BUS 41204

Machine Learning

by Professor Mladen Kolar

Disaster Tweets Classification with BERT

Simon H. Park

Winter 2020

The University of Chicago
Booth School of Business

Contents

1 Introduction

Along with the rise of social media over the last decade, Tweeter has changed the information collection and dissemination in times of emergency. Smart devices are connecting the everyone in real time and thus have become one of the most important communication channel. For this reason, various agencies are interested in systematically monitoring Tweeter. Indeed, there is a huge potential in leveraging Tweeter to accomplish many tasks in disaster management, including sentiment analysis.

Natural language processing (NLP) is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The goal of NLP is for machines to read and understand the natural (human) languages in a manner that is valuable. One of the most popular technique in NLP is sentiment analysis, a famous use case being the restaurant review classification as either positive or negative. It involves applying computer algorithms to *learn* the corpus and interpret the meaning. A restaurant review can go through a data processing such as stemming and lemmatization (what they are will become clear in the later section) and broken down into semantics, or "tokens" that provide context; carrying the information not only of the lexical meaning, but of the interactive relationships with other tokens in the corpus. Ideally, we want the machine to look beyond the words on the page to see the meaning like any good reader. Then, the machine will be able tell us whether the review was good or bad.

This paper analyzes the dataset containing tweets on disasters and make predictions on whether tweets are about real disasters or not using a neural network-based technique for NLP called BERT and is organized as follows. Section 2 introduces BERT. Section 3 describes the dataset. Section 4 introduces various machine learning algorithms to compare. Section 5 combines the findings and compares their performances. Section 6 closes with the final conclusion along with limitations and implications.

2 A glimpse inside BERT

2.1 What is BERT?

In 2018, Google introduced open-sourced a neural network-based technique for natural language processing pre-training called Bidirectional Encoder Representations from Transformers (BERT). The introduction of BERT is described as marking the beginning of a new era in NLP as the technology enables anyone to utilize its pre-trained models and build their own model through a process called fine-tuning, which would otherwise, in the absence of it, require massive amount of time, energy, knowledge, and resources to build from scratch.

BERT builds on top of a number of clever ideas including, ELMo[1], ULMFiT[2], the OpenAI transformer[3], the Transformer[4], and their novel approach on capturing the fully bidirectional contextual word-embeddings.

[1]Matthew Peters at el. [link to paper]
[2]Jeremy Howard and Sebastian Ruder. [link to paper]
[3]Radford at el. [link to paper]
[4]Vaswani et al. [link to paper]

2.2 A brief history of NLP and BERT

2.2.1 Masked Language Model

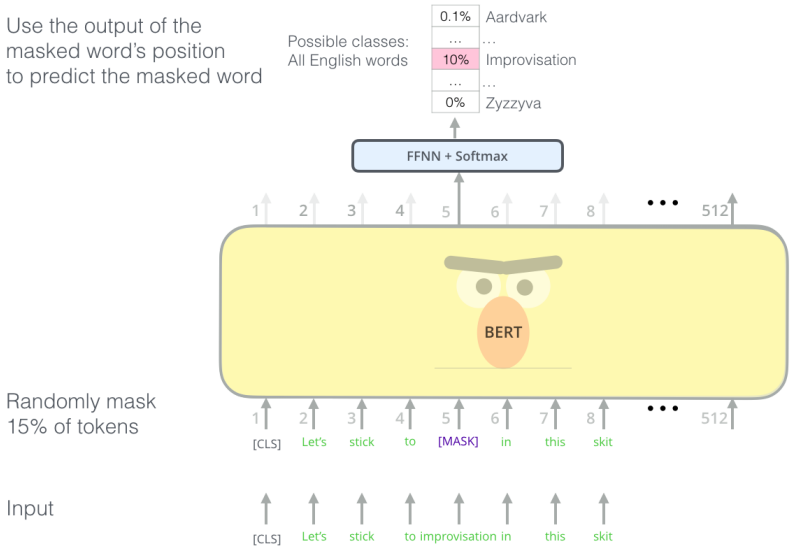

Figure 1: BERT's clever language modeling

For words to be processed by ML algorithms, they need to be converted to numeric representations. Word2Vec showed that a vector can properly represent words in a way that captures both semantic (or contextual) and syntactic (or grammatical) relationships. Lists of words and their embeddings generated by Word2Vec or Glove that were pre-trained on vast amounts of text became available. But under these representations, each word is represented by a unique vector regardless of the context. Accordingly, contextualized word-embeddings were introduced by ELMo. A same word in different context is now represented by different vectors. But, the context was limited in a sense that it looked at next words in forward and backward sequence in isolation. BERT improved this by originating the masked language model that is conditioned on

both left and right context, providing a fully bidirectional word-embeddings.

