A close up of a mans face

Description automatically generated

“Big Data Content Analysis”

A close up of a logo

Description automatically generated

Sentiment Analysis on Twitter Data for Airline Companies using Deep Learning Approaches

Dagre Adriani | Kondyli Afroditi | Gialama Niovi

2019

**Table of Contents:**

[1. Description 2](#_Toc18775253)

[2. mission 3](#_Toc18775254)

[Problem analysis 3](#_Toc18775255)

[What has already been done 4](#_Toc18775256)

[The approach followed 6](#_Toc18775257)

[3. data 6](#_Toc18775258)

[4. Methodology 10](#_Toc18775259)

[Our First Model: Convolutional Neural Networks (CNNs) 10](#_Toc18775260)

[Our model and its Structure 12](#_Toc18775261)

[Model Input 14](#_Toc18775262)

[Loss Function, Evaluation Measures, Optimizers 15](#_Toc18775263)

[5. Results 17](#_Toc18775264)

[Bibliography 19](#_Toc18775265)

## Description

In recent years, there is an increasing effort on behalf of companies around the world, to pay close attention to the voice of customers in order to enhance the customer experience. Collecting and analyzing customer’s feedback and comments from social media about companies, services and products, provides them the advantage to have better information in order to make strategic decisions, while having an accurate understanding of what the customer actually wants and, as a result, a better experience for everyone.

For this reason, more and more companies deploy sentiment analysis in social media platforms in order to understand what customers like or dislike about the products/services they offer. Another advantage that sentiment analysis offers to these companies is that, they can become more competitive in the market, since they can compare their products/services with their competitor ones.

We are a business intelligence services provider that has been appointed by United Airlines in order to build a model that will classify accurately all tweets that refer to the company and/or its main rivals. The aim of our project is that, the company will decrease its response times in negative tweets or its reactions to the positive ones. At the same time, it will conduct further analysis to the tweets concerning its rivals in order to predict similar negative or positive occurrences that may arise.

In the current assignment we are utilizing a dataset of 14.640 tweets, each one of them referring explicitly to one major U.S. airline. The most important feature of the dataset that makes it extremely useful is that tweets are already classified between positive, negative or neutral by the contributors. We can use this classification in order to extract the appropriate information so that in the future, we would not need any manual effort to separate negative tweets from positive or neutral. All tweets are coming from February 2015.

In other words, in this project we will utilize sentiment analysis which focuses on Twitter data, by which airlines companies can automatically process what their consumers write in natural language and get remarkable insights in order to take decisions and make some to-the-point actions. For this reason, we will start by making some pre-processing steps at the beginning, such as data cleaning. After that, we will use machine learning techniques (such as Naïve Bayes) and at last, we will deploy deep learning architectures such as the Recurrent Neural Network and the Convolutional Neural Network,

As we all know, the importance of the social media sentiment analysis every day becomes more and more important for every single company, and not only companies that deal with the airline sector, but generally for all those companies that are active in any marketing sector and focus on retail sales or retail service providing. This is due to the nature of retail markets and the huge number of people who may be current or potential customers of such a retail company and can be affected by any negative or positive criticism. Furthermore, the fact that social networks offer a level of anonymity and privacy which are conditions that are not easily achieved in actual interactions, creates a higher level of sincerity and openness on behalf of the customers compared to other communication channels [1].

There are numerous sentiment analysis solutions and they are being offered either as a separate web or desktop application, or as an Api in order to integrate it into your solution. Their main characteristic though is that what they offer is an already trained algorithm. The only thing the user can do is to provide the input data to be analyzed. This prevents the user from any desired customization he may wish to impose.

Our solution is to use Keras and Tensorflow as the backend in order to implement a sentiment analysis neural network from scratch. It will lack most of the fine polishing of all the existing commercially available solutions, however, it will have the advantage of being free and customizable.

The general business workflows into which a sentiment analysis project may add significant value are mainly the marketing ones, as we mentioned above. This comes natural as a social analysis tool, no matter its concept or its implementation details, is in essence a marketing tool. The most fundamental functionality it serves for a business or an organization is finding and analyzing consumer and public opinions about their products or services [2]. This broad functionality includes reputation management, customer feedback and crisis prevention.

