
  new_test_data.loc[len(new_test_data)] = new_row
```

Figure 2: Code snippet explaining the iterative retrieval of evidences after FAISS indexing

3.1.2 Claim Label Classification

The second stage of this claim verification system was to classify the test claims into one of the 4 possible labels: SUPPORTS, REFUTES, NOT_ENOUGH_INFO and DISPUTED. A pre-trained BERT model is adopted for this stage of the solution. The BERT model uses *BertForSequenceClassification* with "bert-base-uncased" as the underlying model. It is trained on the provided training data and validation data, as a 4-class classifier with a batch size of 32. The learning rate of the BERT model was 2e-5, trained over 3 epochs. The evaluation of the BERT model on the development set (dev-claims.json) is reported in Table 1.

During evaluation on test data, we used the results from the first FAISS based transformer to

[1] [CLS] is a special classification token used in the input formatting for transformers like BERT.

Epoch	Average Training Loss	Validation Accuracy	Validation Loss
Epoch 1	0.51	0.74	0.50
Epoch 2	0.44	0.77	0.50
Epoch 3	0.39	0.76	0.48

Table 1: Performance and error analysis of BERT model on development set

form the embeddings that act as inputs to the second pre-trained BERT model. In the testing phase, a softmax function was applied over the logits (outputs). The predicted labelled was calculated using the index of the highest probability value for a given logit vector (Refer Figure 3). The final block of the algorithm determines the label for a test claim by aggregating the values of all predicted labels for a claim, and classifying the test-claim by prioritizing evidences of SUPPORTS, REFUTES and DISPUTED (Figure 4).

```
logits = logits.detach().cpu().numpy()
softmax_pred = softmax(logits)
new_preds = (softmax_pred == softmax_pred.max(axis=1, keepdims=1)).astype(float)
```

Figure 3: Code snippet explaining the flow of predictions in the evaluation step

```
for id in claim_id_list:
  claim = newest_test.loc[newest_test.claim_id== id, 'claim_text'].values[0]
  evidences = newest_test.loc[newest_test.claim_id== id, 'evidences'].values
  labels = newest_test.loc[newest_test.claim_id== id, 'claim_label'].values
  if 'SUPPORTS' in labels:
    label = 'SUPPORTS'
  elif 'REFUTES' in labels:
    label = 'REFUTES'
  elif 'DISPUTED' in labels:
    label = 'DISPUTED'
  else:
    label = 'NOT_ENOUGH_INFO'
```

Figure 4: Algorithm of Claim Classification using the output from the first neuron of the last BERT layer

4 Results

For the evidence retrieval stage of the models, the *recall value* was adopted as an important evaluation metric. We are attempting to reduce the false negative rate while retrieving the evidences, since a prediction is true only if the entire set of evidences is retrieved.

The architectural design for the final implementation of the model has been visualized in Figure 5. This flow was finalized after multiple other design decisions were implemented.

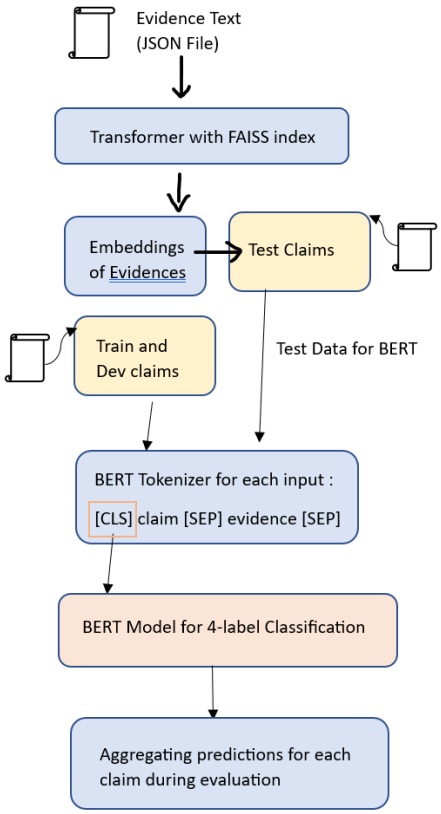

Figure 5: Flow of control of the Evidence Retrieval and Claim-Evidence Classification steps

4.1 Comparative Analysis

The following designs were compared in the scope of this project:

- *Model 1: BM25 Ranking for Evidences and BERT for 4-label Claim Classification*

  This design implemented a basic retrieval algorithm that uses Okapi BM25 as a ranking function on the entire evidence corpus. For each test claim, we get the top 7 results from the ranking function as evidences. A pre-trained BERT model (as explained in section 3.1.2) was used to determine the label for the test claim.

