CLAIM SCAN Task A
Uncovering Truth in Social Media through Claim Detection

M Yagnesh
aib232069@iitd.ac.in
Yardi School of Artificial Intelligence
IIT Delhi

Abstract

For the purpose of this project, I will discuss a method that is based on deep learning that is used to recognize claims in the data that was collected from COVID-19 Twitter. There has been a significant rise in the amount of material that is being produced and the amount of information that is being transmitted as a direct result of the significant growth of social media platforms that are accessible online. On the other side, these platforms have nonetheless developed into safe havens for the transmission of propaganda, false news, and misinformation on the internet. The claims that people make have a tremendous impact on our perception of the world. These statements are frequently misunderstood by those who spread information that is not accurate. When it comes to finding a solution to this issue, I make use of a BERT-based model that was trained on a dataset comprising of tweets on COVID-19 that were either classified as claims or non-claims. This dataset was used to train the model. Extensive research has shown that BERT-based models are beneficial for claim identification in social media data. This research has provided insights into the reliability of material that has been published on Twitter regarding COVID-19. The research has showed that these models are useful for identifying claims in social media data.

1 Introduction

A number of extremely advantageous results have been brought about as a consequence of the exponential growth of online social media platforms, which has made it possible for the production of content and the transmission of information to explode. On these various channels of communication, the individuals who disseminate harmful rumors, propaganda, fake news, and misinformation have created a multitude of opportunities for themselves. Unfortuitously, those who disseminate false news frequently take advantage of claims in order to fool others. This is the case despite the fact that claims play a significant part in determining how we perceive the world.

Among the most prominent instances of this issue is the COVID-19 "Infodemic," which has resulted in the widespread dissemination of false medical claims, as demonstrated in the competition conducted by Megha Sundriyal and colleagues [1]. Social media sites have content moderators that are responsible for deleting bogus news from their networks. This task is done in order to address the problem that has been identified. On the other hand, being able to identify fake news in an efficient manner is made more difficult by the vast amount of data that is already available. As a result of this, the differentiation between assertions that are believable and those that are not, as well as the automatic identification of social media postings that contain assertions of this kind, has arisen as an important area of research in the field of natural language processing. The purpose of this project is to build a method that is based on deep learning for the purpose of claim detection in COVID-19 Twitter data. This methodology will be developed as part of this research. I would use the macro-F1 score as the primary evaluation criterion that I would incorporate into my evaluation process. I carried out experiments in order to validate the effectiveness of the strategy, and I did so by making use of a model that was based on the BERT framework. Throughout the course of this research, we offer some insights into the dependability of information that is associated with COVID-19 and is shared on Twitter. The usefulness of BERT-based models for claim

recognition in social media data is brought to light by my findings.

2 Motivation

This research is being carried out as a part of the competition for the ELL884-Deep Learning for Natural Language Processing course that is currently being offered. Because of the exponential growth of social media platforms that are available online, there has been a significant increase in the dissemination of false claims and misinformation. This is a rising trend that should be taken seriously. It is becoming more and more obvious that social media platforms are becoming more conscious of the need of recognizing and combating disinformation in order to guarantee the validity and reliability of the information that is shared on their platforms. However, this awareness is not yet widespread. A contribution to the ongoing efforts of social media platforms to identify and minimize the spread of misinformation is the goal of this initiative, which aims to give a contribution to such efforts. Building a solution for claim detection in Twitter data that is based on deep learning will be the means by which this objective will be realized.

3 Task Description

In the context of social media, the task at hand is to determine whether or not a claim is contained within the post that is being considered. Due to the fact that claims can take on a wide variety of structures and may be obscured within extensive portions of text, this effort can be fairly tough. Accordingly, it is of the utmost importance for the system to be able to identify linguistic signals and patterns that function as indicators of potential claims. Some examples of such cues include the use of assertive language, assertions that are made explicitly about a certain topic, and references to evidence or sources that provide support for the assertion.

