Real or Not ? NLP with disaster tweets

BOUSBIB Ruben
CentraleSupelec
rbousbib@gmail.com

XAVIER Cédric
CentraleSupelec
cedric.xavier@student-cs.fr

Abstract

Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter (i.e. disaster relief organizations and news agencies). But, it's not always clear whether a person's words are actually announcing a disaster. In this work, we use recent advances in deep learning for Natural Language Processing to learn a classification model of our tweets. In particular, we assess the performance of the BERT model which uses attention mechanisms via the Transformer architecture, now used in almost all state of the art NLP models.

1. Introduction

Natural Language Processing is one of the areas that has benefited the most from the new Deep Learning methods due to the very large amount of data that is easily accessible on the Internet (e.g. Wikipedia database [1]). Many industrial applications of these algorithms are nowadays implemented on platforms that need to efficiently process this data and one of the main tasks to be carried out is classification. Whether it is to predict whether a comment is positive or negative (sentiment analysis [2]), an email is spam or not or a tweet is fake news, these are deep learning models that are used behind the scenes. Here we look at the case of fake news for disaster tweets.

The problem is driven by the fact that social networks have become the main source of information because of their ease of access and the rapid spread of information worldwide. However, these characteristics have also made it very difficult to assess the reliability of the information they contain. Being able to tell whether a tweet is true requires much more advanced language analysis tools than before. For example, the meaning of a tweet could not be inferred from the meaning of its key words alone: the sentence "This car is the bomb !", although containing the word "bomb", has nothing to do with an attack alert and to understand this, a sophisticated language model is needed.

In this work we use a transfer learning technique to take advantage of the robust features learned by BERT, a deep neural network already trained on another big dataset (Wikipedia). By adding only one feed-forward layer to this network, we obtain very good performancse on our classification task, which shows the good generalization abilities of BERT for specific tasks where training data is relatively limited.

2. Problem definition

Our problem can be addressed as a binary classification task where the input is the sequence of handcrafted or learned feature vectors $\mathbf{x} = \{x_1, x_2, ..., x_T\}$ representing a tweet, and the expected output is the corresponding label $\mathbf{y} = 1$ if the tweet is about a real disaster and $\mathbf{y} = 0$ otherwise. Feature vectors $x_1, ..., x_T$ are referred as *tokens* as we will see in the following.

We learn a classification model from a labeled dataset $\{(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), ..., (\mathbf{x_M}, \mathbf{y_M})\}$ provided by the Kaggle competition *"Natural Language Processing with Disaster Tweets"*. We devide this dataset into a training and validation set.

At training time, sequences of tokens $\{\mathbf{x_i}\}_{i=1}^{N}$ are drawn randomly from the training set to form mini-batches of size N. The model is trained to minimize the cross-entropy loss, using standard gradient back-propagation to update the weights θ of the network f_θ :

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y_i} \log f_\theta(\mathbf{x_i}) + (1 - \mathbf{y_i}) \log(1 - f_\theta(\mathbf{x_i}))$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta)$$

At validation time, we evaluate the performances of the model on unseen data to assess its ability to generalize. We use various metrics such as accuracy and F1-score to fine-tune hyperparameters such as the learning rate of the optimization algorithm, or the capacity of the model.

At test time, we compute the predictions of the model f_{θ^*} on the test set and choose the test accuracy as our final score to optimize.

3. Related work

Text classification is a basic problem in the field of natural language processing and different machine learning approaches have been studied successively throughout history. From the 1960s to the 2010s, based on shallow models such as Support Vector Machines (SVM) [5] or Nearest Neighbor (KNN) [6], text classification was still tackled with handcrafted feature engineering to represent sentences in a mathematical space. Event if these methods were still more robust than rule-based methods, this feature engineering process was time-consuming and expensive. Then, when deep learning began to develop with the emergence of very large datasets, text classification gradually changed for these data-driven approaches and started to include feature extraction into the learning process.

In shallow learning models, the first step is to preprocess the raw input text into a simpler representation, easier to classify. Many text representation approaches have been used in the literature such as Bag-of-Words (BOW) [3], term frequency-inverse document frequency (TF-IDF) [4] or word2vec [5]. BOW considers sentences as "bags" containing words and, according to a vocabulary, it maps each word to its frequency within the bag. TF-IDF uses the word frequency within a text and the inverse proportion of texts including this word in the dataset to give greater weight to the less frequent terms that are considered to be more discriminatory. Word2vec is a word-embedding method that takes into account the context of a token within the sentence. It is an unsupervised algorithm which predicts a target word from its context. The learnt embedding matrix gives a new representation of our tokens, useful for further classification.

