Sentiment analysis of the spread of hate in tweets using RNN

Text Mining and Social Media Mining

Alberto Delgado López – 434984 *Serena Bozheku – 429992*

Introduction to the analyzed problem

Social networks have turned the web into a very popular social interaction platform where billions of individuals around the world interact, share, post and conduct countless activities on a daily basis. They have turned into an unrivalled source of real-time information, making them an ideal tool to analyse social phenomena, political opinions, ethical stances on issues, opinion and interests about products.

One of the main intentions in the analysis of the information derived from social media is to know what people think. Here comes into play **Sentiment Analysis**. The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world. Some examples of its applications are: in economics, where shifts in sentiment on social media have been shown to correlate with shifts in the stock market; politics, i.e. the Obama administration used sentiment analysis to measure the public's opinion to policy announcements and campaign messages; market research and customer service.

Twitter is one of the biggest and most popular social networks with around 300 million users, 40% of which are active on the platform multiple times per day. Hence, it's understandable the need to monitor the sentiment of the users, especially if it can help stop the spread of hateful and damaging massages.

Therefore, the purpose of this project is the implementation of a model for Sentiment Analysis on tweets extracted from Twitter. The tool used for this porpouse is **Recurrent Neural Networks (RNNs)**. The main goal is the implementation and evaluation of a RNN model with different layers as LSTM, GRU, and one attempt of adding a CNN layer to compare its performances. Pretrained word vectors from GloVe model trained with tweets are also loaded in our model in an Embeddign layer, which will help to improve the model too.

Data set description

The data used was found on Kaggle. The dataset used to build and evaluate the model contains 31963 rows and three columns:

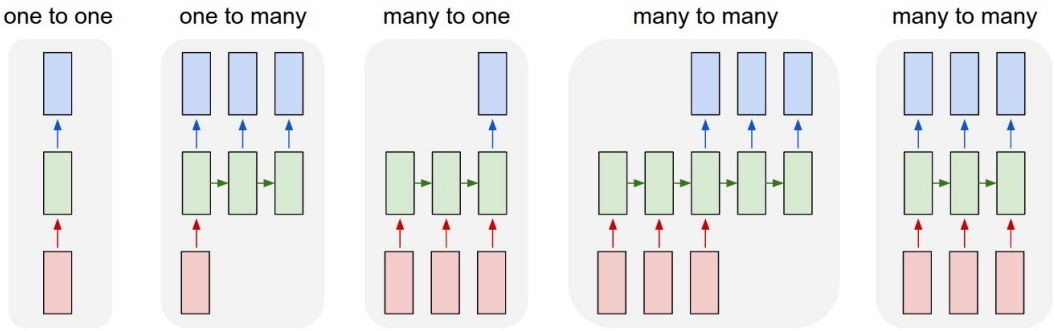
- *id* - ID assigned to each tweet
- *label* - 1 if the tweeter is considered hateful and 0 non-hateful
- *tweet* - string containing tweet's text.

Additionally, in this source a diferent dataset with 17197 unlabeled tweets can be found, which we will be labeled using the final and improved model.

Structure of the model:

The main model structure implemented in this project is **Recurrent Neural Network (RNN)**:

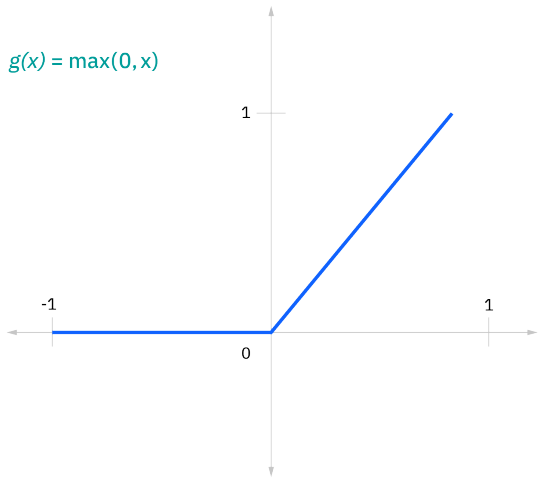
> Commonly applied to language translation and natural language processing, speech recognition and image captioning. It uses sequential or time series data, and it stores information or states of previous inputs in memory in order to generate the next output of the sequence. It shares the parameters across each layer of the network. The weights of the parameters are adjusted through the process of backpropagation and gradient descent.
>
> Two possible issues of RNNs: "*exploding gradient*" and "*vanishing gradient*".
>
> Different types: *One-to-Many*, *Many-To-Many* and *Many-to-One* (a sequence of data as input and a single output to be predicted, like in classification problems).



