DEEP Learning Challenge ii

Final Report – Kaggle case

18-01-2021

Kristina Krasteva

Lia Boyadzhieva

Table of Contents

[Abstract 3](#_Toc93156438)

[1. Introduction 4](#_Toc93156439)

[2. Methods 5](#_Toc93156440)

[2.1. Business understanding 5](#_Toc93156441)

[a) Determine the Business Objectives 5](#_Toc93156442)

[b) Assess Situation 6](#_Toc93156443)

[2.2. Data understanding 8](#_Toc93156444)

[a) Collect Initial Data 8](#_Toc93156445)

[b) Describe Data 8](#_Toc93156446)

[c) Explore Data 8](#_Toc93156447)

[d) Verify Data Quality 10](#_Toc93156448)

[2.3. Data preparation 10](#_Toc93156449)

[a) Select Data 10](#_Toc93156450)

[b) Clean Data 10](#_Toc93156451)

[c) Construct & Integrate Data 11](#_Toc93156452)

[d) Format Data 11](#_Toc93156453)

[a) Select Modeling Techniques 11](#_Toc93156454)

[b) Build Model 12](#_Toc93156455)

[c) Assess Model 13](#_Toc93156456)

[2.4. Evaluation 13](#_Toc93156457)

[a) Evaluate Results 13](#_Toc93156458)

[b) Review Process 14](#_Toc93156459)

[2.5. Deployment 14](#_Toc93156460)

[a) Final report & presentation 14](#_Toc93156461)

[3. Results 15](#_Toc93156462)

[4. Discussion 19](#_Toc93156463)

[5. Feedback Log 21](#_Toc93156464)

[6. Conclusion 22](#_Toc93156465)

[7. References 23](#_Toc93156466)

List of figures

[Figure 1 CRISP-DM structure 5](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157644)

[Figure 2 Example of disaster & non-disaster tweet 9](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157645)

[Figure 3 Keywords target distribution 9](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157646)

[Figure 4 Features extraction 11](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157647)

[Figure 5 Text processing 11](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157648)

[Figure 6 Model Accuracy & Loss on first BERТ 14](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157649)

[Figure 7 Model Accuracy & Loss on unlemmatized BERT model 14](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157650)

[Figure 8 First BERT attempt 15](#_Toc93157651)

[Figure 9 Second BERT attempt 16](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157652)

[Figure 10 BERT on unlemmatized text 16](#_Toc93157653)

[Figure 11 XGBoost on text column without http 17](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157654)

[Figure 12 XGBoost on text column with http 17](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157655)

[Figure 13 XGBoost on all columns 18](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157656)

[Figure 14 LSTM accuracy on text column 18](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157657)

[Figure 15 LSTM conf matrix on text column 19](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157658)

[Figure 16 LSTM conf matrix with features 19](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157659)

[Figure 17 LSTM accuracy with features 19](file:///C:\Users\krist\Desktop\ADS_Minor\WEEKS\DC2\DC2_DisasterTweets\documents\DC2_Report.docx#_Toc93157660)

List of tables

[Table 1 Constraint Table 7](#_Toc93154002)

[Table 2 Risk Table 8](#_Toc93154003)

[Table 3 Feedback Log Table 22](#_Toc93154004)

# Abstract

The access to data on the Internet that people have nowadays has led social media websites to grow into one of the main sources of information. Spreading news, as never before, has become easier and faster. 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 whether Natural Language Processing (NLP) techniques can be used for determining which disasters are real and which ones are not. 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 between actual tweets and metaphorically written ones.

# Introduction

This Deep Learning Challenge is inspired by a prediction competition in Kaggle – [Natural Language Processing with Disaster Tweets](https://www.kaggle.com/c/nlp-getting-started/overview).

Twitter has become an important communication channel in times of emergency. The pervasiveness of smartphones enables people to announce an emergency they are witnessing 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 conveying a literal meaning. 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 it is clear what the author meant but it is not that clear for a machine to distinguish figurative meanings.

The goal of 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 train dataset that is going to be used consist of 7614 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).

# Methods

Diagram

Description automatically generated

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.

Figure 1 CRISP-DM structure

## 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 been done:**

1. **Determine the Business Objectives** – Idea, Planning, 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 structured 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 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.