In deep learning models, neural networks automatically learn high-level features from data, getting better results than shallow learning models. The basic deep learning architectures for text classfication are Feed-Forward neural networks, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and attention mechanisms. Most of the research consists of improving or merging these basic architectures to improve performances of text classification algorithms on benchmark datasets such as AG's News, a collection of more than 1 million news articles for topic classification.

Recently, the development of the BERT architecture (Bidirectional Encoder Representations from Transformers) [6] has led to a significant breakthrough in the field of NLP. It makes use of Transformer [7], an attention model that uses contextual relations between words to learn useful and robust representations that can then be re-used for downstream tasks. BERT takes a different approach than previous models in NLP, it considers all the words of the input sentence simultaneously, unlike RNNs, and then uses an attention mechanism to develop a contextual meaning of the words. Then, unlike Word2Vec which outputs just one vector (embedding) for each word, combining all the different senses of the word into one vector, BERT provides a context-dependant embedding.

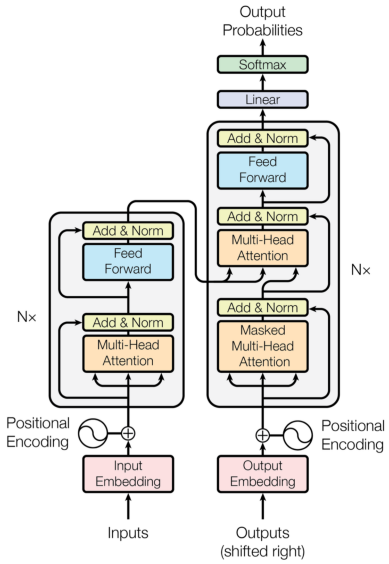

Figure 1. Transformer model architecture. The encoder is on the left and is the only part that is used in BERT. It consists of stacking a multi-head attention layer which computes the attention score for each token with respect to the other surrounding it and a feed forward layer.

In the following, we show how we can benefit from a BERT model pretrained on another dataset than the one that interests us to perform well on our disaster tweets classification task.

4. Methodology

As it has been in use for some years now in computer vision with for example Resnet pretrained on ImageNet, we make use of having easy access to models trained on large amounts of data to obtain a pre-trained model which provides a useful and robust representation of our data samples from the outset. This is a *transfer learning* technique where we use the knowledge of a model trained on a similar task than the one we are interesting, allowing us to not starting from a randomly initialized model.

The architecture of BERT is derived from Transformers which is an attention encoder-decoder model. Since we only need a model that computes features for our specific tweet classification tasks, we only use the encoder part of the Transformer (left side of Figure 1).

4.1. Tokenization

Before feeding our Transformer a sequence of vectors, we need to know how to divide a sentence into word instances called *tokens*. Starting from a raw text such as *"I'm on top of the hill and I can see a fire in the woods."* a first approach could be to form a token according to spaces into the sentence. However one problem of such a naive approach would be a too large resulting vocabulary forcing the model to have an enormous embedding matrix as the input layer, which causes both an increased memory and time complexity. In general, transformers models rarely have a vocabulary size greater than 50,000.

Therefore, we need to do *subword tokenization*. This technique rely on the principle that frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords. For instance "annoyingly" might be considered a rare word and could be decomposed into "annoying" and "ly". Both "annoying" and "ly" as stand-alone subwords would appear more frequently while at the same time the meaning of "annoyingly" is kept by the composite meaning of "annoying" and "ly". Subword tokenization allows the model to have a reasonable vocabulary size while being able to learn meaningful context-independent representations.

As a pretrained model only performs properly if you feed it an input that was tokenized with the same rules that were used to tokenize its training data, we use WordPiece tokenizer which is the subword tokenization algorithm used for BERT. WordPiece first initializes the vocabulary to include every character present in the training data and progressively learns a given number of merge rules.

