DEEP LEARNING CHALLENGE II

Final Report – Kaggle case

10-01-2021

Kristina Krasteva
Lia Boyadzhieva

Table of Contents

Abstract

The access to data on the Internet that people have nowadays has led social media websites to become one of the main sources of information. Spreading news has become easier and faster than never before. However, with this comes the risk of the society being misled and given access to fake news. The purpose of this paper is to find out if Natural Language Processing (NLP) techniques can be used for determining which disasters are real and which ones are fake. The scope of this research is defined by the Disaster Tweets dataset available on Kaggle. For the execution of this Deep Learning Challenge, the standard IBM CRISP-DM methodology was used. Through an applied research and a comparison of various NLP classification methods, the results showcased that machines can definitely play a big role in distinguishing which Tweets are about real disasters and which ones are not.

1. Introduction

This Data Challenge is inspired by a prediction competition in Kaggle – Natural Language Processing with Disaster Tweets.

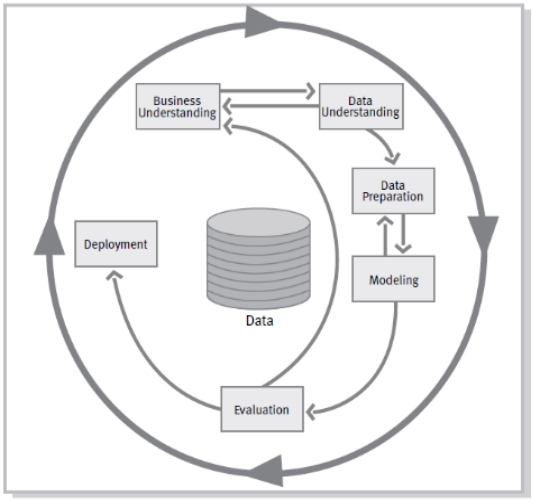
Twitter has become an important communication channel in times of emergency. The pervasiveness of smartphones enables people to announce an emergency they are observing in real-time. Because of this, more agencies such as disaster relief organizations and news agencies, are interested in programmatically monitoring Twitter. But it is not always clear whether a person's words are announcing a disaster. For instance, a person can tweet about the sky being ablaze. However, here the word "ablaze" is used metaphorically. For the people reading this post is clear what the author meant but is not that clear for a machine to distinguish figuratively meanings.

The goal for this challenge is to build a machine learning model that predicts which Tweets are about real disasters and which ones are not by predicting (1) for disaster and (0) for not. For the purpose Natural Language Processing will be applied.

The dataset that is going to be used consist of 10,000 tweets that were hand classified. The columns in both the train and test csv files are "**id**" - a unique identifier for each tweet, "**keyword**" - a particular keyword from the tweet (may be blank), "**location**" - the location the tweet was sent from (may be blank), "**text**" - the text of the tweet. Furthermore, the train dataset has a column "**target**" which designate whether a tweet is about a real disaster (1) or not (0).

2. Methods

For this Deep Learning Challenge the standard IBM CRISP-DM methodology was used. There are 6 phases to be considered – **Business understanding, Data understanding, Data preparation, Modeling, Evaluation and Deployment**. These stages are linked to one another to help us solve the particular data science problem.



Additionally, we intent to focus mainly on the first 5 phases since there we can be more flexible when it comes to the project's iteration. Below a more in-depth description of what are the major task per stage is shown.

2.1. Business understanding

Before implementing any solution, the problem needs to be understood and the goal to be determined. This step is of great importance as it will result in a more reliable starting point for the actual modeling.