- *Model 2: BM25 for potential evidence filtering, BERT for Evidence Retrieval and BERT for 4-label Claim Classification*

  This design is an extension of the first model design, where we used BM25 as a filter that provides input to a BERT model that classifies if a potential statement is an evidence for a claim or not. A second BERT model is stacked on top to perform the claim verification using the 4-label classifier.

- *Model 3: Facebook AI Similarity Search (FAISS) for Evidence retrieval and BERT for 4-label Claim Classification*

  We then experimented using a pre-trained transformer to create an embedding dataset of the evidence corpus for the evidence retrieval. Using the FAISS index, the 5 nearest evidence embedding vectors were queried for each claim embedding. The evidences retrieved were passed on to the stacked BERT model, which remained the same across all experiments.

  This implementation was finalized as the working model for our problem statement, after comparing the evaluation metrics of all models.

4.2 Comparison of Evaluation Metrics

The following results (see Table 2) were obtained on computing the Evidence Retrieval F-score (F) and Claim Classification Accuracy (A) during the comparative analysis on a fraction of the test dataset. (Note: These results were obtained from the scoring output log of Codalab during the Ongoing Evaluation stage.)

Model Description	Evidence Retrieval F-score	Claim Classification Accuracy	Harmonic Mean of F and A
Model 1 (BM25 Ranking and BERT)	0.0699	0.4605	0.1214
Model 2 (BM25 Ranking and 2 stacked BERTs)	0.0704	0.4868	0.1230
Model 3 (FAISS Embedding and BERT)	0.0942	0.4474	0.1556

Table 2: Comparative Analysis of the Evaluation Metrics on multiple model designs

The fourth column of Table 2 projects the harmonic mean of the evidence retrieval F-score and claim classification accuracy. This is the metric that determines how well each model performs, since it takes into consideration both segments of the fact-checking model - the evidence retrieval, and claim verification using the retrieved evidence. It is evident that **Model 3 produces the best output score of 0.1556** among all three models.

The performance of Model 3 when tested on the entire test dataset (during the Final Evaluation Round) has been recorded in Table 3.

Model Description	Evidence Retrieval F-score	Claim Classification Accuracy	Harmonic Mean of F and A
Model 3 (FAISS Embedding and BERT)	0.0900	0.4416	0.1495

Table 3: Performance of Model 3 in the final evaluation on complete test data

On the complete test dataset, **the final model produces an output score of 0.1495** which is evidently close to the evaluation observed during comparative analysis and hyper-tuning. This indicates that *the model is not over-tuned* based on the ongoing test set, and works reliably on unseen data.

5 Conclusions

In summary, investigated the capability of a pre-trained BERT model for evidence retrieval and claim verification. For the evidence retrieval step, we reviewed three key methods that were investigated in the duration of this project. Using BM25 as a ranking system with a BERT model for "evidence-for-claim binary classification" provides a higher accuracy in the subsequent claim verification. However, we concluded that this approach is not necessarily preferable if the recall values of retrieval is taken into account. Our third model that used a transformer with FAISS indexing proved to have a better evidence recall F-score of 0.0942. This improvement in recall holds more significance in the context of our problem statement, which led to the preference of this model in the finalized design.

Considering that the current version of this research had dataset and resource (GPU) constraints, exploring community-based, large BERT models can be considered in the future work. Incorporating ethical theories (Schramowski et al., 2019) to the existing BERT model, *especially in the final aggregation step*, could be an essential part of future work to avoid policy implications or dilemmas regarding climate change.

References

Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. 2012. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. 2019. Target-dependent sentiment classification with bert. *Ieee Access*, 7:154290–154299.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Ammar Ismael Kadhim. 2019. Term weighting for feature extraction on twitter: A comparison between bm25 and tf-idf. In *2019 international conference on advanced science and engineering (ICOASE)*, pages 124–128. IEEE.

David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv preprint arXiv:2111.01243*.

Patrick Schramowski, Cigdem Turan, Sophie Jentzsch, Constantin Rothkopf, and Kristian Kersting. 2019. Bert has a moral compass: Improvements of ethical and moral values of machines. *arXiv preprint arXiv:1912.05238*.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. Bert for evidence retrieval and claim verification. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 359–366. Springer.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Spyros Zoupanos, Stratis Kolovos, Athanasios Kanavos, Orestis Papadimitriou, and Manolis Maragoudakis. 2022. Efficient comparison of sentence embeddings. In *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, pages 1–6.