1. **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 extremely limited 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. We sat up a [**GitHub repository**](https://github.com/krisxtinaa/DC2_DisasterTweets) where all our code was stored. 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 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 18, 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. |
| Scope constraint | The challenge focuses on the hand classified tweets and all predictions are made based on them. |
| Resource constraint | The people (our team), equipment – computers, Internet, available data etc., knowledge which are required to successfully finish the challenge. |

Table 1 Constraint Table

* 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. |

Table 2 Risk Table

## 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:**

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.

1. **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. The “keyword” and “location” are not mandatory therefore there were quite few rows where the information was not filled in.

1. **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.

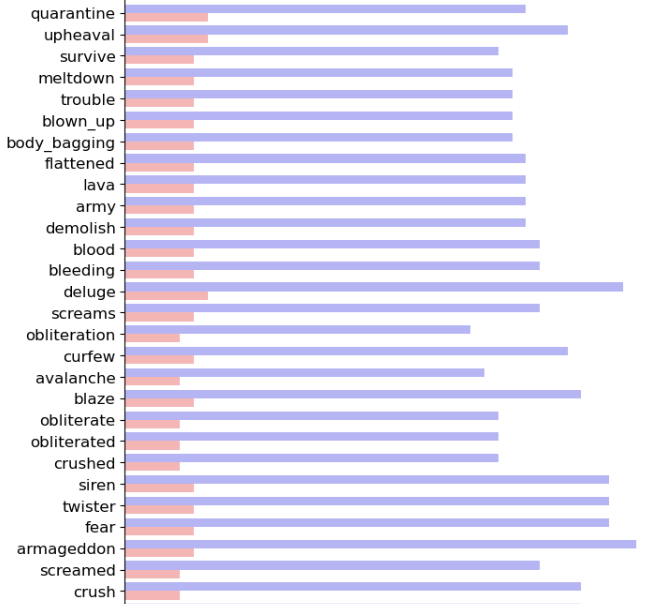
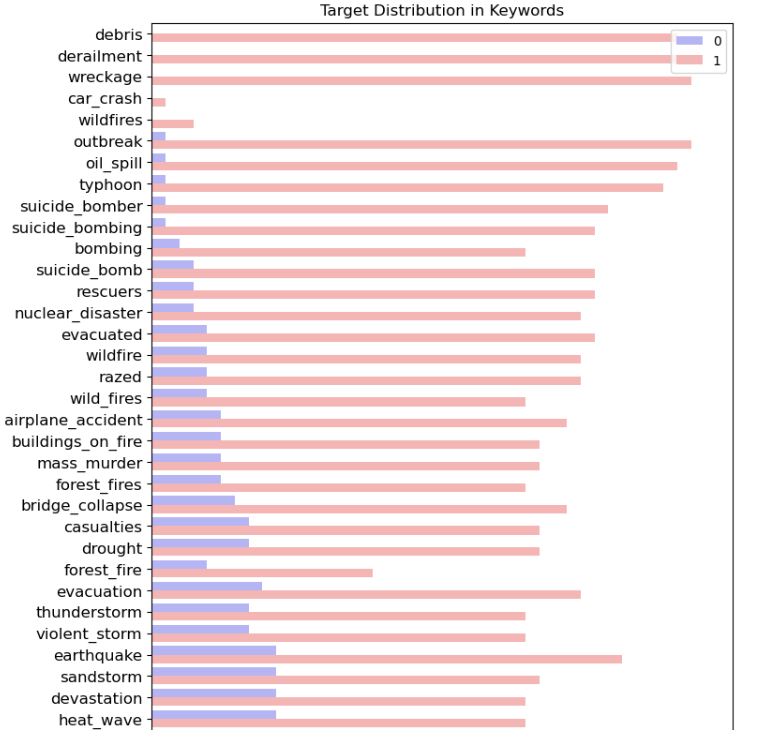
Graphical user interface, text, application, email

Description automatically generatedWe started with something as simple as displaying how a sample of real and non-disaster tweets look like. Immediately, it can be noticed that disaster tweets are more formal, having more information and links in them.