For this part the following things have to be done:

2.2. **Determine the Business Objectives** – Background (what is the influence and impact of disaster tweets), Business Success Criteria (aimed for Disaster Relief Organizations/News Agencies)

- Deep Learning Challenge Plan

The Plan took till the end of week 11 to get finished. The first thing we as a team had to do was to look up the challenges in Kaggle. When we selected several of them we had a discussion what is the potential value of each challenges regarding our learning progression. For instance, one of our options was the Google Landmarks challenge but we decided against it as its focus is on CNN which we are already actively applying in our group project. And since both of us are interested in NLP but have only used it in one Core Program exercise, we wanted to improve our skills and expand our knowledge. We both liked the Disaster Tweets NLP challenge and started to brainstorm a more specific goal for this challenge.

The plan was prepared by both of us – Lia writing the description, goal and expected results, and Kristina taking care of the approach and the timeline. After we were finished we did a peer review of each other's chapters and suggested what could be improved.

Throughout this week we asked and received feedback multiple times – verbally on our initial idea, and written feedback on the plan document itself which we intend to apply in the upcoming weeks.

- Background research

In week 12, we were mainly focused on researching what is the influence on Twitter and the impact that disaster tweets can have.

Twitter was brought to the horizon of the Internet in 2006 and ever since then it has become one of the most famous social website with millions of users around the globe. Despite the fact that other medias such as Instagram, Tiktok and Facebook have far more users, the traffic of information which Twitter accumulates is still impressive and relevant. What differentiates Twitter from the other social networks mentioned is that it has this kind of openness to everyday news and information. While Facebook is more focused on the people one is acquainted to, Tiktok is based on algorithms showing short videos of one's interests and Instagram is mainly about sharing photos, Twitter spreads information which can be discovered with the help of hashtags.

In recent years, this has led to Twitter becoming an integral part of the information flow. The news are no longer only controlled by the newspapers and tv, but also by the Twitter users. They can post, comment, and share about local and global events almost in real time. Exactly here we are faced with the challenge of determining which disaster tweets are real and which are not.

With this data challenge, we aim to explore the power of AI and specifically NLP and test if the models such as BERT, XGBoost, and LSTM are able to distinguish between tweets with real information about disasters and ones where it was meant metaphorically.

Additionally, we will also prepare Constraints and Risks tables which can be seen below.

2.3. **Assess Situation** – Inventory of resources (both students working on the Data Challenge, taking into account how qualified we are), Constraints and Risks tables.

Since both of us had no previous experience with NLP, we had to first read and understand the theory behind some of the main techniques used. We had to get familiar with terms such as tokenization, lemmatization, stop words, word vectors, etc. in order to apply them in our code later.

Furthermore, as both of us wanted to actively take participation in all of the stages of development, the work was split between us. For the EDA, Data Preparation and Modeling part each of us had a separate Jupyter notebook, which were later combined in one. The contribution of each of us can be seen in GitHub where we regularly updated the code.

- Constraints

  The purpose of the below mentioned project constraints is to outline what restricts or dictates the actions of this data challenge. There are some general constraints that can occur in almost every project, but we have also included some more specific ones which are tightly related to the NLP Disaster Tweets challenge.

Constraints Table	
What	**Why**
Time constraint	The Deep Learning Challenge must be finished within 7 weeks. It starts on November 16, 2021, and is expected to be completed on January 21, 2022.
Dataset quality	The dataset provided is from Kaggle and even though it has quite a lot of records, there are very few features. This can lead to poorly performing algorithms, but we plan to extend the dataset with extracted features.
Resource constraint	The people (our team), equipment – computers, Internet, available data etc., knowledge which are required to successfully finish the challenge.

- Risks

  Before starting to work on the Deep Learning challenge more in-depth, certain project risks are acknowledged. This is needed because in its essence a risk is an uncertain event or condition that, if it occurs, influences at least one project objective. In the table the risks are outlined, the effect that they can cause in terms of damage to the final product and development, the probability of such risk to happen and the mitigation method or how to resolve such risks. Effect and probability will be measured as Low, Medium, and High.

