Hashtag Prediction for Tweets on the Ukranian Conflict

Emma Costa - mat. 06114A - A.A. 2021/22
Report on project n. 5 for the course "Algorithms for Massive Datasets"

Abstract

Predicting the first hashtag of a tweet can be tackled as a classification problem. This paper describes the work done on a Kaggle dataset to train an artificial neural network that predicts the first hashtag for a selected subset of these tweets, i.e. the ones with the 20 most frequent hashtags. The dataset considered for this task contains tweets concerning the Ukraine conflict [1]; with the use of a Convolutional network, the model was able to predict the right hashtag with an accuracy of almost 60%. Looking at the data, it seems reasonable that these results are not distant from what a human expert could achieve on the same task.

1 Introduction

1.1 Neural Networks

An artificial neural network learning algorithm, or neural network, is a computational learning system that uses a network of functions to understand and translate a data input into a desired output. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

The neural network learns from processing many labeled examples (data with "answers") that are supplied during training and using this to learn what characteristics of the input are needed to predict the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results. The more examples the program sees, the more accurate the results typically become, because the program learns with experience.

This technique can be used for many tasks and environments, such as image recognition, medical diagnosis, finance, etc.; networks can be designed in many different ways and include various types of layers, that refer to a group of nodes that execute a specific operation on the data.

1.2 The Tweets Classification Problem

I approached the task by considering the issue as a classification problem, which makes it a supervised learning technique: every data is associated with a corresponding label; we've seen a similar example during the course, regarding the digit recognition [6]. I also took inspiration from some notebooks on Kaggle and the Keras documentaton (see references at the end).

In this case, the tweets are the input vector, while their label is represented by the first hashtag of the tweet: this defines the class to which the single tweet belongs; I decided to only consider the first one, as the tweets may contain even 10 or more hashtags, but the single input could only belong to one class.

Since the number of hashtags is very large, and classifying the tweets in a great number of classes would have been too complicated for the network to learn, I narrowed it down to consider only the 20 most frequent ones (in the given input): the tweets that had any other hashtag as the first one were excluded from the dataset that was ultimately fed to the model. The task of the model is to predict the first hashtag for each remaining tweet among the 20 classes.

I used Pandas for the manipulation and analysis of the data, then Tensorflow and Keras for the building and running of the model.

2 Implementation

2.1 Project Setup

I started downloading the whole dataset through the Kaggle API, and began reading the data: the versions of the data I have used is the collection of the tweets that were published in the month of May. I proceeded unzipping the files for the days I had decided to consider: to develop and test the model I only took the datasets corresponding to the first three days of May. For the final tests, I took all of the datasets from that month: those sum up to approximately 9 million rows, even though, after filtering the data, only about 1/10 of the rows will be considered. This seemed like a reasonably large enough dataset to evaluate the scalability of the algorithm as the input size increases.

I created a single dataframe by accumulating all the rows in the files, in order to start operating on the data, which means filtering and transforming it to be suitable for the model.

2.2 Data Transformation

The transformation of the dataset was the most demanding task: the tweets were, of course, in different languages, contained characters and words (such as tags) that would not be significant for the training of the model, and also information that wasn't relevant for the task. For example, each tweet contained not only the text, an id and the user, but also the timestamp, the number of favourites and retweets, which I didn't need for my purpose. In particular, I decided to only consider English tweets: they represented the majority of them and given the fact that for my model I had to create a vocabulary of words, taking into account different languages would have made the job too difficult.

	tweetid	hashtags	text	language	is_retweet
0	1520553587276795905	[{'text': 'StopRussianOil', 'indices': [245, 2...	Remember this 👇 image next time you fix a meal...	en	False
1	1520553587490926594	[]	UA Збройні Сили України продовжують повертати ...	uk	True
2	1520553587763515392	[{'text': 'Politics', 'indices': [51, 60]}, {'...	Everything Is [Not] Fine: Half-... - via @pensi...	en	False
3	1520553587939676160	[{'text': 'Tigray', 'indices': [28, 35]}]	People #Tigray lives in condition of No water,...	en	True
4	1520553588086525952	[{'text': 'Russia', 'indices': [0, 7]}, {'text...	#Russia's forces have stolen "several hundred ...	en	False

Figure 1: Head of the dataset after selecting the needed columns.

