Elevating Disaster Response with Binary Classification of Tweets

Amine Cherif Haouat, Rodrigo Castellano Ontiveros, Filip Zawadka, Robert Praas
KTH Royal Institute of Technology
Drottning Kristinas väg 4
mach2@kth.se, roco@kth.se, zawadka@kth.se, praas@kth.se

Abstract

We investigate the performance of three state-of-the art language models against an LDA baseline for tweet classification. If tweets are about a disaster, the model should give positive output, else not. We find that Transformers are far better in incorporating embeddings than traditional models. The practical relevance of this is yet to be investigated, but it could be useful for early detection of natural disasters through social media. We find that the fine-tuned BERTweet model achieves highest test accuracy and proves to be a potential tool in disaster response.

1 Introduction

Disasters are defined as "a serious disruption of the functioning of a community or a society at any scale due to hazardous events interacting with conditions of exposure, vulnerability and capacity, leading to one or more of the following: human, material, economic and environmental losses and impacts" [1]. Disasters are usually tragic, harmful and require a quick response. At the same time, the increased level of human connection through social media is a potential tool to detect disasters [2] at a potentially higher speed and lower cost than traditional methods [3]. However, channels such as Twitter also contain unrelated tweets or fake news. Tweets can only be trustworthy indicators of disaster if they are checked, preferably automatically. With this goal in mind we compared a baseline probabilistic model to tuned state-of-the-art language models for disaster classification of tweets. The results indicate three levels of success. The traditional baseline (LDA) is frugal but achieves results far less good than humans. The newer, more general and more complex model (BERT) performs already much better. The state-of-the-art model (BERTweet) pretrained on similar data (tweets) as the dataset used, shows signs of (almost) unbeatable performance.

2 Related work

Twitter generates mountains of data and is consequently a popular research area. Related research on disasters and tweets include the case studies on hurricane Irma [4] and Sandy [5]. Moreover, there is a Crisis NLP database of papers and Twitter datasets [6]. A BERT model was used before in this context and achieved a test accuracy of 83.61% as part of a larger set of experiments [7]. Some authors event went a step further by collecting the large Disaster Tweet Corpus 2020 [8]. Unfortunately, this dataset was not available to the authors of this paper.

Latent Dirichlet Allocation (LDA) is landmark contribution to generative probabilistic modeling of text data [9]. Among various applications it is used for online learning [10] and recommending tags for resources [11]. However, newer models emerged the past years. Vector representation of words is known to generally outperform more static models from the past [12]. One landmark contribution was the GloVe (Global Vectors for Word Representation) model [13]. A few years later, BERT followed. These are examples of contextualized word representations, where vector representations differ depending on the context of the word. In previous work, BERT and GloVe were used for

predicting fake news [14] or sarcasm detection [15]. Their popularity and use in similar settings are among the reasons to experiment with these models in particular: preliminary research shows BERT might be able to outperform GloVe for various tasks [16]. The dataset used lends itself perfectly for an extension of BERT, named BERTweet [17].

3 Data

The data was taken from the website 'Data for Everyone' [18]. It was originally shared by the company figure-eight. The dataset we have chosen for this project is composed of different tweets, sequences of text collected from the social media Twitter. The tweets are labelled as natural disaster or non-natural disaster and the total amount of labelled tweets is 7613. The tweets are in their raw state, meaning that this was exactly the way they were written on Twitter. This means that spelling errors can occur, as well as an unpredictable use of cap letters. 4342 Tweets were non-disaster related, and 3271 tweets were disaster related, showing a slight imbalance. Different pre-processing methods were necessary for the different methods, but there are common steps to all the methods that make the tweets more usable and containing less useless information, such as removing usernames, links and dates from the tweets. Then, each method has its specifics:

- The baseline method used "bag of words", meaning that only the relevant words were kept and the position of the words is irrelevant.
- The state-of-the art methods seek the general meaning of a tweet and the semantic correlation between the words that form the tweet. The preprocessing is therefore different. For BERT and BERTweet respectively, their tokenizers were taken from the Hugging Face [19] library.

4 Methods

Our objectives include reliable classification of whether tweets relate to disasters, understanding the difference in performance between an older (2003) baseline model and newer (around 2018) transformer models. Our approach is comparing state-of the art models to a traditional baseline.

4.1 Baseline : Latent Dirichlet Allocation + Multi-Layer Perceptron

For the baseline, we decided to go with a model divided into 2 steps. First, a method that is able to effectively quantize tweets depending on their content. Then, a method that is able to link those quantities to whether the tweet is disaster related or not.