Risk Table			
Risk	**Effect**	**Probability**	**Mitigation**
Bad quality of the dataset which causes troubles with the prediction models	High	Medium	Explore and try to improve the dataset by adding more data to it.
None of the models achieve the desired accuracy	High	Medium	Re-evaluate the models and spend more time on training data, try different machine learning approaches.
The dataset is not cleared and prepared properly	Medium	Medium	Iterate the process, research on different techniques to prepare the dataset in the most optimal way.
Lack of required knowledge of student.	Low	Medium	Perform research activities online. Additionally, ask the other team member for help.
Behind schedule	Medium	Low	The communication between the two students happens frequently and the tutors are updated about the work done weekly. Keeping track of the progress in the report document.

2.4. Data understanding

When the business part of the process has been clarified, the data understanding stage arrives. It is an extremely essential phase since it will provide us with first insights about the data.

For this part the following things have to be done:

2.4.1. Collect Initial Data – Gathering (or accessing) the data chosen from Kaggle. List the datasets acquired, together with their locations, the methods used to acquire them, and any problems encountered.

The dataset was downloaded from Kaggle where 2 csv files are available – train.csv and test.csv; There is also a third csv file called "sample_submission", but it is targeted towards people who want to submit their challenge in Kaggle, so we decided to delete this file. The two main csv files were stored in a folder called "data" and uploaded to our GitHub. We did not face any challenges in retrieving the data as it was a pretty straightforward method and the data is open-source.

2.4.2. Describe Data – Describing the collected data (the format of the data, the quantity of data such as the number of records and fields in each table).

In week 13 we got familiar the data provided by Kaggle. There are 7614 records in train dataset and 3264 in test dataset. The total number of fields in the train dataset are five – "id", "Keyword" – serves as category of the tweet, "Location" – the country or city that is mentioned in the tweet, "Text" – the tweet, and "Target" – defines if a tweet is real or fake disaster by the values of 1(real disaster) and 0 (fake disaster). This last column is not present in the test dataset.

2.4.3. Explore Data –Exploring the gathered data by querying, visualizations (EDA).

Each of us created a Jupyter notebook focused on exploring the data and visualizing it. In this way we were able to understand it.

2.4.4. Verify Data Quality – To verify the data questions like "Is the data complete?", "Are there any missing values?", "Does the data contain errors and, if there are errors, how common are they?" must be answered and reported.

For example, how many missing values and from which columns of the of both the train and test datasets were displayed. The number of fake vs real tweets were also shown. A word cloud with the most common words from the tweets were generated as well.

Since there were null values in the "Keyword" column as well as in the "Location" we checked if that is common for defining whether the tweet is disaster or not. The total null values in "Keyword" were 61. We plotted the null values only where the tweet is marked as fake disaster and the number of values decreased to 19. Therefore, it can be said that the tweet is not defined as a fake one based on whether it has a keyword. Moreover, we decided that we can fill in manually the missing values for "Keyword". It was not difficult to decide what the keyword should be since most of them were about real disaster tweets.
However, regarding the "Location" there were in total 2533 null values which 1458 are for fake disaster tweets. From here comes the thought that when a tweet is about a real disaster most of the times it is mentioned where this disaster has happened. Still the null values are too many and will be difficult to aim the model.

2.5. Data preparation

This stage is about preparing the final dataset that will be fed into the model. Practice shows that the data preparation tasks are likely to be performed multiple times.

For this part the following things have to be done:

2.5.1. Select Data – Decision on what data will be used for the analysis. The criteria to be considered includes relevance to the end goals, quality, and technical constraints.

For two weeks (14 and 15), we are cleaning the dataset and in this way we hope to prepare a better train dataset to be used for our future models. We have done this individually and later we will compare results and give suggestions to each other what can be added as a cleaning step.

2.5.2. Clean Data – Raising the quality of the collected data by removing the irrelevant data.

The first actions taken was to clean the tweets by making all words lowercase, remove hyperlinks, hashtags, mentions, emojis, punctuation signs and all kinds of symbols and numbers which will not be useful later. This removal was conducted with the help of regular expressions.