2.2.2 Fine-tunable Language Model

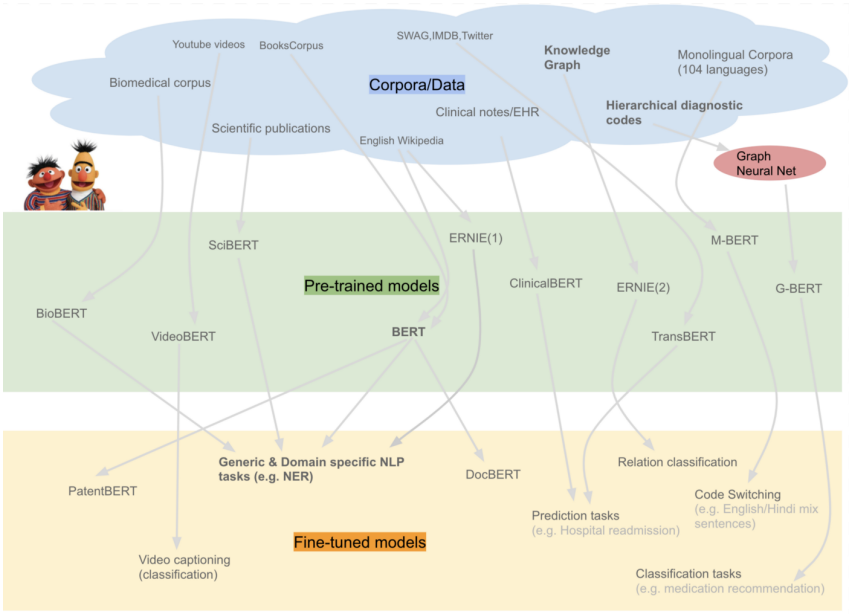

Figure 2: Pre-trained models of BERT

Fine-tuning is a method to effectively utilize a pre-trained model. ULM-FiT introduced a language model and a process to (discriminativly) fine-tune the language model for various tasks. The Transformer adopted a Encoder-Decoder structure that was perfect for machine translation. Then, OpenAI transformer, building upon the Decoder part, introduced fine-tunable pre-trained models for downstream tasks, such as a classficiation problem. But transformer-based models were a forward language model. BERT incorporated the masked language model into transformer-based models and also improves fine-tuning mechanisms to handle, for example, two-sentence tasks and providing task-specific models.

2.3 The encoder: feature extraction

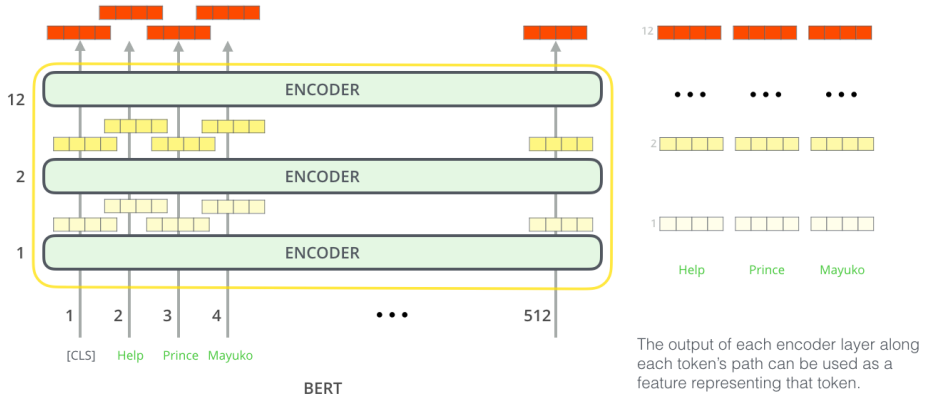

Figure 3: Generating Contextualized Embeddings

There is more to BERT models than fine-tuning approach. Pre-trained models can be used create contextualized word embeddings, 768-dimensional vectors. Then, these vectors can be fed into existing ML algorithms such as Random Forest. The original paper shows results of this process not far behind fine-tuning BERT using this process. In this paper, I compare the performances of DistilBERT embeddings with various ML algorithms versus the performance of fine-tuning BERT BASE.

3 Disaster tweets dataset

The dataset is from Kaggle's competition "Real or Not? NLP with Disaster Tweets", which was created by the company figure-eight and originally shared on their 'Data For Everyone' website.

3.1 The data

The dataset contains a total of 10,875 sample tweets on disasters, and is partitioned into 70% training and 30% test set, 7,613 and 3,263 observations respectively. Each sample in the data has the following information:

- A unique ID
- A binary label, 1 indicating real disaster tweets and 0 otherwise.
- The text of the tweet
- A keyword from that tweet (may be missing)
- The location the tweet was sent from (may be missing)

Table 1 illustrates 9 and 6 samples of the training and test set, respectively.