Figure 2 Example of disaster & non-disaster tweet

Additionally, we have visualized the target distribution in keywords and words such as “outbreak”, “typhoon”, “suicide\_bomber/bombing” were mainly relating to real disaster. On the other hand, “trouble”, “blown up”, “blaze”, “fear”, etc. were used more frequently for non-disaster tweets.

Figure 3 Keywords target distribution



1. **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. A graph showing the number of “not real” vs “real” tweets was 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 there was a correlation between the fact that they were missing and the target of the tweet. The total null values in “Keyword” were 61. We plotted the null values only where the tweet is marked as non-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 from which 1458 are for non-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 it would be difficult and time-consuming to fill them in.

## 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:**

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 compared results and give suggestions to each other what can be added as a cleaning step. We focused on the “text” and “target” column of the train set.

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

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

Furthermore, we have removed any stop words. 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 stop words 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.

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

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.

Application

Description automatically generated with low confidenceMoreover, we created the following three columns – “text length”, which indicated how many words are there in each tweet, “URL” column, which extracts the URL address if it is present in a tweet, and “number”, extracting the number from the tweet.

Figure 4 Features extraction

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

Next, tokenization was applied. The primary purpose behind the tokenization is to break the raw text into small chunks called tokens. These tokens are used for the model to be able to interpret the meaning of the text by analyzing the sequence of the words.

Graphical user interface, text, chat or text message

Description automatically generatedAdditionally, we have applied the most common pre-processing technique in NLP – 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.

Figure 5 Text processing

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:**

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. We selected these techniques based on the common practice and the fact that heavier models such as Roberta, Albert, StructBert would take around a day to train. We wanted to avoid that since dedicating so much time and achieving about the same results would not be optimal for the period of time we had for the execution of the challenge.

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

* *Bert*

Recently, BERT has become one of the most used models for Natural Language Processing. With the use 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 (non-disaster) or indicates a real one.

Firstly, the model was trained 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 we 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 – around 7 hours and was stopped at epoch 8 since the validation accuracy did not change for the last 3 epochs in a row.

Soon after that we were advised by Olaf Janssen that the BERT model can handle unlemmatized sentences. Since until now we lemmatized the text before training the model, we decided to train one more time the model but on unlemmatized text and see if the results change. For the purpose we used the structure from the second BERT model but on 5 epochs. All further results and discussions can be seen in the next two chapters of this document.

* *XGBoost*

Together with BERT, we used XGBoost which is based on decision trees. By asking a number of “if statements” taking into consideration the provided input. When XGBoost is faced with the leaves of current decision tree, it computes whether turning that leaf into an “if statement” with separate predictions would be beneficial for the model. XGBoost is an algorithm which main purpose is gradient boosting meaning that it uses the gradient of the loss. A scoring function is applied to calculate the loss which determines the performance of the algorithm.

We built a confusion matrix showing how many tweets are considered as disaster and how many as non-disaster. A pipeline was also built 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 on the confusion matrix the output from which can be seen in chapter 3 – Results.

Moreover, we made two more models of XGBoost - one on three features – “Text”, “URL” and “number” columns. The second model was trained on all features in the dataset. In both cases, two pipelines had to be created – one for categorical features and other for numerical. Later, we combined the two pipelines with a column transformer.

* *LSTM*

Long Short-Term Memory (or LSTM) is a recurrent neural network architecture meaning it has feedback connections. The power of LSTM is that it can process whole sequences of data 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.

Additionally, we run multiple experiments with adding features to the model. However, for this model to be trained, we need an equal number of x instances and y labels. It is not possible to achieve that in the standard way. Therefore, we thought of an alternative solution which concerned the clean\_text function. In there, we removed the lines which are responsible for the deletion of numbers and urls (our 2 features). In this way, instead of using actual features, we just kept the text closer to its original form. Surprisingly, this was rather beneficial for the model and the results can be seen in the results chapter.

1. **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 and time. 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.

## 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:**

1. **Evaluate Results**– This step is a bit similar to the “Assess Model” one from the previous phase, but here we are focusing on comparison between the different approaches & confusion matrix, plot accuracy and loss.

