Twitter real disaster prediction: using natural language processing to predict real (or not) disaster-related tweets

Gonçalo Duarte[1], Marcos Neves[2] & Rafael Ribeiro[3]

[1] m20201329.novaims.unl.pt

[2] m20201781.novaims.unl.pt

[3] m20201051.novaims.unl.pt

Abstract: With the increase of the use of social networks as means of communication, organizations are monitoring Twitter for emergency cases assessment. Tweets from locals, when accurate, can give real-time information and therefore provide valuable insights into disaster events. In order to identify real from fake disaster-related tweets the data science loop was considered. For that, an exploratory data analysis and preprocessing was performed to better understand the scope of the tweets we had available for the training phase. With that, several neural network models were tested to access the best configuration for the problem at hand. Glove with LSMT showed the least overfitting scenario, with it a hyperparameter search space was configured so that Hyperband tuner determined the best hyperparameters and topology of the neural network. With the optimal parameters, a new network was configured which performed with F1-score performance above 80% over unseen data.

Keywords: Twitter, Disaster, NLP, Neural Networks

I. Introduction

With the increase of data availability and worldwide spread of information, it is fundamental to be able to distinguish right from wrong ones. Namely in the disaster and emergency related fields, organizations are studying Twitter as a potential disaster risk reduction and management tool. [1] [2]

Locals can serve as direct witnesses and provide valuable information through photos, videos and messages but it should be noted that it is not always clear if a tweet is truly related to a disaster or if, for example, keywords are out of context. Consider the following example: "'By accident' they knew what was gon happen" [3]. This tweet despite having keywords related to real disasters, e.g., "accident", "happen", is a fake disaster tweet. With the integration of machine learning solutions, these data can be collected, processed, analyzed and improve the assessment of a disaster situation and consequently manage it accordingly [4].

Considering this, the goal of this project is to create an algorithm that is able to distinguish real from fake disaster-related tweets.

II. Data

a. Description and extraction

The data was retrieved directly from Kaggle competition “Natural Language Processing with Disaster Tweets”. This dataset has 10,000 tweets that were hand classified as True, for real disaster tweets, and False, for fake or non-related tweets [5]. That being said, we choose this dataset for its high availability, since it is a publicly available dataset, and for its size, ideal for the purpose of this project, which will only require a regular computation power and a Google Colab Notebooks environment with Keras.

The dataset is divided into:

- Training dataset: 7617 rows with 5 columns;
- Test dataset: 3263 rows with 4 columns;

with the following columns:

- Id – that uniquely identifies the tweet;
- Keywork – that represent the keyword of the tweet;
- Location – that represent the location where the tweets has been written;
- Text – that represent the entire text of the tweet;
- Target – that represent the label of our problem (only present in the training dataset).

b. Exploratory data analysis

Over the training dataset, the label has a dispersion of 57/43 % which can be considered a quite balanced dataset. This is relevant for the decision over the models’ performance metrics since no label seems to be more important or prevalent than the other.

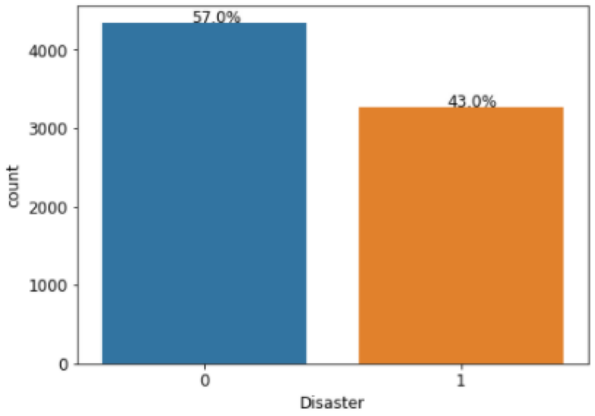

Fig 1. Barplot of the data distribution of the two labels: real and fake disaster-related tweet.

As the average tweet has around 100 words, the following plot shows that there is no difference in the word count between disaster and non-disaster tweets.