Furthermore, we have removed any stopwords. In our case these are the English words which do not add much value to the meaning of a sentence. They can safely be ignored without sacrificing the meaning of the sentence. The following code: "stopwords.words('english')" removes all stopwords in the English language such as "the, a/an, our, we,", etc.

Thanks to the word cloud visualization, we have noticed that specifically in our dataset the words "u", "im", "c" and "amp" are one of the most common ones. However, they most likely will not bring value to the training of the models, so we decided to remove them as well.

2.5.3. Construct Data – Description of any new data records that were made so to make the data more appropriate for the future model.

Moreover, in the "Keyword" column there were a lot of records with "%20" between two words which stands for space. We replaced it with underscore for better interpretation and visualization.

2.5.4. Integrate Data – Describe any instances where the data is combined from multiple tables or records to create new records or values.

As part of the data preparation, we extracted the URLs and numbers used in tweets in separate columns. In this way we can use them as additional features because they might be a sign for better predictions of the model. For instance, most of the tweets which have URLs or numbers are marked as disaster.

2.5.5. Format Data – Syncing the data when all of the above-mentioned techniques are applied.

Next, tokenization was applied. The purpose of the tokenization is to break the raw text into small chunks (words called tokens). These tokens help in interpreting the meaning of the text by analyzing the sequence of the words.

Additionally, we have applied the most common pre-processing technique in Natural Language Processing – lemmatization. For this challenge the understanding of the tweet context is important. Thus, we managed to group together the different inflected forms of a word so they can be analyzed as a single item. After tokenization and lemmatization, we transformed the tokens into sentences.

2.6. Modeling

When the data is ready, the modeling happens where a number of different techniques are selected and applied, and their parameters are tuned in search for the optimal solution.

For this part the following things have to be done:

2.6.1. Select Modeling Techniques – We will select several modeling approaches based on the data, the requirements, the wanted results.

During week 15, we also started to experiment with different models. Lia applied BERT on the dataset, while Kristina used XGboost and LSTM. BERT is significantly heavier for the computer and the learning takes more than 5 hours for 7 epochs, while the other two take just a few minutes.

2.6.2. Build Model – Run the model on the dataset that was prepared from the phases above.

- *Bert*

  Recently, BERT is one of the most used models for Natural Language Processing. With the used of Transformer encoder the model can learn the context of a word based on all of its surroundings. This is exactly what we need for this challenge – to define whether a keyword for disaster is used metaphorically (fake disaster) or indicates a real disaster.

  The model was applied two times because during the first time the acquired validation accuracy was 80% with training on 5 epochs and a batch size of 10. This result was achieved by having two input layers, two dense layers and a dropout of 20% between them. Then it was decided to do some hyperparameter tuning to see if there the model will improve.

  For the second try we increased the number of epochs from 5 to 10. Additionally, we left only the last dense layer for the output, increased the shape to 128 whereas in the first attempt was 64 and set the batch size to 32. The training of the model took longer time (around 7 hours) and was stopped at epoch 8 with validation accuracy of 81%. Interestingly, from epoch 5 to 7 the validation accuracy remained 0,8096.

As next steps for the BERT model can be considered few optimizations in the dataset. Moreover, when the model will be trained again Early Stopping can be applied in case there is no improvement in the accuracy in the range of few epochs.

- *XGBoost*

  XGBoost is based on decision trees that examines the input under various "if" statements. Like other gradient boosting algorithms on decision trees, XGBoost considers the leaves of the current decision tree and questions whether turning that leaf into a new "if" statement with separate predictions would benefit the model. The benefit to the model depends on the "if" statement chosen and which leaf it's placed on—this can be determined using the gradient of the loss. The loss includes a scoring function that measures algorithm performance.