Table 1: Disaster Tweets Data

id	label	text	keyword	location
1	1	Our Deeds are the...		
4	1	Forest fire near ...		
5	1	All residents ask...		
		...	...	...
273	0	@Kiwi_Karyn Check...	ambulance	Loveland Colorado
274	1	when you don't kn...	ambulance	|| c h i c a g o ||
276	1	#reuters Twelve f...	ambulance	
		...	...	...
10871	1	M1.94 [01:04 UTC]...		
10872	1	Police investigat...		
10873	1	The Latest: More ...		

id	label	text	keyword	location
0	-	Just happened a t...		
2	-	Heard about #eart...		
3	-	there is a forest...		
		...	...	...
10868	-	Green Line derail...		
10874	-	MEG issues Hazard...		
10875	-	#CityofCalgary ha...		

3.2 Pre-processing

The data must be transformed into vectorized dataframe that can be fed into various ML algorithms. Figure 4 visualizes the use of DistilBERT, a smaller version of BERT that is specialized for feature extraction.

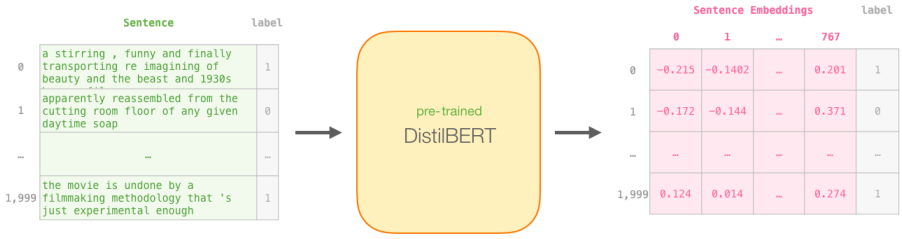

Figure 4: Sentence-Embeddings by DistilBERT

Despite the fact that there is no easy way to explain what goes on in the 'blackbox' of BERT or any other neural network-based technology, I try to demonstrate the process here. First, each tweet goes through stemming and lemmatization and turns into a list tokens of length 128[5], which is then mapped into a predefined numeric representation from nearly 30,000 vocabularies.

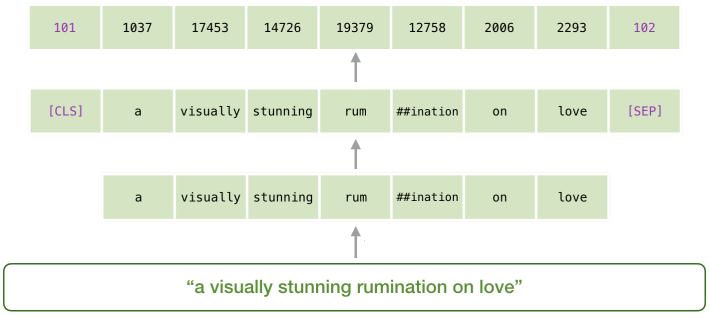

Figure 5: Illustration of BERT's Tokenizer

[5]The length can be set to any number smaller than 128 for BERT base and 256 for BERT large. 128 was chosen based on the maximum length of tweets in our dataset.

BERT takes the output of the tokenizer as input which then passes through 12 encoders, or layers. Each layer applies multi-head attention, and passes its results through a feed-forward network, and feed it to the next encoder. The final output from the last hidden layer returns 768-dimensional vectors,

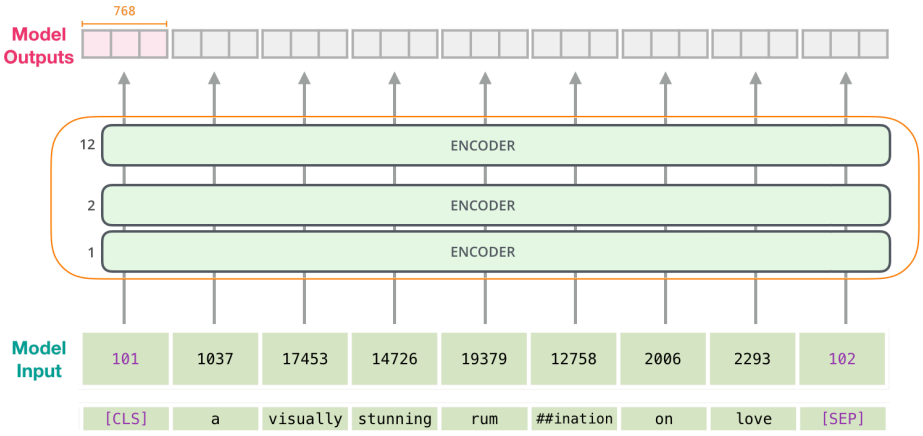

Figure 6: Last Hidden Layer from 12 encoders

the embeddings, for each tokens, a tensor (matrix) of size 768×128=98,304 elements. Notice the [CLS] token that is supplied at the beginning of the list in figure 5. CLS here stands for Classification, which we can translate as the embedding of the tweet as a whole, the vector of our interest.