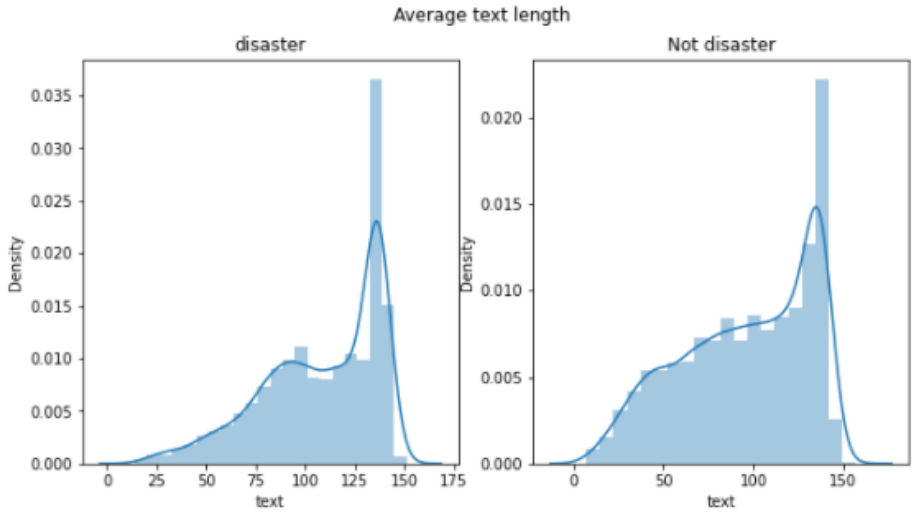

Fig 2. Plot of the distribution of the tweets' text length for real and fake disaster-related tweets.

The same applies for word length per tweet, as seen in the picture below.

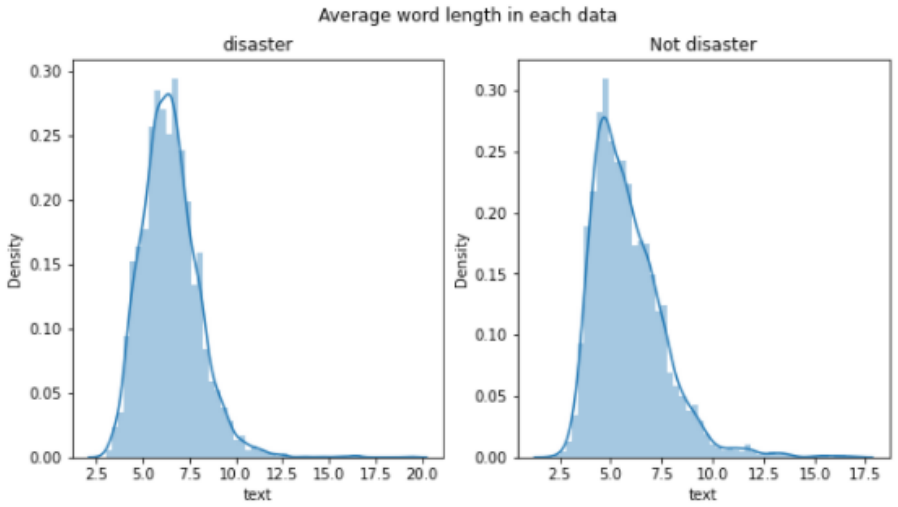

Fig 3. Plot of the distribution of the tweets' average word length for real and fake disaster-related tweets.

c. Transformation

The first transformation done to the data was to create a training and test dataset, from the initial training set. With that, we aim to ensure that the preprocessing of the training and test datasets are independent and that performance metrics can be derived from the test dataset.

Then some data cleaning was performed to the tweets' texts, removing some characters and items non-relevant to the predictability of the problem. Just like, website links and special characters as hashtags and punctuations.