  For us, it was a really good way to test very swiftly the classification. We built a confusion matrix showing how many tweets are considered as disaster and how many as non-disaster. A pipeline using CountVectorizer to convert the text to a matrix of token counts. After that the count matrix is normalized to a tf-idf (term-frequency times inverse document-frequency) with the help of TfidfTransformer. Finally, we fitted the pipeline with the data and displayed it on the confusion matrix the output from which can be seen in chapter 3 – Results.

- *LSTM*

  Long Short-Term Memory (or LSTM) is a recurrent neural network architecture meaning it has feedback connections. It can process entire sequences of data that is fed to the LSTM token by token. The new token is propagated through the network, and it also takes into account the state of the memory cell.

  For the LSTM, we started with getting the values of the following 3 columns – "text" in the train set, "target" in the train set and "text" in the test set. From there, we created a vocabulary with a word tokenizer and calculated its length. Then, we used the100-dimensional GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia to prepare an embedding layer. Pre-trained models like GloVe enhance the performance accuracy. After splitting the dataset into train and test, we trained the LSTM model with the GloVe layer for 7 epochs and got a validation accuracy around 80%. More detailed results can be seen in the Results chapter.

2.6.3. Assess Model – When the model was executed with the specified parameters, assess the output, and tune those parameters to obtain better results.

For the three approaches, we got around the same percentage of accuracy and loss. In week 16, we decided to split the work and try to improve XGBoost and LSTM as they do not require that much computational power. Lia worked on the XGBoost by adding features (the URL and the number), while Kristina did the LSTM with the same features. In terms of accuracy, the results have not improved that much which made us think that maybe with the provided data, we would not be able to achieve a very high accuracy.

2.7. Evaluation

In this phase, the final evaluation and reviewing of the models is done. Here, the importance lays on the business value again and if everything has been sufficiently considered.

For this part the following things have to be done:

2.7.1. Evaluate Results – This step is a bit similar to the "Assess Model" one from the previous phase, but here we will focus on comparison between the different approaches (confusion matrix, plot accuracy and loss, etc.).

2.7.2. Review Process – After the results from the models are satisfactory, here if we have the chance to examine there are some other factors that had lower priority or were overlooked.

2.8. Deployment

Usually, the final step is to launch the model. However, as we are more focused on experimenting with the different approaches, our deployment phase would be more like an advice which model is the best for our particular case.

For this part the following things have to be done:

2.8.1. Produce Final Report – A final report is required at the end of the project.

3. Results

All our progress and results can be found on our GitHub repository.

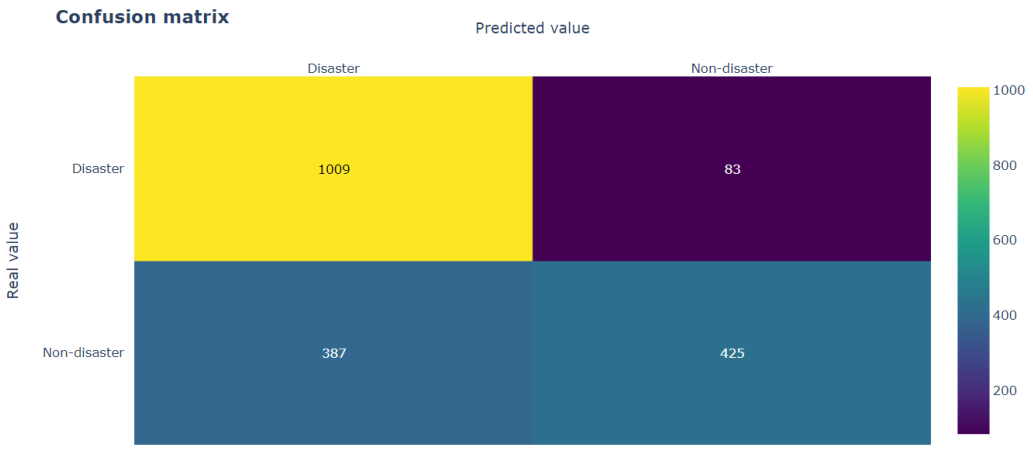
a) BERT

BERT resulted in validation accuracy of 80%.