4 Dataset Overview

The dataset is acquired from a publicly available competition link[2]. Training data, testing data, and validation data are contained within the dataset's three csv files, which are referred to as train, test, and dev files respectively. There are 6986 data items to be found in the train data, whereas there are 1497 items to be found in the validation data. The numbers and ratios that are presented below are examples of their presence in the dataset.

Dataset	Claim	Non-claim
Train	6106	880
Validation	1301	196

Table 1: Data Distribution of Claim Detection Dataset

Dataset	Claim (%)	Non-claim (%)
Train	87.5	12.5
Validation	86.9	13.1

Table 2: Normalized Data Distribution

Based on the analysis of the data, we may conclude that the data is skewed. In other words, the majority of the train and validation data are categorized as "Claim" (about 87.5It was mentioned in the Megha Sundriyal et al. [1] that the data has been annotated in the CLAIMSCAN 2023 in the past.

5 Implementation

In this section, implementation details of the claim detection model using BERT are discussed.

5.1 Preprocessing

The preprocessing steps are as follows:

- The dataset is read from the provided CSV files: `train.csv`, `dev.csv`, and `test.csv`. The training set consists of 6986 data items.
- The text data is preprocessed by removing URLs and non-ASCII characters from the tweets.
- The preprocessed dataset is then split into training, validation, and test sets.

The problem of massive volume of data that Twitter provides is that it includes a lot of information that is not necessary for us. The claim data, for instance, has emotes and emojis which are not a must in identifying the claims. Examples of this kind are hyperlinks and URLs and it will be more difficult for the model to train.

The data was preprocessed in order to ensure that it was free of any errors and thus ready for the analysis. URLs and non-ASCII characters were removed from the data. It was done in this way to make it data analysis-friendly. These kinds of preprocessing operations allow for the removal of noise and unhelpful information, which in turn leaves behind useful data that can be used for more precise modeling and training. The removal of URLs and non-ascii characters can be an efficient strategy for the normalization of textual data, thus, it becomes easier to manage and analyze. As well, it helps in data complexity reduction and increases the efficiency of natural language processing (NLP) models. Below is the raw data that was obtained as the consequence of the pre-processing.

5.2 Model

The model is trained using the pre-trained BERT model "bert-base-uncased". Here's a detailed breakdown of the implementation:

5.2.1 Tokenization and Encoding

for the purpose of tokenization and encoding the text contentthat was preprocessed, the BERT tokenizes each line, and sends the output to the language model. enizer is utilized. A pre-trained BERT model,represented by the token bert-base-cased. is the tool for make the tokenizer training.

- The data is tokenized through the BERT tokenizer which has the following tokenizer. Tokenization means splitting the given data into individual words, syllables, or letter sequences. For example, the sentence "Hello, how are you?" might be converted into a vector, which is ["1275", "0456", "3597", "3546", "6416", "8413"].
- The tokens after tokenization are then processed using a BERT tokenizer. Encoding involves tokening and changing the plain text data into numerical IDs that the BERT model can understand. Every one of a token will be assigned a certain ID instead of being identical. For instance, "Hello" would be assigned an abridged form of 1275 in its token form.
- The coded tokens will then be normalized to have the same length for more homogenous processing. In this instance, they filled up or cut off when they reached the maximum length of 128 tokens. If the number of tokens in a sequence is less than 128, the sequence is padded with special tokens to match its desired length. The words are shortened if it is longer, and if the maximum length is exceeded it is cut to fit the space.
- PyTorch `Dataset` and `DataLoader` classes are used to create data loaders. These classes enable efficient loading of the tokenized and encoded data during both the training and evaluation phases. The `Dataset` class provides an interface to access the dataset, while the `DataLoader` class enables loading of the data in batches, which is essential for training deep learning models.

5.2.2 Model Architecture

The claim detection model is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture. It consists of a pre-trained BERT model (`bert-base-uncased`) with additional layers added on top for fine-tuning.