4.1.1 Topic modelling

The first part of the model is about topic modelling, because it is an efficient method to group documents based on their semantic similarity. The topic modelling method we decided to implement was the **Latent Dirichlet Allocation (LDA)** [20]. This method relies on "meaningful" words, meaning that each tweet has to be cleaned prior to using this method. After the common preprocesing tackled in the Data section, the process is the following:

- The tweets are tokenized, meaning that each tweet is turned into a succession of words, removing therefore any type of punctuation.
- Then, the words are lemmatized, meaning that multiple words that come from the same origin are transformed into the same word. Example: "seeing", "seen", "seeable" are all turned into "see".
- Lastly, stopwords are removed. Stopwords are words that are too general and therefore don't contribute to determining the meaning of the text. Example : "be", "it", "as", "therefore".

LDA is a statistical topic modelling method that considers:
- That each topic is described by a weighted sum of words.
- Each document as a bag of words. Multiple meanings of one word are therefore difficult to grasp.
- Each word as an individual entity for training only. Therefore, if the word "flood" is written twice in the tweet, then for the training part, they will be treated as 2 different entities.

The method relies on the assumption that each document is described by a Dirichlet distribution of topics of parameter α, and each topic is described by a Dirichlet distribution of words

of parameter β. With the number of topics K, they form the hyperparameters. Figure 1 describes the general training process of the model. After the training is done, each topic is described by a weighted sum of words, which helps determining the probability of a document belonging to a given topic, due to the words that constitute the document. The quality of the topic modelling is assessed by the coherence score.

4.1.2 Multi-Layer Perceptron

These probabilities are the new way to represent each and every tweet. At that point, each tweet is represented by as many values as there are topics. This new representation is the input to a Multi-Layer Perceptron, of dimension K (number of topics). The output layer is composed of 2 nodes, the first determining the probability of the tweet not being disaster-related, and the second determining the probability of it being disaster-related.

4.2 State-of-the-art models

In our search to state-of-the-art classification of disaster tweets, the embeddings of GloVe (With LSTM classification layer), BERT and BERTweet are compared.

4.2.1 GloVe embedding + LSTM

Glove[13] is a well established method of representing words as vectors interpretable by machine learning models. Due to the fact that placement of the word and contexts within the sentence is detrimental to extracting the true meaning, we decided that a single bidirectional LSTM layer network has a potential to solve this task.

4.2.2 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. Bert is a bidirectional transformer pretrained using both masked language modeling objective and next sentence prediction [21]. BERT large is used in this project, with a total of 24 transformer blocks, 1024 as the hidden size, 16 self-attention heads and a total of 340 M parameters.The corpus on which it is pretrained is Toronto Book Corpus and Wikipedia. It is a state-of-the-art model that can be used for different purposes and tasks. In this case, we have used BERT as a transformer with sequence classification head on top.

4.2.3 BERTweet

Whereas existing language models are pretained on structured and longer texts, tweets contain much shorter amounts of text with irregular language use such as abbreviations, hashtags and informal grammar. [17]. Therefore, BERTweet is a pre-trained language model for English Tweets. It is trained similarly to the previous BERT model, using the RoBERTa procedure [22]. It was found to produce better performance than state-of-the-art models for text classification [17]. Its pre-training consisted of 80GB of data containing 850M English Tweets. For the implementation the Simpletransformers library [23] is used. Through this library the best-performing HuggingFace model [19] is called, which includes the BERTweet Tokenizer.

5 Experiments

5.1 Baseline : Latent Dirichlet Allocation + Multi-Layer Perceptron

5.1.1 LDA

The difficulty of fine-tuning resides in the randomness in the training process, meaning that there is no continuity of the coherence function. Random search for α, β and K is the way we went in order to determine the optimal parameters. The best coherence score obtained was of around 0.50, with $\alpha = 0.5$, $\beta = 0.5$ and K = 4. The following topics were obtained (the numbers prior to the words represent the weights, and only the 10 words with the highest weights are printed):

Topic 0: 0.024"fire" + 0.010*"news" + 0.008*"wreck" + 0.008*"train" + 0.008*"wreckage" + 0.008*"malaysia" + 0.008*"confirmed" + 0.008*"video" + 0.007*"wild" + 0.007*"home"*

