CS 263 Midterm Project Report
Classifying Disaster Tweets

Niklas Z.
niklasz@ucla.edu

In this project we use modern architectures to classify whether a post on Twitter is discussing a real disaster or not. We approached the question by training 1 BERT and 2 RoBERTa models. To improve their accuracy, we ran further experiments on pre-processing and data augmentation as well.

1 Methods

Prior to model selection, we conducted a brief analysis of the data [3] provided. There were 7,613 rows, with columns `id`, `keyword`, `location`, `text`, `target` respectively. `target` indicates the class, where roughly 57% are in class 0 (not disaster) and 43% are in class 1 (is disaster). `text` contains a user tweet ranging from 7-157 characters and acts as the main feature to use for classification. The optional fields `location` and `keyword` contain an unformatted location and a disaster-relevant keyword respectively.

We initially considered incorporating the optional fields into a model but found that they were not particularly valuable. `location` is only populated in 67% of the rows and has an inconsistent scope, sometimes only referencing a country, state, or city and in other cases, completely non-indicative information. `keyword` is populated in 99% of the data and there are 218 unique keywords for class 0 and 220 unique keywords for class 1. Unfortunately, 217 of these keywords are shared, meaning it is not possible to reliably separate classes by `keyword`. With this in mind, we opted for approaches that could rely solely on the `text` and could be trained on low amounts of examples.

1.1 BERT & RoBERTa

The Bidirectional encoder Representations from Transformers (BERT)[6] is a neural network that bases off of the encoder in the transformer architecture[11]. It is trained on multiple tasks using very large datasets, where the final classification layer can then be removed to use the generated encodings for downstream tasks. The Robustly Optimized BERT Pretraining Approach (RoBERTa)[8] is an enhancement of BERT which optimises hyper-parameters such as batch size, and learning rate and manages to achieve better performance on the same tasks.

For this project, we used the following existing implementations of these models:

- *bert-base*[1] - BERT model trained using wikipedia articles[7] and novels[12].
- *roberta-base*[4] - RoBERTa model trained with the same dataset as *bert-base*.
- *roberta-twitter*[2] - RoBERTa model trained using 128 million tweets[9].

These were chosen so we could determine how different architectures trained on general data compare to one pre-trained on task-specific data (tweets).

1.2 Pre-processing & Data Augmentation

During the initial analysis of the data, we also identified a few issues with the `text` field, which could negatively affect the tokenization and subsequent learning:

- **user tags** - 27% of tweets contain at least 1 user tag, most of which are unique. These names are too rare and will add unnecessary noise.

- **URLs** - 52% contain URLs that have been passed through a URL shortener, meaning there are no relevant human words to extract.
- **hashtags** - 23% contain the # symbol, which is not commonly used elsewhere.
- **web-scraping artefacts** - 15% contain unreadable web-scraping symbols.

To homogenise the data these were either replaced or removed, with examples in Appendix C. Additionally, as the amount of training data is quite low we augmented it via back translation using a pair of English→German and German→English models [10](see Appendix D).

2 Results

We split the data into 90% training and 10% validation. We then trained each model for 3 epochs (more hyper-parameters in Appendix E), while applying each pre-processing step individually. The results are visible in Table 2, where we can observe that all 3 models perform quite similarly with no changes (80.4%-84.3%) with the *roberta-twitter* having a slight edge. Of the pre-processing steps the "lowercasing" and "invalid symbol cleanup" have consistent positive effects an all models.

The data augmentation was applied separately from the pre-processing and is shown in Table 2. Here the results vary by model: *bert-base* remains largely unaffected, whereas *roberta-base* and *roberta-twitter* do improve in accuracy as the augmentation rate increases.

Pre-Processing Step	***bert-base***	***roberta-base***	***roberta-twitter***
baseline	83.6%	80.4%	**84.3%**
user replacement	**83.3%**	82.0%	83.1%
user removal	83.8%	82.6%	**84.0%**
URL replacement	83.3%	82.6%	**84.1%**
URL removal	82.1%	82.2%	**84.7%**
hashtag removal	83.2%	81.2%	**85.6%**
lowercasing	**85.5%**	82.0%	84.6%
invalid symbol cleanup	84.0%	84.5%	**84.6%**

Table 1: Classification accuracy when using different pre-processing steps.

Augmentation Rate	***bert-base***	***roberta-base***	***roberta-twitter***
baseline	83.6%	80.4%	**84.3%**
10%	83.3%	83.3%	**84.5%**
20%	83.1%	82.8%	**84.9%**
50%	83.7%	84.0%	**85.0%**
80%	83.5%	84.4%	**85.3%**

Table 2: Classification accuracy when using data augmentation. Note that 100% was not possible as not every tweet could be back-translated into a different text.

3 Evaluation

From this experiment, we can see that these fine-tunable transformer models perform very strongly from the outset, even when their pre-training data differ considerably. The gains from pre-processing and augmentation are fairly modest, although this is also because we apply them independently. Of course, applying all of the most promising methods together does not guarantee a better model as they do affect each other. However, studying their combined effect requires considerably more computational resources.

References

[1] Bert-base-cased · hugging face. https://huggingface.co/bert-base-cased.