- The BERT model is fine-tuned for the claim detection task by adding a fully connected feed-forward neural network (FFN) on top of the pre-trained BERT model.
- FFN elements, it includes dropout layer with the rate of 0. Thereafter, a combination of a fully connected layer and a

ID	Tweet	Claim
7149	New York AG Warns Televangelist #JimBakker To Stop Selling FAKE #CORONAVIRUS CURE #LisaLandau, chief of the AG's Health Care Bureau, sent a cease and desist letter to Bakker on Thursday threatening legal action #MOG #BlueWave2020 #MAGA #Christians https://t.co/y1iWhZWlsI	0
2334	@NVGOP @McDonaldNV @jeremybhughes The guy you're working so hard to re-elect thinks ingesting disinfectant (you will die) or getting bombarded with UV lights (you will die) are potential cures for #COVID19. https://t.co/TXKrJrZZp8	1
3897	RT @meanguitar: Breaking news: #China will admit #coronavirus coming from its P4 lab #BioWeapon - https://t.co/JywpSu35rN, -	1

Table 3: Sample of Raw Data

ID	Tweet	Claim
7149	New York AG Warns Televangelist #JimBakker To Stop Selling FAKE #CORONAVIRUS CURE #LisaLandau, chief of the AG's Health Care Bureau, sent a cease and desist letter to Bakker on Thursday threatening legal action	0
2334	@NVGOP @McDonaldNV @jeremybhughes The guy you're working so hard to re-elect thinks ingesting disinfectant (you will die) or getting bombarded with UV lights (you will die) are potential cures for #COVID19.	1
3897	RT @meanguitar: Breaking news: #China will admit #coronavirus coming from its P4 lab #BioWeapon -	1

Table 4: Sample of Processed Data

hyperbolic tangent (Tanh) activation function is applied and layer normalization is performed. This is however preceded by a sequential layer has a hidden layer size of 128 followed by a dropout layer with a dropout rate of 0. 3. Lastly, we utilize the 1D convolutional layer with one output node to make the binary classification prediction on whether the input text is a claim or not.

- During fine-tuning, only the top 4 layers of the BERT model are unfrozen for weight updates, while the rest of the BERT model parameters are frozen to prevent overfitting and retain the pre-trained knowledge captured by BERT.

The model architecture is defined in the following way:

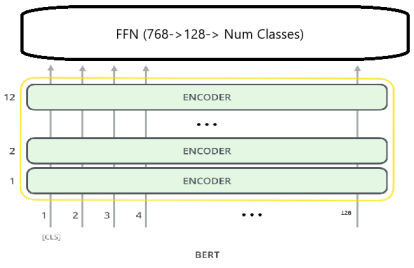

Figure 1: BERT Model Image

- The `BERTClass` class defines the claim detection model. It initializes the pre-trained BERT model (`bert-base-uncased`) and adds the additional FFN layers on top.

- During initialization, the parameters of the BERT model are frozen, except for the top 4 layers, which are unfrozen for weight updates during fine-tuning.

- The `forward` method of the model takes input IDs and attention masks as input and returns the model output, which represents the predicted claim label for each input text.

5.2.3 Hyperparameters

Hyperparameter	Value
Maximum Length	128
Batch Size	16
Learning Rate	1e-5 (Initially)
Class Weights	based on class distribution
Number of Epochs	20

Table 5: Hyperparameters

- **Maximum Length:** 128

  The maximum length is set to 128, considering that the dataset has a maximum word span of 108 after preprocessing, and the maximum characters allowed for a tweet on Twitter is 300. Therefore, 128 is chosen as the maximum length.

- **Batch Size:** 16

  The batch size is set to 16. During training, the data is divided into batches of size 16 for efficient processing.