I started by considering only a portion of the columns: 'text', 'hashtags', 'tweet_id', 'language' and 'is_retweet' (fig. 1); I used the last two to filter the data and consider only English tweets, and then to avoid all the retweets because they didn't bring new information to the dataset, as the retweet refers to an already existing tweet.

Now, the only columns left were 'text', 'hashtags' and 'tweet_id'; in order to filter the tweets that I needed, i.e. those that had a popular hashtag as the first one, I had to extract the first hashtag of each one of them and place it in a new 'first_ht' column: I did this with the 'word_tokenize' function of the nltk library, extracting it from the 'hashtags' column; then I dropped the original 'hashtags' column.

Because the hashtags I wanted to consider are only 20, I extracted the 20 most frequent hashtags by performing a 'value_counts()' on the 'first_ht' column, to eliminate all the tweets that didn't have one of those as the first hashtag. This process, performed on the whole dataset of the tweets published in May, narrowed the data down to approximately 700.000 rows.

At this point I focused on the processing of the text. In order to train the model on the meaning of the tweets, these should have been "cleaned" from everything that doesn't add meaning: stop words,

	tweetid	text	first_ht
0	1520553587276795905	Remember this 👇 image next time you fix a meal for your family.\n\nEvery day, Russia sells 700M euro worth of oil and gas to the world. It's your tax money, too.\n\nW...	StopRussianOil
2	1520553587763515392	Everything Is [Not] Fine: Half-... - via @pensignal #Politics #Art #Trump #JerryNelson https://t.co/bZodwHyd3W	Politics
4	1520553588086525952	#Russia's forces have stolen "several hundred thousand tons" of grain in the areas of #Ukraine they occupy, Ukraine's deputy agriculture minister says.\n\nhttps://t.c...	Russia
18	1520553593430024193	#Russian "killer squad has been linked to vile war crimes against civilians in Bucha—and their own severely injured brothers-in-arms"\n\nhttps://t.co/JQwaX0DBk2\n\n#S...	Russian
20	1520553593950068736	Cain is a prominent guy. In all his habits, facial expressions.\nOur shelter needs your help!\nRaising funds food for animals.\nPayPal: dogandcat.helper@gmail.com\nht...	Ukraine

Figure 2: Head of the dataset after creating the 'first_ht' column to store the first hashtag of the tweet.

urls, emojis, non ascii characters, words with less than 3 characters, tagged users (words beginning with "@") and the hashtags themselves. For the tweet cleaning process I used [3] as reference. I used the swifter library for most of these tasks, and the NLKT library to eliminate stop words.

The tweets, now transformed into strings of words that could each give some particular meaning to it, are now ready to be studied by the model, together with their label, in order to train it.

2.3 The Model

Artificial neural networks are developed so that models can mirror the nonlinear decision-making process of the human brain: they can be trained to make complex decisions or understand abstract concepts and objects. The model will build from low-level features to complex features, understanding complex concepts. Each node within the network is weighted depending on its influence on other artificial neural network nodes. Like other machine learning models, optimisation of artificial neural networks is based on a loss function. This is the difference between a predicted and actual output. The weighting of each node and layer is adjusted by the model to achieve a minimum loss.

For my model I started considering the example of deep learning models seen during the course [6]. However, I had to change the approach since I had to deal with a textual input. As shown in Keras documentation [7], a possibile newtork topology to learn representation of text is a 1-dimensional Convolutional Neural Network. This kind of network search for patterns anywhere in an input sequence in the same way a "regular" (2-dimensional) CNN search in an image.

Convolutional neural networks have three main types of layers, which are:

- convolutional layer
- pooling layer
- fully-connected layer

The convolutional layer is the first layer of the network, and represents the core building block of a CNN, where the majority of computation occurs.