After running the three models for the first time on the “text” column only, we compared each other’s results and noticed that there was not a significant difference between them. The lack of difference can especially be noticed in the accuracy and loss results we obtained. Nevertheless, there were some variations when it came to the confusion matrixes. This is where we were advised to focus on and try to improve with the addition of features. Adding them was beneficial for the LSTM algorithm, but not so much for the XGBoost.

1. **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.

Looking back on how we managed this deep learning challenge, we believe that the obtained results and the run experiments were quite useful for our understanding of NLP. We thought that if we had more time, we would have also tried working with the keyword column. Maybe it would have been useful as a feature. Furthermore, if we had a more detailed look into the text tweets, maybe we would have come up with additional features to improve the performance.

BERT model has been trained 3 times in total. However, there are accuracy and loss curves from the first and the last model since the second one was interrupted.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Figure 6 Model Accuracy & Loss on first BERТ

Figure 7 Model Accuracy & Loss on unlemmatized BERT model

From the figures above can be seen that there is a big difference between the accuracy score on the train set, although the validation accuracy remains almost the same. However, the model loss curves show that the validation loss on third (unlemmatized) BERT model is less in comparison with the first model. Therefore, despite that the difference between the accuracy scores is pretty small the loss is lower in the third model.

## 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:**

1. **Final report & presentation**– A final report and presentation are required at the end of the project.

In the last phase of this challenge we focused on preparing the final presentation and wrapping up the report. Thanks to the fact that we were consistent in writing and updating the report as we progressed, it was fairly easy to finish everything in time.

# Results

All our progress and results can be found on our [GitHub repository](https://github.com/krisxtinaa/DC2_DisasterTweets).

* ***BERT***

In the first attempt BERT resulted with a validation accuracy of 80% with 5 trained epochs and a batch size of 32.

A picture containing text

Description automatically generated

Figure 8 First BERT attempt

Text

Description automatically generatedAfter the parameter tuning – increasing the batch size and the epochs, the validation accuracy reached 81%. However, we interrupted the training because we noticed that for few epochs in a row the accuracy did not change. When it reached 81% it was already better than the first attempt.

Figure 9 Second BERT attempt

The results from training a BERT model on unlemmatized text were not any different from the previous achieved results. The end validation accuracy was 0,8109. Obviously, there is almost no change in the results when training BERT on lemmatized text and on unlemmatized. Although, if I must compare the obtained accuracy on the 5th epoch exactly from the second and the third model (with lemmatized and unlemmatized text, and input shape of 128) we believe that the BERT really works better and faster on text which is not lemmatized because for epoch 5 in the second model the accuracy is 0.8096.

Table

Description automatically generated with low confidence

Figure 10 BERT on unlemmatized text

* ***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 figure, 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.

Chart, treemap chart

Description automatically generated

Figure 11 XGBoost on text column without http

Chart, treemap chart

Description automatically generatedThis 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.

Figure 12 XGBoost on text column with http

Regarding the XGBoost on three features the results are not very promising – the accuracy score is the lowest in comparison with the other models almost 65%. With such a small percentage, the confusion matrix was also the least accurate with 800 disaster tweets classified correctly, 286 disasters incorrectly and only 436 non-disasters classified correctly (382 mistaken).

However, the XGBoost on all the features scored better accuracy of 0, 72. Below is the confusion matrix from which can be concluded that the best results remain the first two – XGBoost on the “text” column only and especially the one that has the http string left in the column.

Chart, treemap chart

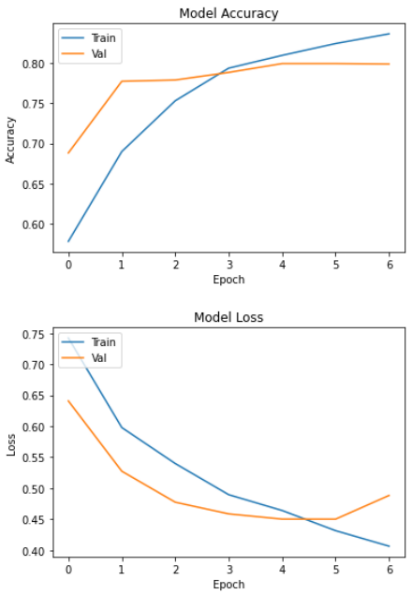
Description automatically generated

Figure 13 XGBoost on all columns

* ***LSTM***

Using LSTM without any features and only with the “text” column altered in the data preparation section, never reached accuracy more than 79%.