4.2. Transformer encoding

Once the vocabulary is built, a tweet is represented as a sequence of one-hot encoding vectors. Transformer processes it through a self-attention layer. For each input vector, the first step in computing self-attention is to create three vectors : a *query* **q**, a *key* **k** and a *value* **v**. These vectors are calculated by multiplying the input vector with three different matrix that we learn during the training :

$$\mathbf{q} = W_q\mathbf{x}$$
$$\mathbf{k} = W_k\mathbf{x}$$
$$\mathbf{v} = W_v\mathbf{x}$$

Then, we calculate an attention score for each word according to another word within its context. The score determines how much focus to place on other parts of the input sentence as we encode a word at a certain position. The score of the query vector $\mathbf{x}_i$ with respect to another token vector $\mathbf{x}_j$ is $q_i \cdot k_j$. In order to have a probability distribution of the attention among the different words we use softmax function:

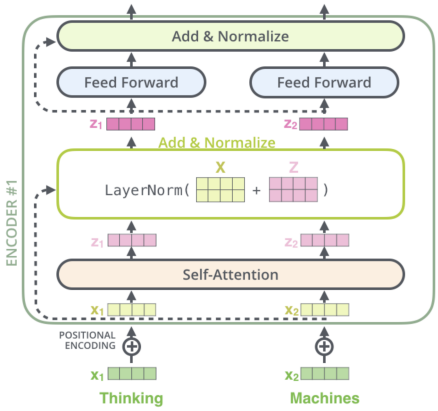

Figure 2. Transformer encoder layer diagram from alammar.github.io/illustrated-transformer/

$$\boldsymbol{\alpha}_i = \left[\frac{e^{q_i \cdot k_j}}{\sum_k e^{q_i \cdot k_k}}\right]_j$$

is the attention vector for query token $\mathbf{x}_i$.

From this attention vector and the value vector, we compute another vector z that will be fed to a feed-forward layer: $\boldsymbol{z}_i = \boldsymbol{\alpha}_i \odot \boldsymbol{v}_i$ (component-wise multiplication).

Transformer uses *multi-head attention* which is the same mechanism replicated multiple times with different sets of W_q, W_k and W_v to have multiple representation subspaces. In matrix form, it sums up to :

$$Z = \text{softmax}(\frac{1}{\sqrt{d_k}}QK^T) \odot V$$

where d_k is the dimension of key vectors.

With multi-head attention, we end up with different Z matrices that we concatenates and then multiply by another matrix W_o which is also learned during the training. Hence, the resulting matrix captures the information from all the attention heads. We can send it to the feed-forward layer.

Finally, in order to capture dependencies between tokens of the sequence, the model needs to have a sense of their position. This is why a positional encoding, which we will not detail here, is added to the input vector.

5. Evaluation

5.1. Dataset

We use the dataset provided by the Kaggle competition cited before. It includes 10875 tweets where nearly half of these tweets is about true disaster (43 percents are about disaster).

A disaster tweet will speak specifically about disaster like "flood storm rain people train issue severe weather rescue violent" or "crash kill accident helicopter air train fear say police car". However, a non disaster tweet does not have

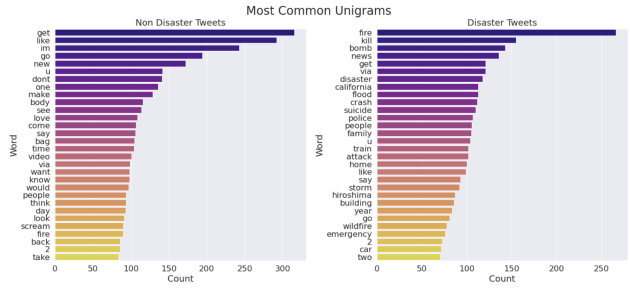

Figure 3. Unigram sorted by frequence in the dataset.

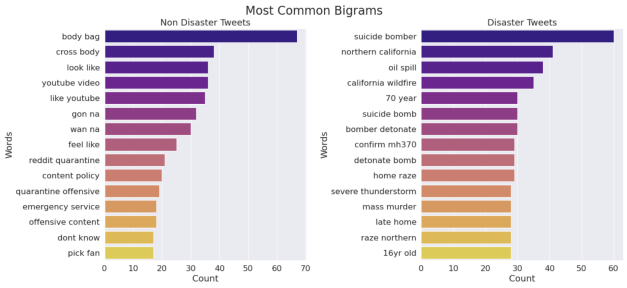

Figure 4. Bigram sorted by frequence in the dataset.

connection with disaster like :"get lol blow good bomb first day demolish someone play" even though some words could fool a poorly trained algorithm.

Statistically, we observe that non-disaster tweets are longer than the disaster ones. When we focused precisely on the words used, we notice that disaster tweets are more about terms indicating disaster such as wildfire or bomb (Figure 3). However the words used in non-disaster tweets are generic. Then, if we look at bigrams (Figure 4) we have the same conclusion as above. That it to say some bigrams about disaster come more often. Bigrams in non-disaster tweet are generic.