Topic 1: 0.013"get" + 0.009*"wa" + 0.009*"day" + 0.008*"storm" + 0.006*"ha" + 0.006*"time" + 0.006*"people" + 0.005*"wrecked" + 0.005*"going" + 0.005*"love"*

Topic 2: 0.010"amp" + 0.007*"old" + 0.007*"look" + 0.007*"wa" + 0.007*"see" + 0.006*"bomb" + 0.005*"ha" + 0.005*"way" + 0.005*"woman" + 0.005*"best"*

Topic 3: 0.012"family" + 0.012*"disaster" + 0.010*"police" + 0.010*"wildfire" + 0.010*"wounded" + 0.009*"emergency" + 0.008*"nuclear" + 0.007*"officer" + 0.007*"dead" + 0.006*"suspect"*

Some words do not make sense, due to tweets naturally containing odd words. Topics that are disaster related or non-disaster related cannot be clearly distinguished, but the difference of weights is the key to having a relevant representation of each tweet to get through the MLP.

However, one can notice that each topic seems to tackle one disaster in particular, meaning that **Topic Modelling can indeed be used in order to group the tweets by type of disasters**. However, since this was not the main focus of the project, we did not go further in creating the very optimal topics for classification within the "disaster related" class, but the first results with 4 topics are really promising.

5.1.2 MLP

Then comes the model selection of the MLP. Input and output size were fixed, since the input size only depends on the number of topics chosen for LDA, and the output size is 2 because of the 2 classes. Different architectures were tested (layers width, number of layers, for instance) andwee obtained results after a training of 300 epochs. Figure 6 in the Appendix shows wider and deeper networks have room for more learning, but learning will be subjected to overfitting and will not improve the validation nor the test performances, as can be seen on Table 1.

Table 1: Test performances for the baseline model

Architecure	Test loss	Test accuracy
(4, 100, 2)	0.588	69,55%
(4, 50, 50, 2)	0.590	69,29%
(4, 100, 100, 2)	0.584	69,69%
(4, 200, 200, 2)	0.583	69,55%

The **best test performance achieved (when averaged across numerous simulations) is about 70 %**, which is set to be the baseline for classification.

5.2 Advanced methods - Visualization

We obtained the first n-grams in order to see what words were the most common. Before visualizing which n-grams were the most common, we decided to clean the text by removing websites, misspelled words and so forth. The first two n-grams can be seen in Figures 2, 3 in the Appendix. In Figure 3 there are some 2-grams that are expected to be in the disaster tweets, such as suicide bomber or suicide bomb, whereas for the most common words we have 2-grams such as reddit quarantine, which does not fit the context of natural disaster. Nevertheless, some non-disaster 2-grams as “burn building” may be confusing.

As part of the results, we decided to visualize the data on different embeddings. The embeddings used ranged from CountVectorizer and TfidfVectorizer to GloVe and BERT embeddings. In order to be plotted, different dimensionality reduction techniques were applied, such as Truncated SVD and UMAP. The results were not satisfactory in general, in the sense that it was not visually possible to clearly distinguish the classes. One of the most successful cases is shown in Figure 4 in the Appendix.

5.3 GloVe embeddings + LSTM

We trained a single bidirectional LSTM layer with different numbers of units with GloVe embeddings as input. What we noticed is that in most of the trainings around epoch 7-9 models started overfitting as can be seen in 5. A model with only 50 units performed the best.

Table 2: Results of the different number of units with GLoVe-LSTM.

LSTM units	Test accuracy
50	82.4%
100	81.4%
200	80.1%
512	79.3%

5.4 BERT

Another model that we decided to apply is BERT model for sequence classification, from [19]. We used the tokenizer from [19] and then fine-tuned the model for 3 epochs. Different configurations were applied by changing the optimizer (Adafactor and AdamW) and different learning rates. The rest of the parameters were the default from the model, and the learning rate was a linear schedule with warmup. The best results are shown in table 3.

Table 3: Results of the different trainings with BERT.

Optimizer, lr	Test accuracy
AdamW, 6e-6	84.53%
Adafactor, 1e-5	83.46%
Adafactor, 6e-6	84.85%

The best result was obtained for the BERT model with Adafactor as optimizer and learning rate $6\mathrm{e}{-6}$, leading to an accuracy of 84.85% in the test set.