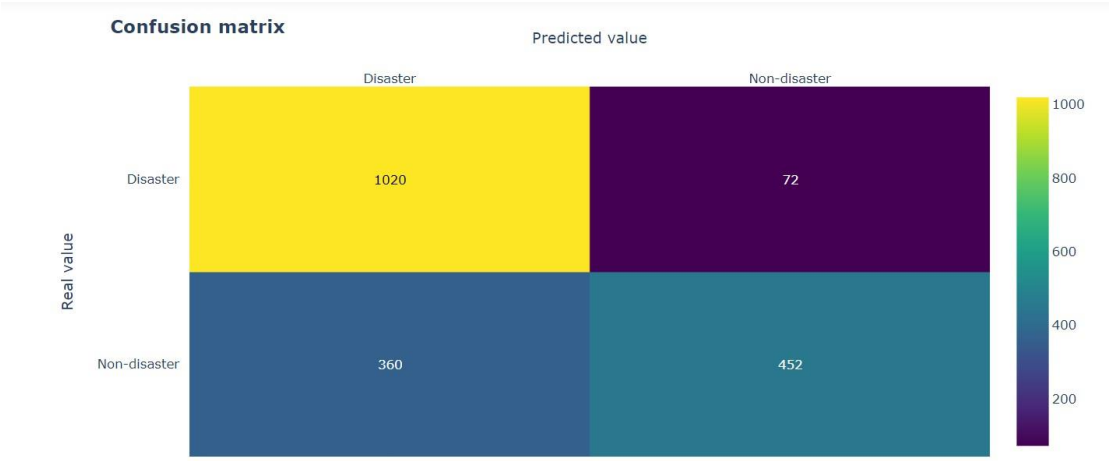
```
Epoch 1/5
609/609 [==============================] - 1736s 3s/step - loss: 0.4904 - accuracy: 0.7759 - val_loss: 0.4426 - val_accuracy: 0.8070
Epoch 2/5
609/609 [==============================] - 1265s 2s/step - loss: 0.3849 - accuracy: 0.8476 - val_loss: 0.4087 - val_accuracy: 0.8214
Epoch 3/5
609/609 [==============================] - 1273s 2s/step - loss: 0.3112 - accuracy: 0.8813 - val_loss: 0.4249 - val_accuracy: 0.8122
Epoch 4/5
609/609 [==============================] - 2946s 5s/step - loss: 0.2383 - accuracy: 0.9108 - val_loss: 0.5921 - val_accuracy: 0.7886
Epoch 5/5
609/609 [==============================] - 1290s 2s/step - loss: 0.1768 - accuracy: 0.9355 - val_loss: 0.5406 - val_accuracy: 0.8017
```

b) XGBoost

Below, you can see the Confusion Matrix from the XGBoost where it the http links are removed from the original tweet. As it can be seen from the image, this method performed very well when it comes to determining the tweets that were a disaster (1009) and only got mistaken for 83 of them. However, a big discrepancy can be seen for the non-disaster tweets. XGBoost recognized correctly only 425 tweets and the other 387 were said to be disasters when they should have been classified as non-disasters.



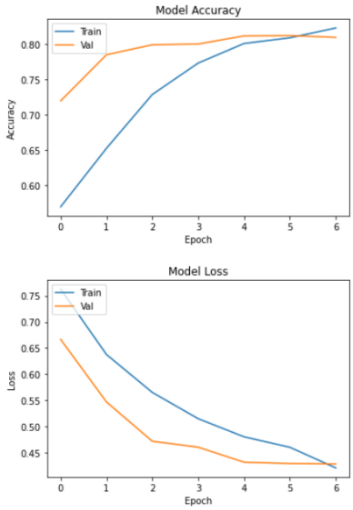
This second picture shows the results when the same technique was applied but the string "http" was kept in the original tweet. Here, we see that the outcome was slightly improved for both disaster and non-disaster classification. The conclusion we have drawn from this experiment was that with a bit of tuning and not removing all information which we preliminary thought would not be necessary, the results can be improved.



