Disaster Tweets Classification by BERT and Its Variants

Xuefang Hu, Changchang Liu, Anupam Kumar

CS 6120 Natural Language Processing, Fall 2021

Abstract

Bidirectional Encoder Representations from Transformers (BERT) [1] is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. Recently, BERT has become a ubiquitous baseline in NLP experiments. And there are also many variants of BERT, for example, Robustly Optimized BERT Pretraining Approach (RoBERTa) [2] and A Lite BERT (ALBERT) [3]. Given a dataset of Disaster Tweets, we do the sentiment analysis by BERT and these two variants to predict if a tweet is announcing a disaster or not and compare the performance of those different models.

1 Introduction

Many agencies, for example, news agencies and disaster relief organizations are now interested in monitoring Twitter as it becomes more and more important in people's communications everyday, and even in emergency times. However, the words from smartphones can be very ambiguous. Things such as metaphor, equivocation and implicitness exist in human languages, and they are much unclear to machines compared to humans. We are going to build a model that can predict which Tweets are about real disasters and which ones are not.

We use the dataset from Kaggle [4], where each row contains the text of a tweet, a keyword from the tweet and the location that the tweet is sent from. However, the keyword and the location might be null in some rows. And we want to predict whether a given tweet is about a real disaster or not. The main results we have are the labels showing whether a tweet is a real disaster (label = 1) or not (label = 0).

The libraries we used are Scikit-learn and Tensorflow, as well as matplotlib for some visualization. We use the BERT model as well as its variants ALBERT and RoBERTa to do the prediction and compare their training time as well as performance.

In section 2, we briefly introduce BERT, RoBERTa and ALBERT and their differences. In section 3, we present our experiments on the data using those three models. And we bring a summary of our experiments in section 4.

2 Methodology

2.1 Data Analysis and Preprocessing

Checked for duplicate data entry based on ('text', 'target') and ('keyword', 'text') columns, and removed ∼200 duplicates out of ∼8k entries. We looked up common abbreviations and replaced them with their meaning. We used spacy for lemmatization and regexp for removing URLs, HTML, emojis, etc. We used TensorFlow hub universal sentence encoder to extract a keyword from a sentence to check if it acts as a representer of disaster for tweets. For example, we get 'crash' for a tweet 'just happened a terrible car crash'. We used TensorFlow datasets to load the data in a batch of 32.

Figure 1: Non-disaster word cloud

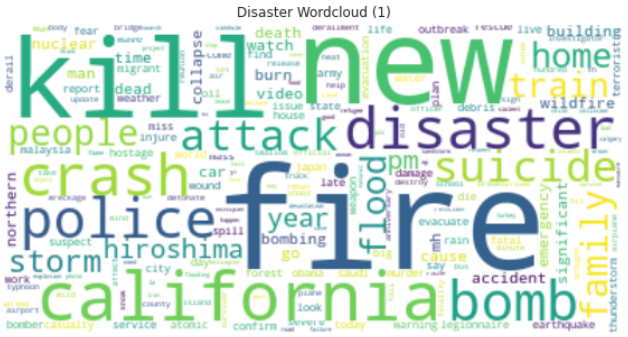

Figure 2: Disaster word cloud

2.2 Building the Classifier Model

The general idea for classifying the text is to find an array representation of the text and use a classification algorithm like Random Forest, SVM, Neural Networks, etc. One can also use Naive Bayes or LSTM which club the process of representation of the text and classification. Though current state-of-the-art methods are based on BERT, and these are based on the Transformer concept. Ideally, we should have used a more traditional framework for baseline and used BERT variants to check the accuracy gain over the baseline. We decided to use a reference https://www.kaggle.com/zinebkhanjari/disaster-tweets-multiple-vectorizers-and-models which used various traditional frameworks and got 0.76 F1-score using LogisticRegression using TfIdf vectorization and Word2Vec transformation using gensim library. For this task, we used BERT, ELECTRA, RoBERTa, and ALBERT methods.

2.3 BERT and its variants

2.3.1 Why BERT and how it works

BERT was built upon recent work and ideas in pre-training contextual representations including Generative Pre-Training, ELMo, the OpenAI Transformer, ULMFit and the Transformer. While these models are all unidirectional (LSTM) or shallowly bidirectional (ELMo), BERT is fully bidirectional reading both directions at the same time, with interactions from both directions. BERT gives incredible accuracy and performance on smaller datasets which solves a huge problem in natural language processing, as datasets in NLP tasks are very sparse and often customized. BERT is a language representation model, and only provides the encoder part. It takes input text data and creates fixed shape segments or sentence embeddings. These embeddings can be processed further for different types of tasks.