3.3 Embeddings of DistilBERT

Table 2 illustrates 9 samples of training data that is retreived from DistilBERT's output, specifically the embeddings for [CLS] token. Note that I appended the target labels and original texts for illustration purposes.

Table 2: DistilBERT Tweets Embeddings

id	label	text	V0	V1	V2	···	V765	V766	V767
1	1	Our Dee...	-0.159	-0.060	0.092	···	0.000	0.331	0.198
4	1	Forest ...	-0.222	0.106	0.177	···	-0.048	0.137	0.247
5	1	All res...	-0.306	-0.046	0.106	···	0.003	0.350	0.167
		...							
273	0	@Kiwi_K...	-0.194	-0.166	0.086	···	0.062	0.474	0.104
274	1	when yo...	-0.255	-0.123	0.065	···	-0.022	0.364	0.286
276	1	#reuter...	-0.267	-0.271	0.202	···	0.054	0.258	0.092
		...							
10871	1	M1.94 [...	-0.253	-0.067	0.271	···	0.035	0.284	0.104
10872	1	Police ...	-0.291	-0.023	0.126	···	0.086	0.342	0.101
10873	1	The Lat...	-0.196	-0.063	0.100	···	0.061	0.376	0.131

4 Models and evaluation metric

I compare the performance of two BERT technologies. With DistilBERT embeddings, three ML algorithms were used for binary classification after fine-tuning their hyperparameters: Logistic Regression, Classification Tree, and Random Forest. Then, I present BERT-base's fine-tuned binary classifier.

For evaluation, I report the confusion matrix, formatted as illustrated below, to describe the performance of each classifier.

`Actual`\`Prediction`	Positive	Negative	
Positive	True Positive	False Negative	Sensitivity
Negative	False Positive	True Negative	Specificity
	Precision	Neg-preci	Accuracy
		Total Observation	F1 score

5 Model performances

5.1 Logistic Regression

Actual\Prediction	Positive	Negative	
Positive	766	636	Sensitivity=0.55
Negative	439	1,422	Specificity=0.76
	Precision=0.64	Neg-preci=0.69	Accuracy=0.67
		Total n=3,263	F1 score=0.73

5.2 Classification Tree

Actual\Prediction	Positive	Negative	
Positive	751	651	Sensitivity=0.54
Negative	704	1,157	Specificity=0.62
	Precision=0.52	Neg-preci=0.64	Accuracy=0.58
		Total n=3,263	F1 score=0.63

5.3 Random Forest

Actual\Prediction	Positive	Negative	
Positive	534	868	Sensitivity=0.38
Negative	382	1,479	Specificity=0.79
	Precision=0.58	Neg-preci=0.63	Accuracy=0.62
		Total n=3,263	F1 score=0.70

5.4 Binary Classifier of BERT-base[6] (fine-tuned)

Actual\Prediction	Positive	Negative	
Positive	959	443	Sensitivity=0.68
Negative	196	1,665	Specificity=0.89
	Precision=0.83	Neg-preci=0.79	Accuracy=0.80
		Total n=3,263	F1 score=0.75

[6]The default classifier uses logit and GELU activation. See here for GELU activation.

6 Conclusion

The results indicate that fine-tuned BERT-base binary classifier outperforms any other models that were applied to DistilBERT embeddings. The former reports overall accuracy of 0.80 while the latter reports 0.67, 0.58, 0.62 for the logit, CART, and RF models, respectively.

Throughout the analysis, I utilized pre-trained BERT models, developed and open sourced by HuggingFace team. Indeed, it would cost tremendous time and effort if I were to train a language model on a comparable to BERT as it was trained on Wikipedia and Book Corpus, a dataset containing over 10,000 books of different genres. Undoubtedly, the two BERT models enabled me to build machine learning models that make binary classifications with remarkable performances given the (relatively) minimal input.

Specifically, I was able to demonstrate the strength of fine-tunability of BERT, which yielded a 13 percentage point increase in overall classifications accuracy compared to same logit model with DistilBERT[7]. The results were as anticipated as the fine-tuning approach is designed to enhance the performance on a specific task. For further improvements, I expect that fine-tuning BERT-large will provide far better outcomes when properly executed.

Potential use cases for BERT are immeasurable. For instance, this competition could grow into a disaster alert system in collaboration with Twitter, or similar semantic analysis of product reviews can be done for marketing purposes. Assuredly, this powerful technology is worth exploring.

[7]Opportunely, I could not train DistilBERT due to the memory capacity problems.

Acknowledgement: the figures are recreations of the images from Jay Alammar's blog, which guided this project by providing great illustrations of and insights on BERT.