c) LSTM

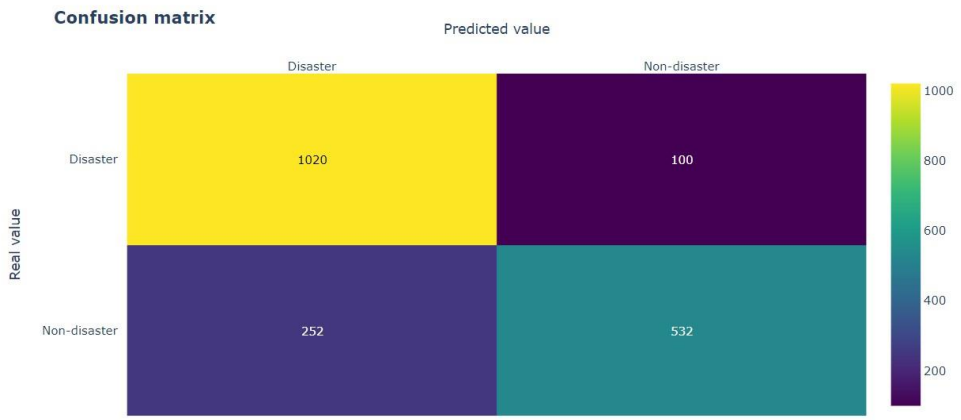
Using LSTM without any features and only with the text column altered in the data preparation section, never reached accuracy more than 81%.

```
Epoch 1/7
179/179 [==============================] - 14s 41ms/step - loss: 0.7628 - accuracy: 0.5689 - val_loss: 0.6667 - val_accuracy: 0.7195

Epoch 00001: val_loss improved from inf to 0.66670, saving model to model.h5
Epoch 2/7
179/179 [==============================] - 7s 38ms/step - loss: 0.6378 - accuracy: 0.6521 - val_loss: 0.5471 - val_accuracy: 0.7852

Epoch 00002: val_loss improved from 0.66670 to 0.54710, saving model to model.h5
Epoch 3/7
179/179 [==============================] - 7s 37ms/step - loss: 0.5653 - accuracy: 0.7287 - val_loss: 0.4720 - val_accuracy: 0.7994

Epoch 00003: val_loss improved from 0.54710 to 0.47204, saving model to model.h5
Epoch 4/7
179/179 [==============================] - 7s 37ms/step - loss: 0.5151 - accuracy: 0.7733 - val_loss: 0.4607 - val_accuracy: 0.8004

Epoch 00004: val_loss improved from 0.47204 to 0.46070, saving model to model.h5
Epoch 5/7
179/179 [==============================] - 7s 38ms/step - loss: 0.4805 - accuracy: 0.8012 - val_loss: 0.4320 - val_accuracy: 0.8120

Epoch 00005: val_loss improved from 0.46070 to 0.43204, saving model to model.h5
Epoch 6/7
179/179 [==============================] - 7s 37ms/step - loss: 0.4605 - accuracy: 0.8092 - val_loss: 0.4295 - val_accuracy: 0.8125

Epoch 00006: val_loss improved from 0.43204 to 0.42951, saving model to model.h5
Epoch 7/7
179/179 [==============================] - 6s 36ms/step - loss: 0.4211 - accuracy: 0.8233 - val_loss: 0.4287 - val_accuracy: 0.8099

Epoch 00007: val_loss improved from 0.42951 to 0.42866, saving model to model.h5
```