BERT is not pre-trained using traditional left-to-right or right-to-left language models. Instead, it is pre-trained using two unsupervised tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). For the first task, 15% of all WordPiece tokens in each sequence at random is masked and the task is predicting those masked words rather than reconstructing the entire input. In order to train a model that understands sentence relationships, The second task pre-train the BERT model to determine whether a sentence is the next sentence or not.

2.3.2 BERT

We used BERT model architecture with 12 hidden layers, 768 hidden units, and 12 attention heads. This architecture has 110 million parameters.

2.3.3 ELECTRA

BERT variants require often require large compute for producing good results when transferred to downstream NLP tasks. ELECTRA provides a more efficient alternative using replaced token detection as

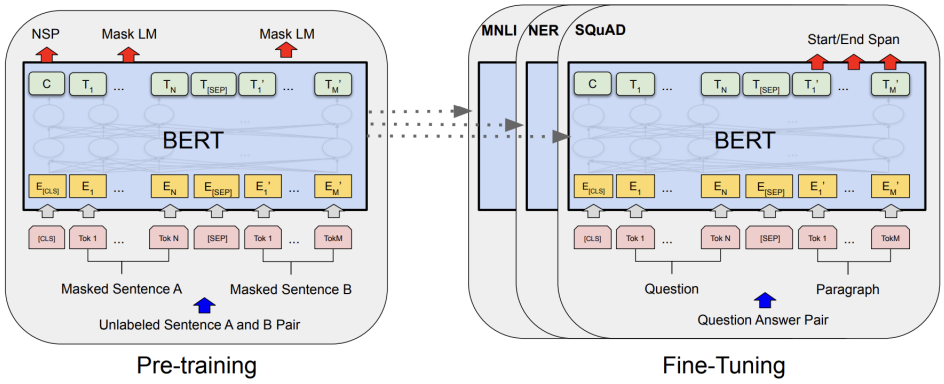

Figure 3: Overall pre-training and fine-tuning procedures for BERT

pre-training task instead of replacing tokens with mask. As can be seen in the results section, ELECTRA small takes less time to train due to less number of parameters to train but at a cost of overall perfomance.

2.3.4 ALBERT

ALBERT stands for 'A Lite BERT' and the work was published in February 2020. Increasing model size and increased parameters often results in better performance on downstream tasks in natural language representations. But there can often be memory and compute limitations, especially on mobile devices.

ALBERT is designed to address these issues. It incorporate two parameter reduction techniques: factorized embedding parameterization and cross-layer parameter sharing. The first one makes it easier to grow the hidden without significantly increasing the parameter size oand the other one prevents the parameter from growing with the depth of the network. Both of them significantly reduce the number of parameters for BERT without hurting performance seriously. In ALBERT, we can see that larger structure does not lead to more parameters. The Albert-base model we used has 31 million parameters compared to 110 million parameters in BERT.

2.3.5 RoBERTa

RoBERTa stands for 'Robustly Optimized BERT Pretraining Approach' and this language model was published by Facebook AI researchers in July 2019. After building a model, a lot of effort is invested in hyperparameter tuning to get the optimal performance from the network. This process can be computationally very expensive. The authors found that BERT was quite undertrained. They proposed modifications to the BERT pretraining procedure improving the overall robustness of the model and performance.

For example, BERT is trained on 16GB of uncompressed text and RoBERTa uses 10 times larger (160GB) uncompressed text. During training, RoBERTa model uses dynamic masking instead of the static masking in BERT. Also, it enlarges the batch size of BERT from 256 to 8k and uses byte-level encoding instead of the character-level encoding of BERT. It is also evident in the result section as Roberta-base model outperform other BERT variants.

3 Experiments and Results

We used TensorFlow hub for implementation of BERT, Electra-small, and ALBERT-base while transformers module for the implementation of Roberta-base. The text preprocessing before implementation of

Roberta-base was less heavy than for implementations of other BERT variants. We have submitted these two codes along with project report submission.
We used around 7k tweets to classify them into disastrous and non-disastrous tweets. We used 80% data for training and 20% for validation purposes. We trained each BERT variant model for 3 epochs. We took 5e-5 as learning rate for BERT, ALBERT-base, 1e-5 for Electra-small and 2e-5 for RoBERTa-base model. We used KFold with 6 splits for RoBERTa-base model but not for other architectures. We used the GPU on Kaggle to run all of the models and compare the training time. We can see a comparison of the training time, validation set accuracy, F1-score and the number of parameters is shown in the below chart:

BERT variants	training-time (sec.)	val-accuracy	F1-score	# parameters
BERT	1156.578	0.812	0.768	110M
Electra-small	422.668	0.766	0.725	14M
ALBERT-base	1125.614	0.812	0.761	31M
RoBERTa-base	426.732	0.837	**0.812**	123M
Distilbert		0.841	0.806	66M

The code base and project report can be found at
https://github.com/anupam312nwd/CS6120_Disaster_Tweet_project.