The dataset needed to be split in the three sets: training, validation and test. Because certain topics may have been discussed more in some days than others, the tweets had to be shuffled before this passage, so that they would be uniformly distributed in terms of content and topics.

The training set (to which I assigned 80% of the data) is used by the model to fit the parameters (weights) and learn the optimal combinations of variables that will generate a good predictive model.

The validation set (10% of the data) is used to tune the hyperparameters (the architecture) of the classifier, for example the number of hidden units in each layer.

The test set (10% of the data) is a set of examples used only to assess the performance of a fully specified classifier.

I created the input dataframe for the data (text of the tweet) and the labels (first hashtag of the tweet), and I encoded the latter with a one hot encoding. To apply a one hot encoding also to all input

```
import tensorflow as tf
from tensorflow.keras import layers

text_input = tf.keras.Input(shape=(1,), dtype=tf.string, name='text')
x = vectorize_layer(text_input)

x = layers.Embedding(max_features, embedding_dim)(x)

x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=2)(x)

x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=2)(x)

x = layers.GlobalMaxPooling1D()(x)

x = layers.Dropout(0.6)(x)

x = layers.Dense(128, activation="relu")(x)

x = layers.Dropout(0.6)(x)

predictions = layers.Dense(20, activation="sigmoid", name="predictions")(x)

model = tf.keras.Model(text_input, predictions)

model.compile(loss="binary_crossentropy", optimizer=tf.keras.optimizers.Adam(learning_rate=0.00003) ,metrics=["accuracy"])
```

Figure 3: The model for the neural network.

text tokens, I proceeded with a process called vectorization: this consists in creating a vocabulary of the most used n words (20000 in my case) in the dataset. Vectorization maps words or phrases from vocabulary to a corresponding vector of real numbers, which will be given as input of the specific word to the model. This represents the first (non trainable) layer of the model itself, which contains 8 more layers:

1. the embedding, to learn dense vectors of fixed size for each word;
2. two convolutional layers, both with 128 filters (one for each embedding dimension), a kernel size of 7 and a step size (stride) of 2;
3. a global max pooling, to reduce the output to a 1-dimensional vector (same size as embedding);
4. two dropouts with a 0.6 dropout rate, because my model tended to overfit;
5. two dense layers, one with a RELU activation function, the last one with 20 nodes and a sigmoid activation to generate a probability distribution over the 20 classes.

Each of the 20 outputs represents the probability of each hashtag being the correct one for the given tweet. I tried various values for the learning rate and ultimately decided to use a very small value (0.00003) as I noticed that higher values resulted in the model stopping improving after two iterations.

I chose to train it with 15 epochs as the accuracy level didn't grow much after that point. Then, I tested the model on the test set, which gave the same accuracy and loss of the validation set, as expected. Lastly, I print out some results of a random group of tweets taken from the test set, considering the original text of the tweet, the first hashtag and the top 3 hashtags predicted by the model.

3 Results

At first sight, the results might not seem excellent, as the model is able to predict the right hashtag approximately half of the times: it reached a 0.56 maximum accuracy and 0.108 loss on the validation set (and the test set), with a 0.60 accuracy and 0.099 loss on the training set. This gap between training and validation results denotes a little overfitting, which occurs when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