Figure 14 LSTM accuracy on text column



We plotted the prediction results on a confusion matrix as well. From the image below it can be observed that the model is actually performing quite well when it comes to the non-disaster tweets classification.

Treemap chart

Description automatically generated

Figure 15 LSTM conf matrix on text column

Treemap chart

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceOur idea was to see if adding more features would end up in giving us better results. After the procedure was done, the model increased its accuracy to 81%, but what was more interesting was the confusion matrix that you can see below.

Figure 16 LSTM conf matrix with features

Figure 17 LSTM accuracy with features

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 – only 241, instead of 330.

# Discussion

As mentioned before BERT is one of the most common models to use for Natural Language Processing. Therefore, when we were ready with the Data Preparation part the first model to run was BERT. This was the first-time using BERT, so we had to do an extensive research. There were reliable and useful tutorials which aimed for preparing the BERT environment. This included the installation of the necessary packages. During the first training session of the model, we took parameters from similar case like ours in order to see how it will perform. The time taken for the whole training was around 2 hours. The scored accuracy was pleasing however, we couldn't put up with the first attempt. The research continued because we wanted to be sure what parameters would be worth to change. Additionally, since we trained only on 5 epoch we assumed if the epochs are increased the model will have a bigger chance to improve its results. However, when we increased the epochs, the batch size, and the input size it was not a surprise that the training time increased as well. Interestingly, at some point the model remained with the same result. For that reason, when the accuracy increased to 0,8102, we stopped the Kernel. We assumed that for the other two epochs the accuracy won`t increase above 81% and because the model was training already for 9 hours. Basically, it took an hour for one epoch.

Since BERT was requiring that long time to train, we continued with the other two models. However, when one of our teachers for the core program shared with us that BERT can handle unlemmatized sentences we decided to give it another try and see the results. The parameters of the second BERT model were used along with the text not being lemmatized. The new model was trained on 5 epochs and run for no more than 2 hours. We believe that the unlemmatized text has something to do with the faster execution. Furthermore, the validation accuracy was a little bit higher than the previous two models as well as the validation loss was lower. Even that small change is an improvement. Therefore, when we take into consideration the faster time to run, the higher accuracy and the lower loss, we firmly believe that BERT works better on unlemmatized text.

Even though XGBoost had nice results from the first attempt, we received feedback about adding some features to the dataset and run them through the models. Since BERT required more time to train while XGBoost and LSTM needed at maximum 2 minutes, we trained the new features on both of them. At the beginning we trained only the two new features together with the already trained one – “text”. To be honest we did not expect such a low accuracy score, thus this was a big surprise for us. As a next step we included all the features in the dataset, hoping for better results. Indeed, the results were better however, not higher than the obtained results from training only on the “text” column. Since, the column was cleaned from all links and numbers, we thought by adding more features showing if the tweet has a link or number in it the model will be more precise. Overall, the best XGBoost model remains the first one considering the higher accuracy score.

LSTM was the other approach we were focusing on, and the first attempt was performed on the “text” column only. In terms of accuracy the results were not the best, but when we plotted the prediction on the confusion matrix we soon realized that it performs rather well. We noticed that specifically for the non-disaster prediction, where XGBoost performed poorly. Therefore, we wanted to go one step further and see if either the accuracy or the prediction would improve. And indeed, the results were better once again. However, it is worth mentioning that XGBoost performed slightly better for classifying disaster tweets – with 1020 of them, while LSTM was with 1003. This small difference, nonetheless, can be “ignored” when the non-disaster classification improved from 488 (in XGBoost) to 551 (in LSTM).