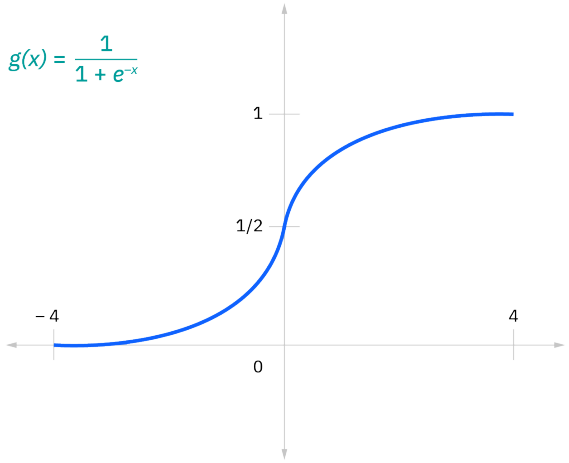
(Source)

Activation functions are implemented in **Dense** added to the model layers for determining whether a neuron should be activated during the sequence. The nonlinear functions typically convert the output of a given neuron to a value between 0 and 1 or -1 and 1. The ones used in the developed model are:

Relu



Sigmoid



In the model more layers are added such as Dropout, LSTM and Gru to account for the issues that might arise when implementing a RNN model:

> Both **Long Short-Term Temory (LSTM)** and **Gated Recurrent Unit (GRU)** layers are both variant RNN architectures used to account for "*exploding gradient*" and "*vanishing gradient*" issues.
>
> **Dropout** is used "*to prevent over-fitting*" by ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random.

Also, pre-trained weights are used in the model by passing pre-trained GloVe word vectors to the **Embedding** layer. This improves the performance of the model since it gives information to the model about the probability for most of the words to appear in the tweets and how:

> **Global Vectors for Word Representation (GloVe)** is an unsupervised learning algorithm developed by researchers at Stanford University aiming to generate word embeddings (vector representations for words). Its training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space which have the potential for encoding some form of meaning. It's essentially a log-bilinear model with a weighted least-squares objective**.**

The attempt of adding **Convolutional Neural Network (CNN)** layers is also made in the process of building the model. The mix of convolution and max pooling layers with RNN cells has been used recently and achieved state of the art results in NLP problems. In short:

> The **Conv1D** has filters that determines the output dimension, kernel size which is the window size for convolution and padding which determines if the input should be padded (add zeros around the matrix) or not.
>
> The **MaxPooling1D** contains the parameter pool size that determines the window to look for the max value.

Structure of the final model (in the Results section building process is explained):

Layer (type)	Output Shape	Param #
Embedding (Embedding)	(None, 34, 200)	6070600
Dense1 (Dense)	(None, 34, 200)	40200
dropout_72 (Dropout)	(None, 34, 200)	0
LSTM (LSTM)	(None, 34, 64)	67840
GRU (GRU)	(None, 64)	24960
Dense2 (Dense)	(None, 64)	4160
dropout_73 (Dropout)	(None, 64)	0
Dense3 (Dense)	(None, 32)	2080
Output (Dense)	(None, 1)	33

```
Total params: 6,209,873
Trainable params: 139,273
Non-trainable params: 6,070,600
```

Optimized parameters

The main focus of the model improvement was on the following parameters:

- *batch size of the model* - it pertains to the amount of training samples to consider at a time for updating the network weights
- *trainable argument of the embedding layer* - false for keeping the weights fixed and True if the weights will be updated during the training,
- *the dropout rate* - fraction of the input units to drop,
- *adding Conv1D layer from Convolutional Neural Network* - creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs.