- **Learning Rate:** 1e-5 (Initially)

  The initial learning rate is set to 1×10^{-5}. The AdamW optimizer is used for optimization. Additionally, a learning rate scheduler, ReduceLROnPlateau, is employed to adjust the learning rate during training. If the validation loss does not improve for 2 consecutive epochs, the learning rate is reduced by a factor of 0.5.

  In the first 8 epochs, the learning rate is reduced by 20% at the end of each epoch.

- **Class Weights:**

  The class weights are calculated to handle the class imbalance in the dataset. The ratio of the number of negative samples (N_{neg}) to positive samples (N_{pos}) is used to compute the class weights. These class weights ($\text{ClassWeight}_{\text{pos}}$ and $\text{ClassWeight}_{\text{neg}}$) are used in the loss function to give more weight to the minority class during training.

  The class weights are calculated using the following formula:

$$\text{ClassWeight}_{\text{pos}} = \frac{N_{\text{neg}}}{N_{\text{pos}}}$$

$$\text{ClassWeight}_{\text{neg}} = \frac{N_{\text{pos}}}{N_{\text{neg}}}$$

- **Number of Epochs:** 20

  The model is trained for 20 epochs. One epoch refers to one complete pass through the entire training dataset.

6 Training

The training process involves several key steps:

- **Loss Function:** Binary Cross-Entropy with Logits Loss (also written as BCEWithLogitsLoss) is the loss function that is utilized on this system. It is appropriate for multi-label classification problems, such as the one that is currently being discussed. In addition to this, the loss function features the incorporation of class weights in order to address the issue of class imbalance. When it comes to the positive class, the class weights are carried out.

- **Training Loop:**For the amount of epochs that have been specified, the training loop

will continue to iterate over the training data. The model is trained during the course of each epoch by making use of the training data. During the training process, the loop monitors both the loss and the accuracy of the model.

- **Forward Pass:**It is the model's responsibility to compute the forward pass for each batch of data. After that, the output of the model is compared with the actual targets in order to carry out the computation of damage.

- **Backward Pass and Optimization:** The gradients are obtained by the use of backpropagation once the loss has been calculated, and the optimizer then adjusts the model parameters in order to lower the loss as much as possible. To prevent gradients from exploding, the technique of gradient clipping is utilized.

- **Evaluation:** Each epoch is followed by an evaluation of the model that makes use of the validation data. Because of this, assessing the performance of the model and avoiding overfitting are both made easier. Recall, accuracy, precision, F1 score, and macro F1 score are some of the metrics that are used in the evaluation process.

- **Learning Rate Scheduling:** During the training process, a learning rate scheduler is established in order to make adjustments to the learning rate. A ReduceLROnPlateau scheduler is utilized in this scenario. This scheduler has the capability to decrease the learning rate by a factor of 0.5 in the event that the validation loss does not improve after a predetermined number of epochs.

When taken as a whole, this training procedure guarantees that the model will properly learn the patterns that are present in the data, while simultaneously preventing overfitting and addressing class imbalance.

7 Evaluation and Validation Results

The F1 score is the evaluation metric that is used for both of the activities. In order to complete Task A, the Scikit-learn Library in Python is utilized to generate Macro-F1 scores, accuracy scores, and other appropriate metrics. Accuracy is utilized so that we can determine the proportion of samples that have been correctly classified, and Macro F1 is utilized so that we can evaluate and contrast the precision and recall of the samples according to their respective classes.

In order to evaluate the performance of the model, the validation set is utilized. Following the completion of all epochs, the validation findings are as follows:

Metric	Value
Accuracy	0.8965
Macro F1 Score	0.7032
Precision	0.9081
Recall	0.98
F1 Score	0.9427

Table 6: Validation Metrics after the final epoch

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall** (also called Sensitivity):

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**:

$$\text{F1 Score} = 2 \times \frac{\text{Pr} \times \text{Rec}}{\text{Pr} + \text{Rec}} = \frac{2TP}{2TP + FP + FN}$$

- **Macro F1 Score**:

$$\text{Macro F1 Score} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times \text{Pr}_i \times \text{Rec}_i}{\text{Pr}_i + \text{Rec}_i}$$

8 Results

The model achieves a Macro-F1 score of [0.7032] and an accuracy of [0.8965] on the validation set.