Finally, we can use Non-negative Matrix Factorization (NMF) on TF-IDF features to do some topic modeling :

Topic 1	get lol blow good bomb first day demolish someone play
Topic 2	body bag cross shoulder woman full lady read ebay really
Topic 3	scream fuck phone face good song loud hit baby time

Table 1. Some topics defined by key words for non disaster tweets.

Topic 1	fire forest building truck evacuate wild burn california service set
Topic 2	hiroshima atomic bomb year japan still anniversary 70 war bombing
Topic 3	california wildfire home northern late news raze abc collapse burn

Table 2. Some topics defined by key words for disaster tweets.

5.2. Implementation details

We divide our dataset into a training set and a validation set with a 80-20 split. For the implementation we use

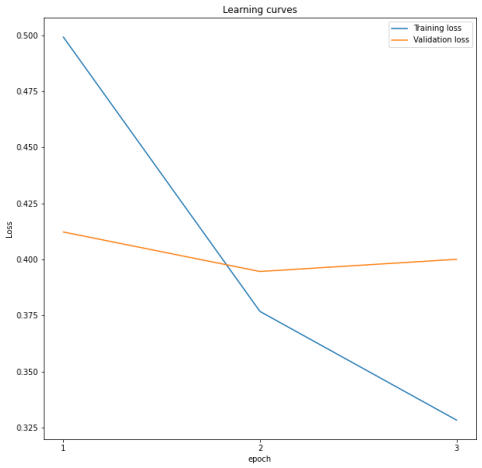

Figure 5. Learning curves for 3 epochs.

the HuggingFace library to load the BERT tokenizer and architecture pretrained on Wikipedia. We used the *bert-base-uncased* model which has the following properties : 12 Transformer layers, hidden space of dimension 768, 12 multi-heads attention and 110M parameters. On top of this specific architecture we simply add one feed-forward layer to solve our specific downstream classification task.

Each sequence of token has a size of 84 which is the maximum length in our dataset. Smaller sequences are padded to this size with a special token `[PAD]` and the tokenizer provides vectors called "masks" that tell where are the non padding tokens. It is useful for a forward pass into a Transformer layer.

To train our network, we use AdamW [8] which is a stochastic optimization method that modifies the typical implementation of weight decay in Adam, by decoupling weight decay from the gradient update. After some finetuning, we use a learning rate of $10e^{-6}$ and trained the model only on 3 epochs.

5.3. Evaluation metrics

We use the basic accuracy and F1-score such as implemented in `scikit-learn` :

$$\text{Acc} = \frac{\text{Nb. of correct predictions}}{\text{Nb. of predictions}}$$

$$\text{F1-score} = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.4. Results

After training only on 3 epochs we were able to achieve a test accuracy of $83\ \%$ and a F1-score of $81\ \%$.

From the learning curves we see that only after 2 epochs we might start overfitting the training set. This shows that using such a deep pretrained network for such a task does not require a lot of training. The model is very well suited for our classification task and we don't need to develop models from scratch to achieve good performances.

5.5. Conclusions

This work explores the transfer learning approach which consists in using pretrained features of a very large deep neural network such as BERT to solve a specific downstream task on another dataset. By using BERT, which is the state-of-the-art in many different Natural Language Processing tasks, we were able to achieve good performances on a difficult problem : telling if tweets were fake disasters or not.

This is useful when the amount of training data is limited and we want to use knowledge from other tasks. This is also an interesting approach when we don't have time or knowledge to do handcrafted feature engineering. Here, the Transformer architecture allow to learn a good feature extraction by taking advantage of the self-attention mechanism which takes the input sequence as a whole (unlike RNNs that read it from left to right) and tells us which words are useful to build efficient and robust representations of the tokens.

Better results could have been obtained by doing better finetuning.

References

[1] S. Singh, A. Subramanya, F. Pereira, and A. McCallum, "Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia," *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012*, vol. 15, 2012. 1

[2] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011. 1

[3] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning*, pp. 977–984, 2006. 2

[4] Z. Yun-tao, G. Ling, and W. Yong-cheng, "An improved tf-idf approach for text classification," *Journal of Zhejiang University-Science A*, vol. 6, no. 1, pp. 49–55, 2005. 2

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. 2

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 2

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017. 2

[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017. 4