5.5 BERTweet

As an extension to the BERT classifier, we perform experiments with the BERTweet algorithm. Given that our BERT model with 84.85% scored better than the baseline, we do small experiments on batch_size, learning rate and epochs to look for improvements. Note that all learning rates decrease over time with the same standard scheduler. Table 5 shows the results for varying the starting learning rate (lr) for standard settings of 2 epochs, batch size of 60 and training with early stopping. The results are clearly sensitive to the learning rate, too high (4e-4) or too low (1e-5) hurt the test accuracy, although a too high learning rate penalizes the test accuracy much harder.

After the learning rate, we investigate the batch size in Table 6. Here the starting learning rate is 4e-5 and 2 epochs are used. The effect of the batch size is small, going lower than 8 or 12 as a batch size leads to a slightly smaller test accuracy. Possibly even more importantly, training with a smaller batch size is much faster and increasing the batch size from 4 to 8 was not too costly in terms of time.

As a final experiment the amount of epochs is tweaked for the best model so far. This means a learning rate of 4e-5, batch size of 12 and a varying amount of epochs. Clearly, more epochs lead to higher test accuracy. Even though the indicator is test accuracy, this might still be a sign of overfitting. Disasters are typically characterized by words such as “danger, fire, earthquake, evacuate”. BERTweet might just get good at recognizing such type of words.

Table 4: Results of the different training with BERTweet.

Epochs	Test accuracy
1	88.1%
2	91.0%
3	93.6%
4	95.3%
5	96.6%
6	97.6%
7	97.8%

6 Discussion

At first glance, it can seem odd that all architectures from the baseline, no matter how complex (to a certain extent), provide practically the same results. This means that the model works well with non-ambiguous cases. However, the number of false negatives is almost 3 times as high as the number of false positives, and that is due to the construction of the dataset: it contains a lot of ambiguous cases (the word "ablaze" is almost never disaster related for instance), so much that the model ends up thinking that it is always tricked. To counter this effect, the model therefore throws most of the ambiguous but actual real disaster cases in the non-disaster tweets class. This should be confirmed by a more thorough analysis of the LDA part of the baseline, which was not the main concern since this only is the baseline.

Due to the comparison and implementation of different models, we have learned in this project what models suit better this dataset and therefore potentially also other datasets similar to this one. We have seen how transformers give the best result in any case and outperform the other models. Besides, we have also understood what possibilities are used nowadays to obtain data embeddings, in the case of GloVe or BERT, for instance. It is clear that a model that is modern and trained on similar data (BERTweet) performs better than other models.

There is further work that can be done, as collecting a wider dataset or doing data augmentation (for instance, a possibility could be to use tools or models to paraphrase). This could lead to better results in most of the models. Furthermore, a wider fine-tuning can be applied and probably lead to improvements in the performance. Another possibility is to perform some sort of semi-supervised or unsupervised learning in this problem. There could be several new applications to different datasets, and use it not only for disaster prevention, but also, for example, to classify which tweets come from reliable sources and based on that give recommendations to different users on which tweets are reliable or not. Another use could be detecting offensive tweets and try to ban accounts which often use social media to offend or detecting bots.

7 Conclusion

The best test accuracy obtained for the baseline was in average 70% for the LDA/MLP method. However, a method only based on seeing tweets as bags of words represents the most crucial subtlety in natural language processing: words with multiple meanings such as "ablaze" and "floaded", and the upmost importance of context.

BERT helps overcoming this issue, because training actually takes into consideration the general meaning of the whole tweet. Thanks to fine-tuning BERT, we obtained a 84.85% of accuracy in the test set. This means that the BERT model outperformed the MLP by more than 15% accuracy, which we can consider as successful.

However, BERTweet being trained on Twitter data achieved 93.6% test accuracy for similar settings in comparison to the BERT model that achieved 84.85%. Therefore, this leads to the conclusion that using the BERTweet tokenizer and later BERTweet model is the best alternative we have found throughout this project. The state-of-the-art is changing the game and it seems more relevant than ever to be aware of recently developed models, when the objective is similar to maximizing test accuracy. The code used in this project is available at https://github.com/rodrigo-castellano/Disaster_tweets

References

[1] *Disaster | UNDRR*. URL: https://www.undrr.org/terminology/disaster.

[2] Huiji Gao, Geoffrey Barbier, and Rebecca Goolsby. "Harnessing the crowdsourcing power of social media for disaster relief". In: *IEEE Intelligent Systems* 26.3 (2011), pp. 10–14.