8.1 Losses over epochs

The propogation of loss and evaluation metrics across the epochs are as follows:

Figure 2: Loss vs Epochs during Training

From the above loss vs Epochs graph, we can infer from the validation loss, we can see that the loss is increasing across epochs, but the training loss is decreasing which implies that the model is trying-to overfit the data. To avoid this, I have used regularization techniques like dropout and L2 regularization and early stopping. Even though I have not stopped the model early, I am storing the model in the epoch which is having optimal values of Macro-F1 score and accuracy for validation data. This leads to further evaluate the model and check if the model is recovering/ reducing loss in following steps.

8.2 Metrics over epochs

From the below graph we can see that the model is starting from a good scores and as the training proceeds, it is improving a bit better for some epochs and after some time, they are arriving at a plateau. This implies that the hyperparameters are better for the model as they give the model a good starting point and this help the model way better as the training data is small.

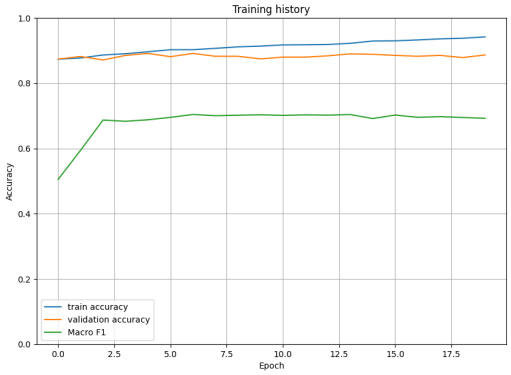

Figure 3: Evaluation Metrics during Training

8.3 Comparison with competition

The top 4 contenders accuracy scores in the competetion of kaggle are as follows

Index	Name	Accuracy
	benchmark.csv	1.00000
1	test_claim	0.90448
2	**Yagnesh**	**0.89971**
3	cehen44936	0.89780
4	Samarth Singla	0.89398

Table 7: Claim Scan Task A submission comparison

From the above submissions, I can say that for the test cases, my model performs nearly as better as the top performer. The top performer on the leaderboard performs by 0.5% more accurate than my model.

8.4 Comparison with Previous Competition

In the paper Megha Sundriyal et al [1],, the previous year models' Macro-F1 scores are displayed along with my model Macro F1 score.

When compared with the Claim Scan 2023 results, my model has outperformed the best performer NLytics. My model is based on BERT while teh NLytics model is based on RoBERTa which is of very high size. With these results, I am able to conclude that my model (BERT) is efficient, small and outperforms the previous year best model even though the previous best performer was using large and advanced model (RoBERTa)

Rank	Name	Macro-F1
	Yagnesh	**0.7032**
1	NLytics	0.7002
2	bhoomeendra	0.6900
3	amr8ta	0.6678
4	CODE	0.6526
5	michaelibrahim	0.6324
6	pakapro	0.4321

Table 8: Comparison with Claim Scan 2023 Rankings based on Macro-F1

9 Future Work

While the current model achieves good performance, there are several avenues for future work that could further advance research in claim detection:

- **Ensemble Methods:** Ensemble methods, such as stacking, blending, or boosting, could be explored to improve model performance. By combining predictions from multiple models, ensemble methods often result in better generalization and robustness.
- **Applying the Model to Other Domains:** The current model is trained on a dataset focused on a specific domain. Extending the model to other domains, such as social media, news articles, or scientific literature, could enhance its applicability and generalization capability. This extension would involve retraining the model on domain-specific datasets and fine-tuning its parameters to adapt to the characteristics of the new domain. For instance, the model could be adapted to identify misinformation and fake news in social media or to detect claims in scientific literature. Additionally, exploring techniques for domain adaptation and transfer learning could facilitate the adaptation of the model to new domains.