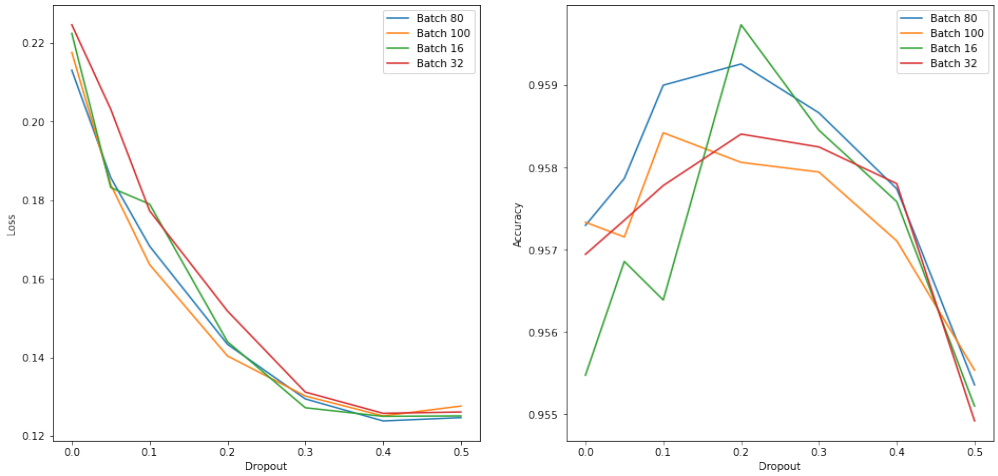
Results

The following models were built and evaluated:

1. **Basic RNN model with LSTM, GRU and Dropout layers**
   `Loss: 0.1631 Accuracy: 95.51 %`

2. **Adding Glove pretrained word vectors to the model**
   `Loss: 0.1492 Accuracy: 95.92 %`

3. **Parameters tuning**



The highest accuracy is achieved with *drop=0.2* and *batch_size=16*, but since the loss could be still improved by increasing the drop rate, the chosen paramaters will be *drop=0.3* and *batch_size=80* (this also increases the speed of the model trainings).

Chosen parameters model

```
Train:
 Loss: 0.0191 Acc: 99.42 %
Test:
 Loss: 0.1359 Acc: 96.21 %
```

4. **Trainable Embedding layer**

```
Train:
 Loss: 0.0009  Acc: 99.97 %
Test:
 Loss: 0.3721  Acc: 95.57 %
```

Performance improved for the training dataset, but worsened for the test dataset. This seems logical, since by training more parameters we make the model more specific, i.e. we train it to make better predictions on tweets similar to those used to train it. This makes the model less general, and it predicts new and more different tweets worse. This is similar to what happens when we reduce the drop_out rate.

Therefore, we stablish the *trainable argument = False*.

5. **Convolution layer**

```
Train:
 Loss: 0.0066  Acc: 99.74 %
Test:
 Loss: 0.2336  Acc: 95.4 %
```

In a similar way, like the trainable embedding layer, the convolution layers improved the performance of the model for the training dataset, but it got worse for the test one. The test loss increases with the trainings and the accuracy seems to lightly decrease and it becomes more 'unstable'.

Therefore, we decide to reject the idea of adding CNN layers to the model.

The final decision is to choose the **3rd one** as the best one, and to do an **evaluation of the predictions** made by this model.