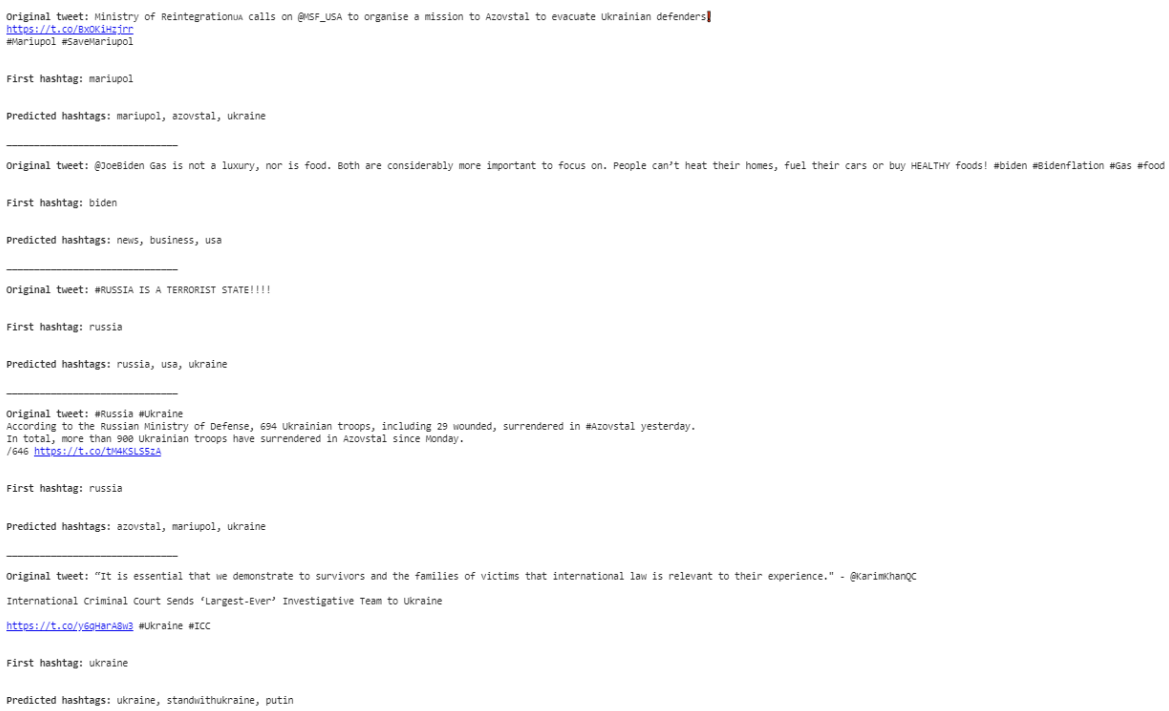

Figure 4: An example of the predicted result compared with the expected one.

However, there are some factors that have to be taken into account:

1. the goodness of an algorithm must be compared with the results that a human expert could achieve on the same task: looking at the tweets and the corresponding hashtags, it seems reasonable that even a human will only get maybe 50/60% of them right;

2. among the hashtags, some are very similar to each other (ukraine, ukranian, ukrainewar; russia, russian; etc.), which makes it harder to predict the exact one;

3. the data isn't totally cleaned, as it contained a great amount of tweets that didn't exactly concern the Ukranian conflict.

For these reasons, the accuracy and the loss of the model were hard to optimize further. For the selected tweets that I printed out as example I considered the top 3 predictions and not just the first: this way, the model predicted approximately the 70/80% of the hashtags right. Thinking of a use for this model, in fact, there could be an engine that suggests hashtags while writing the tweet, and ideally it could give the 3 most probable ones.

4 Possible Optimizations

The main optimization that could be done for this task is using a Recurrent Neural Network; RNNs can use their internal state to process variable length sequences of inputs. In these kinds of networks, we perform the same operation at each step, with the input to each step being dependent upon the output from prior steps. This way, understanding the meaning of the tweets and classificating is much easier than having a model that takes the words of the tweets all at once.

Another one could be clustering the hashtags in different topics, and predicting the topic, instead of a specific hashtag, for every tweet. Grouping similar hashtags means finding some kind of distance between them, to take the closest ones and aggregating them. For example, this could be done by considering how similar the two strings are, which means considering their edit distance, or by

computing the Jaccard similarity between every pair of hashtags:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Here, with A and B being two specific hashtags, the similarity is given by the number of tweets where both hashtags appear divided by the total number of tweets that contains at least one of them.

5 References

1 Ukraine Conflict Twitter Dataset

2 Tweets on Ukraine Crisis

3 Generate Wordcloud from English Tweets

4 Sentiment analysis for tweets on the Ukraine conflict

5 Notes on Convolutional Networks

6 Digit recognition

7 Text classification from scratch

8 Neural Network Models Explained

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.