4 Conclusion

By the experiments, we can see that, with more parameters and dynamic masking, RoBERTa-base outperformed the other models in training time, val-accuracy as well as F1-score. Though ELECTRA-small has the lowest val-accuracy and F1-score, it has nearly
ALBERT-base and BERT-base perform similarly in all of those three metrics. As in [3] stated, ALBERT will perform better when the structure becomes larger. So ALBERT-base does not have significantly good performance since it has much fewer parameters and do not have large enough structure.

5 Further Topics

We did not perform much hyperparameter tuning on BERT variants model, because it took us some time to have each model working and partly due to compute constraints as we were using Google Colab and Kaggle GPU resources. We did tune learning rate for Electra-small as the learning rate 5e-5 used for other variants resulted in very bad performance on Electra-small. For future work, we can do hyperparameter tuning. Some easy to tune hyperparameters are learning rate, batch size, warm up steps for AdamW optimizer. We would also like to use KFold split on dataset for all architecture such that the average validation accuracy and F1-score is a better representative of result.

Given more time, we would like to do a more elaborate EDA of the text data, and implement several baseline models like Random Forest, SVM and LightGBM using gensim and others Word2Vec transformation.

	id	keyword	location	text	target
3577	5110	famine	Charter Member of the VRWC	Russian 'food crematoria' provoke outrage amid crisis famine memories - Yahoo News http://t.co/6siiRlnV6z	1
2700	3874	detonation	NaN	Ignition Knock (Detonation) Sensor-Senso BECK/ARNLEY 158-1017 http://t.co/ryoByQJFCE http://t.co/LW9O2kDk18	0
5969	8522	screaming	Jariana Town	@justinbieber @ArianaGrande Can you hear me screaming !!!!!	0
1916	2757	curfew	NaN	And the fact that i have a curfew	0
5430	7749	police	NaN	These fucking police can't touch me these fuck niggas ain't fucking w me	0
4792	6817	loud%20bang	Bedford, England	The chick I work with chews chewing gum so loud ?? feel to bang her	0
243	346	annihilation	California, USA	@rvfriedmann Hell is just a fraction of his belief of total annihilation destruction of USA @LodiSilverado @ritzy_jewels	1
3678	5233	fatality	Grand Rapids MI	@FaTality_US need a team? We need one.	0
5870	8387	ruin	youtube.com/channel/UCHWTLC9B4ZjUGh7yDlb55Iw	@nbc I wanna see you reboot The Fresh Prince of Bel-Air bring back the original cast and everything & do nothing that will ruin the show.	0
5666	8085	rescue	Karachi	Flood-zone : General Raheel Sharif visits Chitral: He also lauded the efforts of FWO army troops and army aviation in rescue opera...	1

Figure 4: Dataset Samples

6 Statement of Contributions

Authors: Xuefang Hu, Changchang Liu, Anupam Kumar

Each member contributed evenly in this project.
Xuefang was responsible for cleaning the data and evaluating the models on the test set, Anupam implemented and trained the BERT and ALBERT model and did the data visualization as well and Changchang mainly implemented and trained the RoBERTa model. All of the team members wrote the report and slides together.

References

[1] Jacob Devlin; Ming-Wei Chang; Kenton Lee and Kristina Toutanova (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805 [cs.CL].

[2] Yinhan Liu; Myle Ott and Naman Goyal; Jingfei Du; Mandar Joshi; Danqi Chen; Omer Levy; Mike Lewis; Luke Zettlemoyer and Veselin Stoyanov (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv:1907.11692 [cs.CL].

[3] Zhenzhong Lan; Mingda Chen; Sebastian Goodman; Kevin Gimpel; Piyush Sharma and Radu Soricut (2020). *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. arXiv:1909.11942 [cs.CL].

[4] https://www.kaggle.com/c/nlp-getting-started/data

7 Appendix

7.1 Dataset

The training dataset we used has 5 columns and 7613 rows, in which around 43.55% are labeled as non-disaster (0) and around 57.45% of the data is labeled as disaster tweets (1). And the test dataset has 4 columns (without label) and 3263 rows. A sampling of 10 of the training dataset is as follows.

7.2 Codes

The detailed codes of this project can be found in GitHub:
https://github.com/anupam312nwd/CS6120_Disaster_Tweet_project