By exploring these avenues for future work, it is possible to advance research in claim detection and develop models that are more accurate, robust, and applicable to various domains.

10 Limitations

While the developed claim detection model demonstrates promising performance, several limitations should be considered:

- **Data Imbalance:** The dataset used for training the model is imbalanced, with a majority of samples belonging to the "Claim" class. This class imbalance may lead to biased model predictions and affect the model's ability to generalize to new data.
- **Limited Domain:** The model is trained and evaluated on a specific domain, which may limit its generalizability to other domains. While the model performs well within the domain it was trained on, its performance may degrade when applied to different domains, such as social media or scientific literature.
- **Feature Engineering:** The current model relies solely on text data for claim detection. While the BERT architecture captures rich semantic information from text, incorporating additional features such as user metadata, temporal information, or external knowledge sources could further enhance the model's performance.
- **Model Interpretability:** Deep learning models, including BERT-based models, are often considered black-box models, making it challenging to interpret their predictions. Enhancing model interpretability could help users better understand the model's decision-making process and build trust in its predictions.
- **Scalability:** The current model may face scalability challenges when applied to large-scale datasets or deployed in real-time systems. Optimizing the model's architecture and training procedures for efficiency and scalability could address these challenges.

Considering these limitations is essential for understanding the model's performance and

identifying areas for improvement in future research.

References

1. Megha Sundriyal, Md Shad Akhtar and Tanmoy Chakraborty, Overview of the CLAIMSCAN-2023: Uncovering Truth in Social Media through Claim Detection and Identification of Claim Spans. https://arxiv.org/pdf/2310.19267

2. ClaimScan Task A. https://www.kaggle.com/competitions/claimscan-task-a

3. M. Sundriyal, P. Singh, M. S. Akhtar, S. Sengupta, T. Chakraborty, Desyr: definition and syntactic representation based claim detection on the web, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1764–1773.

4. T. Chakrabarty, C. Hidey, K. McKeown, IMHO fine-tuning improves claim detection, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp.558–563. https://aclanthology.org/N19-1054.

5. T. Chakrabarty, C. Hidey, K. McKeown, IMHO fine-tuning improves claim detection, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 241–251. https://www.aclweb.org/anthology/N19-1023/.

6. S. B. Naeem, R. Bhatti, The covid-19 'infodemic': a new front for information professionals, Health information and libraries journal 37 (2020) 233–239. https://europepmc.org/articles/PMC7323420. doi:10.1111/hir.12311.

7. T. Chakrabarty, C. Hidey, K. McKeown, IMHO fine-tuning improves claim detection, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 241–251. https://www.aclweb.org/anthology/N19-1023/.

8. I. Jaradat, P. Gencheva, A. Barrón-Cedeño, L. Màrquez, P. Nakov, ClaimRank: Detectingc check-worthy claims in Arabic and English, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 26–30. :https://aclanthology.org/N18-5006.

9. D. Wright, I. Augenstein, Claim check-worthiness detection as positive unlabelled learning, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 476–488 :https://aclanthology.org/2020.findings-emnlp.43.

10. TS. Zhi, Y. Sun, J. Liu, C. Zhang, J. Han, Claimverif: A real-time claim verification system using the web and fact databases, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 2555–2558. :https://doi.org/10.1145/3132847.3133182.

11. M. Sundriyal, G. Malhotra, M. S. Akhtar, S. Sengupta, A. Fano, T. Chakraborty, Document retrieval and claim verification to mitigate covid-19 misinformation, in: Proceedings of the Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situations, 2022, pp. 66–74.

12. S. Rosenthal, K. McKeown, Detecting opinionated claims in online discussions, in: Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing, ICSC '12, IEEE Computer Society, USA, 2012, p. 30–37. :https://doi.org/10.1109/ICSC.2012.59..