[3] Himanshu Shekhar and Shankar Gangisetty. "Disaster Analysis Through Tweets". In: Aug. 2015. DOI: 10.1109/ICACCI.2015.7275861.

[4] Muhammed Ali Sit, Caglar Koylu, and Ibrahim Demir. "Identifying disaster-related tweets and their semantic, spatial and temporal context using deep learning, natural language processing and spatial analysis: a case study of Hurricane Irma". In: *International Journal of Digital Earth* 12.11 (2019), pp. 1205–1229. DOI: 10.1080/17538947.2018.1563219. eprint: https://doi.org/10.1080/17538947.2018.1563219. URL: https://doi.org/10.1080/17538947.2018.1563219.

[5] Kevin Stowe et al. "Identifying and categorizing disaster-related tweets". In: *Proceedings of The fourth international workshop on natural language processing for social media*. 2016, pp. 1–6.

[6] *CrisisNLP*. URL: https://crisisnlp.qcri.org.

[7] Santiago González-Carvajal and Eduardo C. Garrido-Merchán. "Comparing BERT against traditional machine learning text classification". In: *CoRR* abs/2005.13012 (2020). arXiv: 2005.13012. URL: https://arxiv.org/abs/2005.13012.

[8] Matti Wiegmann et al. "Analysis of detection models for disaster-related tweets". In: *Analysis of Detection Models for Disaster-Related Tweets* (2020), pp. 872–880.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[10] Matthew Hoffman, Francis Bach, and David Blei. "Online learning for latent dirichlet allocation". In: *advances in neural information processing systems* 23 (2010).

[11] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. "Latent dirichlet allocation for tag recommendation". In: *Proceedings of the third ACM conference on Recommender systems*. 2009, pp. 61–68.

[12] Kawin Ethayarajh. "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings". In: *arXiv preprint arXiv:1909.00512* (2019).

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *EMNLP*. 2014.

[14] Azka Kishwar and Adeel Zafar. "Predicting Fake News using GloVe and BERT Embeddings". In: *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2021, pp. 1–6. DOI: 10.1109/SEEDA-CECNSM53056.2021.9566243.

[15] Akshay Khatri et al. "Sarcasm detection in tweets with BERT and GloVe embeddings". In: *arXiv preprint arXiv:2006.11512* (2020).

[16] Santiago González-Carvajal and Eduardo C Garrido-Merchán. "Comparing BERT against traditional machine learning text classification". In: *arXiv preprint arXiv:2005.13012* (2020).

[17] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. "BERTweet: A pre-trained language model for English Tweets". In: *arXiv preprint arXiv:2005.10200* (2020).

[18] *Source of the dataset*. URL: https://appen.com/datasets-resource-center/.

[19] Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[20] Arun K Sharma. "Understanding Latent Dirichlet Allocation (LDA)". In: (2020). URL: https://www.mygreatlearning.com/blog/understanding-latent-dirichlet-allocation/.

[21] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[23] *ThilinaRajapakse/simpletransformers: Transformers for Classification - GitHub*. URL: https://github.com/ThilinaRajapakse/simpletransformers.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See section Introduction and Conclusion.
   (b) Did you describe the limitations of your work? [Yes] See section Conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] The work is very objective, as it only aims at distinguishing disaster-related and non disaster-related tweets.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [N/A]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See beginning of each method description.
   (b) Did you include complete proofs of all theoretical results? [No] Because we aim at giving an overview of the methods. References are provided to go deeper into the theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See section Conclusion, a github link is provided, containing multiple notebook about each method, each notebook providing the different steps to run in order to reproduce the results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Our main focus was to average the results. However, since the results were close, there was no need to perfectly reproduce the results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Since the work was split, it was impossible to keep track of the computing time and the different methods each member used to run his code.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In the beginning or the end of each notebook.
   (b) Did you mention the license of the assets? [Yes] No license was needed to be mentioned.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Data was public since it was available for a Kaggle competition.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Content was pre-checked by the Kaggle competition creators. Personally Identifiable Information is not an issue since those tweets were public, so the users knew that what they published is traceable back to them.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