With the cleaned texts a Tokenizer was fitted so that latter the text could be transformed to a sequence of integers (done with method *texts_to_sequences*). We used Keras Tokenizer, since Keras library also offers this preprocessing tool. The Tokenizer generates a word dictionary (named, *word_index*) where each word corresponds to an integer such that lower integers mean more frequent words. This is a critical step as neural networks only work with numbers as inputs. Another important remark is that the Tokenizer word dictionary derived from the training dataset is also applied to the test dataset, which simulated a real-life situation where test data is not available when generating the initial tokenization for training phases. After the lists of integers is generated, function *pad_sequences* was employed to turn it into a 2D Numpy array with a shape of the number of samples by the length of the longest sequence/tweet in place. Finally, the padded object was vectorized and recast as *float32* to ensure compliance with the data insertion requirements of the neural networks. Vectorization ensures that the data has only boolean values where each column corresponds to one word, from the most common to the least.

III. Results and Discussion

a. Neural network models

Neural networks are a type of iterative algorithms that aim to address machine learning process such as the binary classification of the problem at hand. Neural networks are composed of input, hidden and output layers of neurons which receive data and try to capture the hidden rules behind the data to best achieve the desired outcome. For that Keras library was used since written in Python, which best feature is the rapid scripting and modeling feature that allow quick and intuitive experimentation of several neural networks for a given problem.

First, neural network training performed by Keras does not provide an F1-score as a performance metric for monitoring the training phase of the networks. So, for that, an F1-score function was defined to be included in the compile phase of the networks to be taken into consideration. F1-score is a performance metric that equally weights both accuracy and precision, in other words, giving equal importance to being correct when predicting and not failing at predicting disaster tweets. We consider this option the best, as we considered both true-positives (identifying a correct disaster tweet) and true-negative (identifying a fake or non-related disaster tweet) as equally important. Additionally, early stop measures were defined considering the f1-score of the model with “patience” of 10 epochs to be conservative, to time-constraint the training epochs and therefore reduce the training time.

For the modelling phase six neural network models were experimented for the given problem of binary classification:

- Multilayer perceptron (MLP)

By building a simple MLP model for this problem we aim to showcase the most simplistic case scenario for modeling with neural networks for binary text classification. A simple two layer 16-neurons network with a dense output layer with one neuron and activation function sigmoid to ensure binary classification was configured. Despite good performance on the chosen metric, this model showed a lot of overfittings Then several modeling improvements were considered and tested as shown below.

- LSTM

LSTM, as long short-term memory neural networks, are a type of recurrent network that aims to derive information from the relations between sequential inputs both from close neighbors as from distant inputs. That being said, they can be ideal for text classification as they can derive semantic and grammatical information for words sequential inputs, as proposed by [6]. By using layers of embedding and encoding in LSTM, the model tries to find the actual meaning from the sequence of words. Dropout layers were added because as from [7] they perform well when reducing overfitting when compare to other regularization methods. Nevertheless, Lasso regression, l1, and Ridge regression, l2, were considered which aim to reduce overfitting by impacting the gradient descent phase by controlling variable weights and eventually streamlining the training phase to high valuable variables.This model has a worst performance than the previous one with no improvements in the distance between validation and training metrics.

- Bidirectional LSTM

Bidirectional LSTM are a variant of regular LSTMs where two models are trained, the first to train the regular sequence of inputs and the latter to train on the reverse sequence, as in [8]. After this, both models are combined to a single model over a specified merge method, for our problem we used sum. This can be advantageous to text classification as it can provide insights to the model about reverse text structure which might be helpful. Again, we didn't gain much with considering direct and reverse information.

- GloVe with LSTM

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. It tries to represent the semantic relationships between words from the co-occurrence matrix between words on texts [9]. Hence, embedding the neural network with GloVe builds over the pre-trained relationships to feed the network linear substructures between words which ultimately helps it improve over the performance metrics. Additionally, as regularizer we used *SpatialDropout1D* which works just like a regular dropout but drops the 1D feature maps instead of individual elements. It promotes independence between feature maps and does not decrease the learning rate of the model like a regular dropout.

- 1D Convolutional Neural Network

1D Convolutional neural networks (CNN) can be applied to text machine learning as they aim to derive spacial relationships between inputs. This can be applied to our problem as CNN will search the word embeddings and generate vectors which can be fed to dense layers.

