# SSL Assignment - Real or Not? NLP with Disaster Tweets


## Robert-Marius Draghici, SSA[1]

[1]*Email: marius.robert.draghici@gmail.com*


This document presents the approach chosen for the Kaggle challenge "Real or Not? NLP with Disaster Tweets" [1]. The purpose of this challenge is to train a machine learning model that can identify if a tweet refers to a real disaster or not.

The solution was implemented in Google Colaboratory, which is a free cloud service that supports a free GPU (Tesla K80 GPU) useful for developing deep learning applications.

For the development of my solution, I have used the following libraries:

- pandas [2], version 1.0.3 for reading the dataset and storing it into a DataFrame
- train_test_split method from sklearn.model_selection (version 0.22.2.post1) [3] to split the training data in a train set and a validation set
- spacy [4], version 2.2.4, a new NLP library that's designed to be fast, streamlined, and production-ready, for text preprocessing and cleaning
- python string module for punctuation
- python re module for regular expressions
- ktrain library [5] for training models and inference. ktrain is a lightweight wrapper for tf.keras in TensorFlow 2 and from version 0.8 it offers a simplified interface to Hugging Face transformers [6] for text classification.
- simpletransformers library [7] which is based on the Transformers library by HuggingFace and allows quick training and evaluation of Transformer models.

While implementing the solution, for the first milestone I used the following articles and notebooks for guidance: a Medium article for ktrain [8], the ktrain tutorials from the Github repository [5] and these Kaggle kernels [9], [10]. For the second milestone, I used the following articles and notebooks for guidance: an article about the simple transformers library [11], the simple transformers tutorials from the Github repository [7], these Kaggle kernels [12] and [13] and the official page for the Hugging face transformers library [6].

I tried two approaches namely fine-tuning pretrained transformer models with and without text cleaning and preprocessing. The cleaning and preprocessing steps applied for the first milestone are lemmatization and the removal of punctuation, stopwords and urls. For the second milestone, I removed additional spaces, additional punctuation characters that were not included in the python library string.punctuation, the numbers and I have expanded syntactic constructions such as "I'm" into "I am".

For the first milestone, I trained several transformer based architectures using the ktrain library, such as BERT [14], DistilBERT [15], Albert [16] and XLNet [17]. For the second milestone, I trained other transformer based models using the ktrain and simple transformers libraries. The models are Electra [18], RoBERTa, DistilRoberta [19] and "DeepPavlov/bert-base-cased-conversational" [20]. In addition, I have also tried to train simpler models such as logistic regression, svm and GRU neural networks, but the results obtained were not satisfactory (under 0.7 f1 score).

The results for the first milestone are: For the unclean data, I obtained the following results:

- BERT 0.82310
- DistilBert 0.81390
- Albert 0.77811
- XLNet 0.80061

For the cleaned data, I obtained the following results:

- BERT 0.82208
- DistilBert 0.79856

The results for the second milestone are: For the unclean data, I obtained the following results:

- Deep Pavlov 0.80163
- electra-small-generator 0.78016
- electra-small-discriminator 0.76073
- distilroberta-base 0.81901
- roberta-base 0.81901
- albert v2 base 0.80572

For the clean data, I obtained the following results:

- Deep Pavlov 0.82106 with ktrain and 0.80572 with simple transformers
- electra-small-generator 0.82413
- electra-small-discriminator 0.81901
- distilroberta-base 0.80777
- roberta-base 0.80981

Transformer models seem to underperform when text cleaning and preprocessing are used. The removal of stop words may have a negative impact on the performance, as it can change the meaning of a sentence completely (for example, by removing "not" a negative sentiment can become a positive one, which is not desired). However, this does not seem to apply for ELECTRA based models, which have poor performance on unclean data and much better performance on clean data.

After analysing some of the wrong predictions, I discovered that some of them even though contain references to past disasters, the context does not imply that the conversation is about a disaster. Example: "over half of poll respondents worry nuclear disaster fading from public consciousness http fu ##kushima". Other predictions are wrong, because the tweet contains words that could represent a disaster, but in reality they refer to domain specific items. For example, in the tweet "leicestermerc icymi ashes 2015 australia collapse at trent bridge how twitter reau http http", ashes refer to a Test cricket series played between England and Australia, Trent Bridge is the name of a cricket ground and collapse refers to an australian defeat. However, the model interprets this tweet as being about a bridge collapse in Australia. Another interesting observation is that http is associated with disaster tweets and removing this word may improve the performance.

The next steps proposed after the first milestone were to focus more on data exploration, feature engineering and more advanced data cleaning techniques in order to increase the performance of the models. Another step was to train simpler models such as SVMs on features extracted from text or on word embeddings in order to see if they perform better or worse than the transformer based models.

## References


[1] Real or not? nlp with disaster tweets. https://www.kaggle.com/c/nlp-getting-started/overview.

[2] pandas. https://pandas.pydata.org/.

[3] sklearn.model_selection.train_test_split. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[4] spacy. https://spacy.io/.

[5] ktrain. https://github.com/amaiya/ktrain.

[6] Hugging face transformers. https://huggingface.co/transformers/index.html.

[7] simpletransformers. https://github.com/ThilinaRajapakse/simpletransformers#usage.

[8] Arun Maiya. Text classification with hugging face transformers in tensorflow 2 (without tears). https://towardsdatascience.com/text-classification-with-hugging-face-transformers-in-tensorflow-2-without-tears-ee50e4f3e7ed.

[9] Ratan Rohith. Start from here : Disaster tweets eda+basic model. https://www.kaggle.com/ratan123/start-from-here-disaster-tweets-eda-basic-model#3.-Target-Value-Distribution.

[10] Jean-Daniel Schiano. Disaster tweets analysis for beginners wip. https://www.kaggle.com/jdesert/disaster-tweets-analysis-for-beginners-wip/data#Introduction-&-first-level-analysis.

[11] Thilina Rajapakse. Simple transformers — introducing the easiest way to use bert, roberta, xlnet, and xlm. https://towardsdatascience.com/simple-transformers-introducing-the-easiest-bert-roberta-xlnet-and-xlm-library-58bf8c59b2a3.

[12] nxhong. tweet-predict1. https://www.kaggle.com/nxhong93/tweet-predict1.

[13] Gunes Evitan. Nlp with disaster tweets - eda, cleaning and bert. https://www.kaggle.com/gunesevitan/nlp-with-disaster-tweets-eda-cleaning-and-bert#8.-Test-Set-Labels.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[16] Zhen-Zhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

[17] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

[18] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[20] Deep pavlov - bert-base-cased-conversational. https://huggingface.co/DeepPavlov/bert-base-cased-conversational.