Train dataset:	Test dataset:
Total number of tweets: 25569 Hatred labeled: 1786 Non-hatred labeled: 23783	Total number of tweets: 6393 Hatred labeled: 456 Non-hatred labeled: 5937
Predicted train data:	Predicted test data:
Loss: 0.0191, Acc: 99.42%	Loss: 0.1359, Acc: 96.21%
Total number of tweets: 25569 Hatred labeled: 1693 Non-hatred labeled: 23876	Total number of tweets: 6393 Hatred labeled: 360 Non-hatred labeled: 6033
Confusion matrix:	Confusion matrix:
[[23755 28] [121 1665]]	[[5864 73] [169 287]]

Predictions on new data

```
Total number of tweets: 17197
Hatred labeled: 969
Non-hatred labeled: 16228
```

tweet	label_predicted
@user idk what's worse.. that joey didn't get to raise the cup, i don't get to see his beard anymore, or that sid won htâ¦	0
that awesome feeling when the client says they have no comments and really like it. #designlife #designthinking #designmatters	0
â£â£â£best dayâ£â£â£ #summer #relax #i #2016 #life #friends #girl #instasize #instadailyâ¦	0
@user i had no idea she was transphobic and misogynist! sad. her mom talks a lot about in the #judo world.	1
oh noes, @user ist leer. #moment	0

tweet	label_predicted
@user i can't wait till you are impeached traitor. you deserve to be in jail for so many things. #impeachtrump #traitor	1
@user is not an #opinion. racism is #degrading people based on race to justify #discrimination or open #violence againstâ¦	1
@user @user we hear your ,#bigotry ,#homophobia, #sexual assaults, #womanizing, #treason , #lies you hearâ¦	1
there is lot of #racialhate in america that provides these two lunatics suppo to perpetuate hate. #dumptrump	1
women nude anal girls nude sex act	1
this year in stupid (2016) #regressiveleft #lefty #maga #feminism #endrapeculture #migrantcrisis #corruption	1
book - enduring conviction: fred korematsu & his quest for #justice #humanrights htâ¦	1
.@user caught with the hand in the cookie jar. he should just man up and admit he is racist	1
#wordstwittermademehate "alt-right". let's just call it what it is, folks.	1
somebody tell @user #thepeople didn't chose @user #hispeople choose #trump ~ big difference!	1

The interesting thing of these particular sets of tweets is that during the collection of them the trending topics were the #TrumpImpeachment and the #BlackLivesMatter movement. The model classify correctly lots of tweets but probably is failing on others because it was trained that words like 'Trump' and 'race', 'black', 'women', etc. usually belong to hate tweets, even when it is not exactly the case. Lots of tweets are not saying anything hateful perse, but the topic and context of the sentence, from the political and social point of view, is creating polarization of the society even if the tweet was using respectful words and manners.

Conclusions

The best model performance was shown by the RNN model with LSTM, GRU, Dropout and GloVe Embedding layers.

The parameters that showed best results where the *batch_size=80* and *drop_out=0.3*.

Training the Embedding layer parameters or adding CNN layers to the model did improve the results for the train data (they made it more specific to our tweets), but did not for the test data, we need a more general model which predicts more diverse tweets. That is why the final model was the one chosen without Training Embedding layer and without CNN layers.

A final Accuracy of 96.21% and a Loss value of 0.1359 in the test data was achieved with the final model. This is a pretty good result that we can check by looking at the predictions made on the new_data set of unlabeled tweets which were labeled by the model.

Bibliography

1. David Cecchin, Chester Ismay, Adrián Soto, *Recurrent neural networks for language modelling in python*, DataCamp, https://app.datacamp.com/learn/courses/recurrent-neural-networks-for-language-modeling-in-python

2. Arkhoshghalb (2019, January 7), *Detecting hate tweets*, Kaggle, https://www.kaggle.com/arkhoshghalb/detecting-hate-tweets/notebook

3. *Text classification with an RNN,* TensorFlow, https://www.tensorflow.org/text/tutorials/text_classification_rnn

4. Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014), *GloVe: Global Vectors for Word Representation*, https://nlp.stanford.edu/projects/glove/

5. Analytics Vidhya. (2020, July 19), *What is Elmo: Elmo for text classification in python*, https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

6. Team, K., *Keras documentation: Model training APIs*, Keras, https://keras.io/api/models/model_training_apis/

7. Team, K., *Keras Documentation: CONV1D layer*, Keras, https://keras.io/api/layers/convolution_layers/convolution1d/