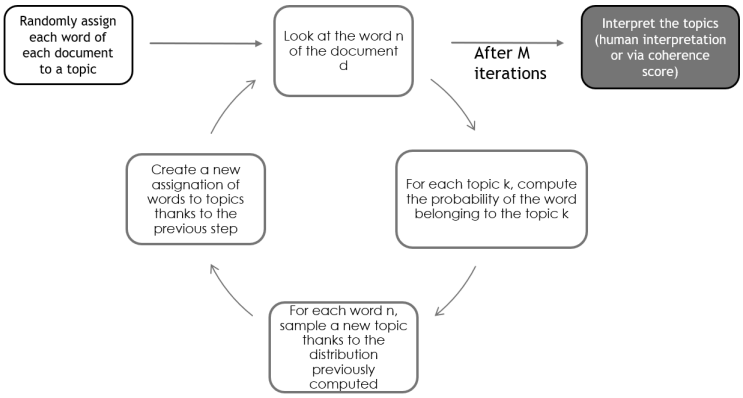

Figure 1: LDA training process

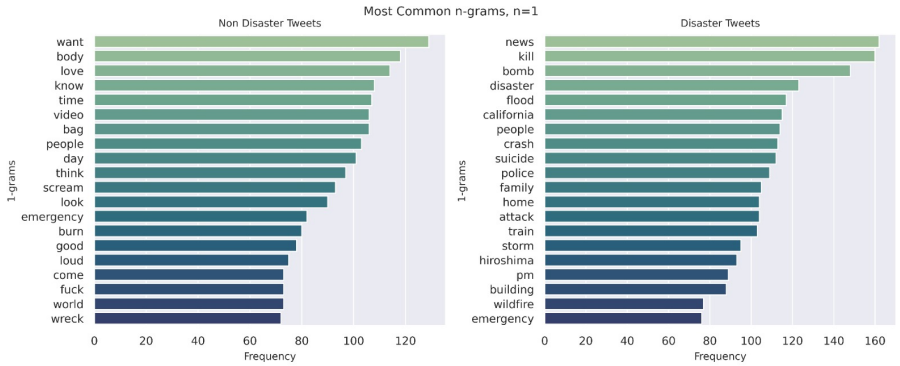

Figure 2: Most common words in the dataset.

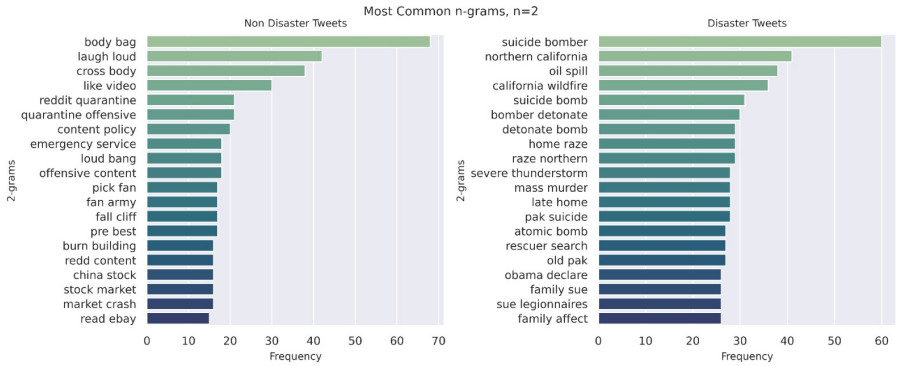

Figure 3: Most common 2-grams in the dataset.

Figure 4: Visualization of the embedding obtained by applying CountVectorizer after performing dimensionality reduction with Truncated SVD.

Figure 5: LSTM size 100 accuracy and loss over time

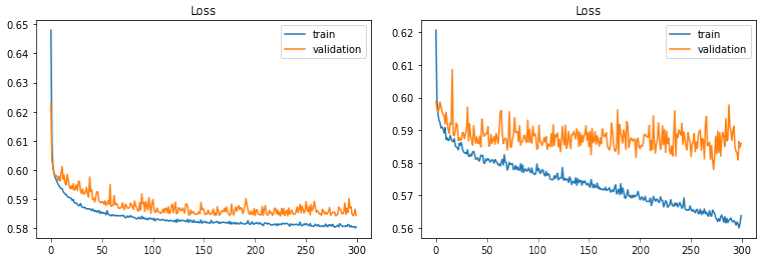

Figure 6: Training and validation loss for architectures (4, 100, 2) and (4, 200, 200, 2) (the less complex and most complex architectures tried)

Table 5: Results of the different trainings with BERTweet.

lr	Test accuracy
4e-4	57%
4e-5	90.2%
2e-5	88.4%
1e-5	86.4 %

Table 6: Results of the different trainings with BERTweet.

Batch size	Test accuracy
4	90.4%
8	91.0%
12	91.0%
60	90.2%