The model that showed the best performance over the validation dataset was the **GloVe with LSTM** as the image 5 and 6 show. Validation datapoints are the closest to the training sets datapoints. Compared to the MLP model which had the best final F1-score for the validation dataset (see fig. 4), we can state that there the GloVe-based model has less overfitting. This model ended up having the same issues of the LSTM and Bidirectional LSTM.

	precision	recall	f1-score	support
Real Disaster	0.76	0.93	0.84	870
Fake Disaster	0.87	0.62	0.72	653
accuracy			0.80	1523
macro avg	0.82	0.77	0.78	1523
weighted avg	0.81	0.80	0.79	1523

Fig 4. Classification report of the GloVe with LSTM algorithm performance.

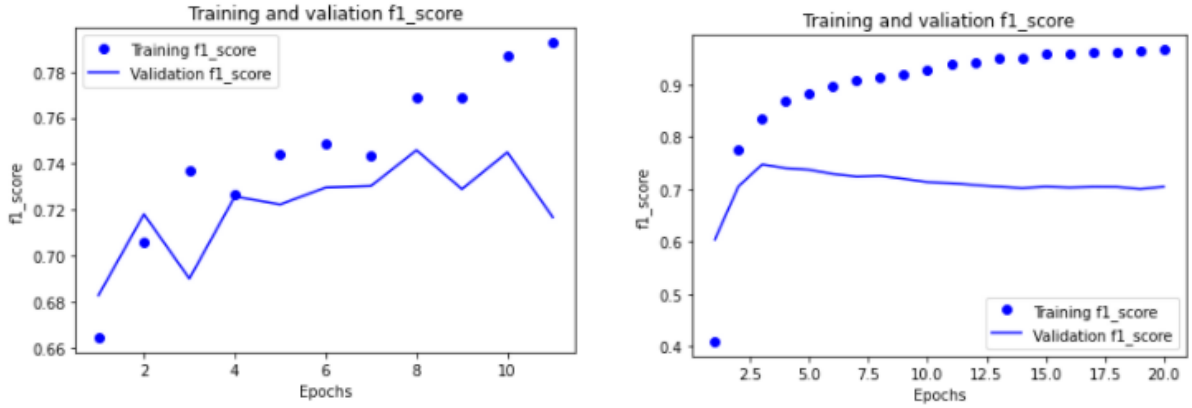

Fig 5, 6. Results of the F1-score in the training and validation set for the GloVe with LSTM algorithm and MLP model respectively.

b. Hyperparameter tunning

With *keras_tuner* library, a hyperparameter optimization framerwork designed for Keras, we defined the search space for the parameters we desired in the neural network which will be then explored using a search algorithm. We considered the Hyperband method as [10] considers it less computationally intensive than other keras-tuner's search algorithms, like the Bayesian optimization. Not only we can optimize parameters of the network, like the dropout rates, LSTM units, or the optimizer but also the topology of the network itself. By considering in the search space as parameters the number of layers and dropouts to be considered. As the objective function of the search algorithm, we choose to maximize the validation F1-score. Considering a performance metric from the validation set, aims to ensure that the optimization is not actually overfitting the model parameters and the topology to the training set.

```
Trial 508 Complete [00h 01m 49s]
val_f1_score: 0.754648745059967

Best val_f1_score So Far: 0.7753095626831055
Total elapsed time: 04h 17m 26s
INFO:tensorflow:Oracle triggered exit
{'dropout_rate': 0.2, 'LSTM_units': 75, 'LSTM_dropout_rate': 0.30000000000000004, 'LSTM_recurrent_dro
pout_rate': 0.1, 'layers': 2, 'Dense_units_0': 7, 'Dense_act_0': 'elu', 'dropout_0': False, 'optimize
r': 'rmsprop', 'Dense_units_1': 11, 'Dense_act_1': 'elu', 'dropout_1': True, 'dropout_rate_1': 0.1,
'dropout_rate_0': 0.2, 'Dense_units_2': 13, 'Dense_act_2': 'elu', 'dropout_2': False, 'dropout_rate_
2': 0.30000000000000004, 'tuner/epochs': 34, 'tuner/initial_epoch': 12, 'tuner/bracket': 2, 'tuner/ro
und': 1, 'tuner/trial_id': 'd47842a655a89899bdca4dc6f3a19da1'}
```