# 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. |
| Second modeling experiments | 12/01/2022 | From Livia – is pleased with our progress which was consistent during the challenge and the new obtained results are good | We have to wrap things up and apply the feedback from the report document |

Table 3 Feedback Log Table

# Conclusion

For this data challenge we ended up experimenting with three approaches for the classification of disaster tweets. Due to the fact that BERT requires a lot of computational power, we only run 3 experiments on it. The two other models used were XGBoost and LSTM, via which we were able to test our ideas most frequently. Thanks to their fast results, the improvement was relatively easy to track and visualizing it through a confusion matrix was efficient. All these models, however, would have not been possible if the pre-processing part was not performed. The tokenization and lemmatization techniques were vital for the preparation of the dataset. Moreover, the exploratory analysis was beneficial for our understanding of the data especially with what values are missing, how both disaster and non-disaster tweets look like, what are the most common keywords, etc.

Each of the three models have their own advantages. BERT is very powerful and with a more in-depth tunning we believe, it would be able to obtain very good results. LSTM and XGBoost are remarkably fast, many experiments can be run, and overview predictions can be shown. Depending on the time and computational consumption we as a group provided solutions which are flexible and can be expanded in the future. Even though we did not achieve impressive accuracy scores (above 90%) for any of the models, we were extremely happy with the obtained results. They served as a proof to us that when we work in an iterative way, the progress can be observed, and comparison can be done.

# References

1. *Natural Language Processing with Disaster Tweets*. (n.d.). Kaggle.com. Retrieved November 21, 2021, from <https://www.kaggle.com/c/nlp-getting-started/overview>
2. *Following the data science methodology.* (n.d.). IBM Developer. Retrieved November 21, 2021, from <https://developer.ibm.com/blogs/following-the-data-science-methodology/>
3. *Step-by-step data mining guide. (2000)*. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. Retrieved November 21, 2021, from [https://the-modeling- agency.com/crisp-dm.pdf](https://the-modeling-agency.com/crisp-dm.pdf)
4. Dizikes, P. (2018, March 8). *Study: On Twitter, false news travels faster than true stories*. MIT News | Massachusetts Institute of Technology. Retrieved November 30, 2021, from <https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308>
5. *Hutchinson, A. (2016, March 18). Here’s Why Twitter is so Important, to Everyone. Social Media Today.* RetrievedNovember 30, 2021, from[*https://www.socialmediatoday.com/social-networks/heres-why-twitter-so-important-everyone*](https://www.socialmediatoday.com/social-networks/heres-why-twitter-so-important-everyone)
6. *XGBoost Simply Explained (With an Example in Python)*. (n.d.). Www.springboard.com. Retrieved December 14, 2021, from <https://www.springboard.com/library/machine-learning-engineering/xgboost-explainer/>
7. *XGBoost*. (2021, September 18). GeeksforGeeks. Retrieved December 14, 2021, from <https://www.geeksforgeeks.org/xgboost/>
8. *What is XGBoost?* (n.d.). NVIDIA Data Science Glossary. Retrieved December 14, 2021, from <https://www.nvidia.com/en-us/glossary/data-science/xgboost/>
9. Wikipedia Contributors. (2018, November 22). *Long short-term memory*. Wikipedia; Wikimedia Foundation. Retrieved December 15, 2021, from <https://en.wikipedia.org/wiki/Long_short-term_memory>
10. *Understanding of LSTM Networks*. (2020, May 10). GeeksforGeeks. Retrieved December 15, 2021, from <https://www.geeksforgeeks.org/understanding-of-lstm-networks/>
11. *Demystifying BERT: The Groundbreaking NLP Framework*. (2019, September 25). Analytics Vidhya. Retrieved December 15, 2021, from [https://www.analyticsvidhya.com/blog/2019/09 /demystifying-bert-groundbreaking-nlp-framework/](https://www.analyticsvidhya.com/blog/2019/09%20/demystifying-bert-groundbreaking-nlp-framework/)
12. *Explanation of BERT Model - NLP*. (2020, April 30). GeeksforGeeks. Retrieved December 15, 2021, from <https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/>
13. Rani Horev. (2018, November 10). *BERT Explained: State of the art language model for NLP*. Medium; Towards Data Science. Retrieved on December 15, 2021, from <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>
14. Hargurjeet. (2021, July 6). *Fake or Not ? Twitter Disaster Tweets*. Geek Culture. Retrieved on December 21, 2021, from <https://medium.com/geekculture/fake-or-not-twitter-disaster-tweets-f1a6b2311be9>

‌