[2] Cardiffnlp/twitter-roberta-base-mar2022 · hugging face. https://huggingface.co/cardiffnlp/twitter-roberta-base-mar2022.

[3] Natural language processing with disaster tweets. https://www.kaggle.com/competitions/nlp-getting-started/data.

[4] Roberta-base · hugging face. https://huggingface.co/roberta-base.

[5] Training parameters - hugging face. https://huggingface.co/docs/transformers/v4.19.0/en/main_classes/trainer#transformers.TrainingArguments.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] W. Foundation. Wikimedia downloads.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[9] D. Loureiro, F. Barbieri, L. Neves, L. E. Anke, and J. Camacho-Collados. Timelms: Diachronic language models from twitter. *CoRR*, abs/2202.03829, 2022.

[10] N. Ng, K. Yee, A. Baevski, M. Ott, M. Auli, and S. Edunov. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*, 2020.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

4 Appendix

A Project Code

The project code is available for download via the following URL:

`https://drive.google.com/file/d/1TnuJ2r9H91XrNY90EuEIZH34qnoW66qE/view?usp=sharing`

B Data Examples

id	keyword	location	text	target
290	ambulance	None	What's the police or ambulance number in Lesotho? Any body know?	0
287	ambulance	USA	Twelve feared killed in Pakistani air ambulance helicopter crash http://t.co/TH9YwBbeet #worldNews	1
6532	injuries	Riverview, FL	4 Common Running Injuries and How to Avoid Them http://t.co/E5cNS6ufPA	0
6534	injuries	California	Enjoying a little golf this summer? Take care to avoid injury – back and shoulder injuries can happen quickly http://t.co/f1R5ISBVks	1

Table 3: Examples of the training data. Note that the labelling is a bit odd. The `id=6532` `id=6534` are quite similar, but are classified differently.

C Pre-processing Examples

This section contains an example for each type of pre-processing step. Most of these are self-explanatory, but we will list all for completeness.

state	text
before	@JulieChen she shouldn't. Being with them is gonna ruin her game and Vanessa is a great player
after	@user she shouldn't. Being with them is gonna ruin her game and Vanessa is a great player

Table 4: Example of the *user replacement* operation.

state	text
before	@JulieChen she shouldn't. Being with them is gonna ruin her game and Vanessa is a great player
after	she shouldn't. Being with them is gonna ruin her game and Vanessa is a great player

Table 5: Example of the *user removal* operation.

state	text
before	Oil and Gas Exploration http://t.co/PckF0nl2yN
after	Oil and Gas Exploration http

Table 6: Example of the *URL replacement* operation.

state	text
before	Oil and Gas Exploration http://t.co/PckF0nl2yN
after	Oil and Gas Exploration

Table 7: Example of the *URL removal* operation.

state	text
before	Dr Jack Stern Interview Ending Back Pain for #Military #Injury. Listen now: http://t.co/YhH7X0MAio
after	Dr Jack Stern Interview Ending Back Pain for Military Injury. Listen now: http://t.co/YhH7X0MAio

Table 8: Example of the *hashtag removal* operation.

state	text
before	Reality Training: Train falls off elevated tracks during windstorm http://t.co/JIOMnrCygT #Paramedic #EMS
after	reality training: train falls off elevated tracks during windstorm http://t.co/jiomnrcygt #paramedic #ems

Table 9: Example of the *lowercase* operation.

state	text
before	@IcyMagistrate \x89ÛÓher upper arm\x89ÛÒ those /friggin/ icicle projectiles\x89ÛÒ and leg from various other wounds the girl looks like a miniature more\x89ÛÓ
after	@IcyMagistrate her upper arm those /friggin/ icicle projectiles and leg from various other wounds the girl looks like a miniature more

Table 10: Example of the *invalid symbol cleanup* operation. This operation includes a range of replacement and removal patterns. The full list can be viewed in the accompanying code.

D Augmentation Examples

version	id	text	target
original	6343	Holmgren describing 96 World Cup: we were Lou's hostages	0
translated	106343	Holmgren on 96 World Championships: We were Lou's hostages	0
original	478	@Erker Again?? Eep! Thought of you yesterday when I saw that hella scary hail. #armageddon?	1
translated	100478	@ bay window Again?? Eep! Think of you yesterday when I saw that hella scary hail. # armageddon?	1

Table 11: Examples of the data augmentation via back-translation. This approach allows for more contextual variation than for example synonym replacement. It does however also introduce mistranslations that add noise to the training data.

E Training Hyper-parameters

We generally used fixed random seed for training in order to allow for reproducibility (see code). Aside from this, we predominantly use the `huggingface` library's default hyper-parameters[5].

Hyper-parameter	**Setting**	**Notes**
Training epochs	3	This is the same for all training runs. However, for augmentation the actual number of training steps does increase with the size of the data.
Learning rate	1^{-5}	Hugging face default.
Learning rate decay	linearly to 0	Hugging face default.
Optimizer	AdamW with weight decay 0 $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 1^{-8}$	This is a modified variant of the adam optimizer.
Batch Size	8	Mostly dependended on the available computing resources.
Model Hyper-parameters	N/A	These are set during pre-training. See respective URLs and papers in the references section.