Fig 7. Tuner's best parameters performance on the validation set.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 15, 100)           1485600

 spatial_dropout1d_3 (Spatia  (None, 15, 100)          0
 lDropout1D)

 lstm_3 (LSTM)               (None, 75)                52800

 dense_9 (Dense)             (None, 7)                 532

 dense_10 (Dense)            (None, 11)                88

 dropout_3 (Dropout)         (None, 11)                0

 dense_11 (Dense)            (None, 1)                 12

=================================================================
Total params: 1,539,032
Trainable params: 53,432
Non-trainable params: 1,485,600
_________________________________________________________________
```

Fig 7. Final model topology

The tuner made 508 trials to get to the best parameters to which it found the best validation F1-score. With it, a final model was training over the entire training dataset without test or validation dataset. Then, the predicted values were fitted for the initial test dataset. This prediction was submitted on the Kaggle competition related with this dataset and the submission had a F1-score performance of 0.80294.

IV. Conclusion

For the dataset considering disaster tweets, a neural network configuration was designed using Keras library. The most suited model ended up being LSTM with GloVe embedding with regularizers, which provided good results in terms of overfitting for few epochs. This topology and configuration were then fine-tuned with a Hyperband method in an effort to optimize the hyperparameters. The resulting model was shown to have good results in unseen data, which suggests further developments can be done to improve and that disasters veracity can be effectively accessed from tweets with neural networks.

As further work and directions of improvement the following options can be considered:

- Better and more complex preprocessing methods could be employed to this dataset:
  - Removal of duplicated and non-relevant characters;
  - Identification and correction of spelling errors;
- Identification and exploration of other neural network configurations which can be best suited for the dataset;
- Increase the search space of the tuner algorithm;
- Consider not only parameters of the neural network layers and compilation but also preprocessing parameters for the tunning process, e.g., the maximum number of word or the batch size.

V. References

[1] Cooper, G.P., Yeager, V., Burkle, F.M. and Subbarao, I. (2015). Twitter as a Potential Disaster Risk Reduction Tool. Part I: Introduction, Terminology, Research and Operational Applications. *PLoS Currents*.

[2] Panagiotopoulos, P., Barnett, J., Bigdeli, A.Z. and Sams, S. (2016). Social media in emergency management: Twitter as a tool for communicating risks to the public. *Technological Forecasting and Social Change*, 111, pp.86–96.

[3] Twitter. (n.d.). Retrieved December 9, 2021, from https://t.co/Ysxun5vCeh

[4] Onorati, T. and Díaz, P. (2016). Giving meaning to tweets in emergency situations: a semantic approach for filtering and visualizing social data. SpringerPlus, 5.

[5] *Natural language processing with disaster tweets*. Kaggle. (n.d.). Retrieved December 15, 2021, from https://www.kaggle.com/c/nlp-getting-started/overview

[6] Yuandong Luan, Shaofu Lin, "*Research on Text Classification Based on CNN and LSTM*", Published in: 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 29-31 March 2019

[7] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). "*Dropout: a simple way to prevent neural networks from overfitting*". The journal of machine learning research, 15(1), 1929-1958.

[8] Rafał Pronko, "*Simple Bidirectional LSTM Solution for Text Classification", Proceeding of the PolEval 2019 Workshop, pages 111-1192019, Warszawa*

[9] Jeffrey Pennington, Richard Socher, Christopher D. Manning, "*GloVe: Global Vectors for Word Representation*", Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, October 25-29, 2014, Doha, Qatar.

[10] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar 2018, *"Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization"*, Journal of Machine Learning Research 18 (2018) 1-52, from https://arxiv.org/abs/1603.06560