After adding the features, the model increased its accuracy to 81%, but what was more interesting was the confusion matrix that you can see below.

```
Epoch 1/5
179/179 [==============================] - 11s 36ms/step - loss: 0.7630 - accuracy: 0.5912 - val_loss: 0.6434 - val_accuracy: 0.7647

Epoch 00001: val_loss improved from inf to 0.64335, saving model to model_f.h5
Epoch 2/5
179/179 [==============================] - 6s 31ms/step - loss: 0.6201 - accuracy: 0.6814 - val_loss: 0.5378 - val_accuracy: 0.7857

Epoch 00002: val_loss improved from 0.64335 to 0.53780, saving model to model_f.h5
Epoch 3/5
179/179 [==============================] - 5s 30ms/step - loss: 0.5408 - accuracy: 0.7439 - val_loss: 0.4642 - val_accuracy: 0.8020

Epoch 00003: val_loss improved from 0.53780 to 0.46419, saving model to model_f.h5
Epoch 4/5
179/179 [==============================] - 6s 31ms/step - loss: 0.4996 - accuracy: 0.7809 - val_loss: 0.4405 - val_accuracy: 0.8125

Epoch 00004: val_loss improved from 0.46419 to 0.44046, saving model to model_f.h5
Epoch 5/5
179/179 [==============================] - 6s 31ms/step - loss: 0.4704 - accuracy: 0.8071 - val_loss: 0.4397 - val_accuracy: 0.8162

Epoch 00005: val_loss improved from 0.44046 to 0.43971, saving model to model_f.h5
```



In comparison with the XGBoost, which had quite some trouble with identifying the non-disaster tweets, with this version of the LSTM model, it can clearly be seen that the mistaken non-disaster tweets are less.

4. Discussion

5. Feedback Log

Activity	Date	Feedback received	Future improvement
Initial idea and plan	16/11/2021	The chosen Kaggle challenge is appropriate for the assignment. However, be more specific about the end goal since it is not very clear what we want to achieve with the challenge. Set the goal to be realistic but also concrete. Think about the planning and try to submit the Data Challenge Plan before the deadline if we want to receive feedback.	We tried to define our goal and what learning outcomes we want to cover. Also, we submitted the plan earlier in order to receive feedback.
Data Challenge Plan	19/11/2021 & 23/11/2021	From Livia – Not very clear what is meant by predicting "not real disasters". Submit the documents in PDF as it is more professional. From Bartosz – The third chapter for the approach is quite generic and not very precise for the project itself	We specified what we understand as "not real disasters" and how it translates to our goal. For the approach section of the plan, we are expanding it and making it more concrete in the final report.
First EDA	1/12/2021	From Livia – We are on track with the challenge as we have already completed the first version of our EDA.	We want to add some more visualizations to show the difference after the tweet texts were pre-processed.
Second EDA	7/12/2021	From Bartosz – We might have discarded some of the information in the text. Instead of deleting it, try to extract it since it might be useful in the future modeling. Make some more meaningful visualizations to show how many missing values are there location per tweet type (disaster/non-disaster) and see if there is some kind of a correlation.	We proceeded with our final EDA by applying Bartosz feedback. We extracted the URL and numbers from the tweet text to new columns which might later be used as features. Moreover, by plotting the location we noticed that when a tweet is a real disaster most of the times it is mentioned where this disaster has happened.
First modeling experiments	14/12/2021	From Bartosz – It is good that we have tried different approaches (BERT, LSTM, XGBoost). Since we get almost the same accuracy for all of them (~80%), focus on improving them, rather than trying new models. See where things need a bit more tuning (i.e., XGBoost not very good with classifying fake disasters).	We left the http in the text column to see if the accuracy improved. It did improve and now we want to focus on working with features for the LSTM and XGBoost as they are executed fast and we can test easily in this way.

First modeling experiments	15/12/2021	From Livia – It is good that you are still on track and making progress. Keep writing the final document and submit it even if it is not the final version. For the BERT model, look up at "Early stopping".	We have applied early stopping to the model and also plan to submit the document for feedback after the holidays.

6. Conclusion