## mission

### Problem analysis

Social media produce daily a vast stream of data that large corporations need to be in place to handle on real time. How can this be possible when each day each company receives hundreds of messages through all of its social media accounts? This is the problem we are trying to solve. Our mission is to build a model that predicts if tweet’s writer’s opinion is positive, neutral or negative about the company and the services it provides.

Our main purpose is not trying to provide a static summary description of our data, accompanied by some visualization. Although, for reasons of data exploration we are going to provide some statistical analysis, but this is not the main problem of the current project that is trying to solve. On the contrary, we are attempting to use our dataset in order to extract a statistical model that will help us classify future tweets into the three basic sentiment categories our dataset is divided into. Negative, neutral and positive.

After providing our model to our client, United Airlines, then the company can create a client that will consume in the form of a stream, all the tweets that refer to it or its opponents, through the Twitter API. After getting the tweets by using the Api’s, the company can build a pipeline that will receive it, perform some ETL process, feed it through our model and finally perform some action according to the sentiment category the tweet has fallen into. The transformation process can be exactly like the one we have provided in our code.

### What has already been done

Sentiment analysis is the automated mining of attitudes, opinions, and emotions from text, speech, and database sources through Natural Language Processing (NLP). As already mentioned above (chapter), in this analysis, a classification of opinions in text are held. [3].

For sentiment analysis are commonly used 3 approaches: machine learning-based, lexicon-based and hybrid. In the first one, named “Machine learning-based” approach, uses classification technique to classify text. Regarding the “Lexicon based” method, uses sentiment dictionary with opinion words and match them with the data in order to determine polarity. In more details, with this method they designate sentiment scores into words, describing the degree of their opinion, how Positive, Negative and Objective the words contained in the dictionary are [3]. The combination of these two approaches (machine learning and lexicon-based) to address Sentiment Analysis is called Hybrid. Although “Hybrid” method is not widely used, very often produces auspicious results compared to the other two.

In regard to the machine learning approach it refers to supervised classification. In this method two datasets are required, the training and the testing. The training dataset is used by an automatic classifier in order to ‘’learn’’ the different characteristics of documents, while the testing dataset is used to see actually how well the classifier performs. As an example, Naïve Bayes (NB), Maximum Entropy (ME) and Support Vector Machine (SVM) classifiers are universally used in sentiment analysis projects [3].

If we take a look at the literature, [4] compared the performance of these three classifiers (NB, ME & SVM) in Sentiment analysis at document level with different features (like considering only unigrams, bigrams, combination of both, combining unigrams and parts of speech, taking only adjectives and combining unigrams and position information) [3]. Their finding was that, more important is the feature presence than feature frequency. Moreover, when the feature set is small, NB performs better compared to SVM. On the other hand, when the feature space is increased, the SVM and ME classifiers might have better performance from NB, but ME deals with the problem of overfitting in this case.

Another classification technique has been proposed by [5] that uses stylistic and syntactic features. They introduced a new algorithm entropy weighted genetic algorithm (EWGA), which is hybrid genetic algorithm that uses the information gain heuristic to improve feature selection. Using this hybrid algorithm, they managed to achieve accuracy of 95.5% [3].

Lexicon-based, is the second of the three (mentioned earlier) approaches for sentiment analysis. In unsupervised techniques, classification is performed by comparing the features of a given text against sentiment lexicons whose sentiment scores are determined prior to their use. Sentiment lexicon contains lists of words and expressions used to express people’ s subjective feelings and opinions. For example, start with positive and negative word lexicons, analyze the document for which sentiment need to find. Then, if the document has more positive word lexicons, it is considered as positive, otherwise as negative. The lexicon-based techniques to sentiment analysis is unsupervised learning because it does not require prior training in order to classify the data which is a great advantage.

The basic steps of the lexicon-based techniques are [6]:

1. Preprocess each text (i.e. remove HTML tags, noisy characters etc.).

2. Initialize the total text sentiment score: s ← 0.

3. Tokenize text. For each token, check if it is present in a sentiment dictionary.

* If token is present in dictionary,

a) If token is positive, then s ← s + w.

b) If token is negative, then s ← s − w.

4. Look at total text sentiment score s,

* If s > threshold, then classify the text as positive.
* If s < threshold, then classify the text as negative.

Three methods can be used in order to create a sentiment lexicon: manual construction, corpus-based methods and dictionary-based methods. The manual construction of sentiment lexicon is a difficult and time-consuming task. In dictionary-based techniques the basic idea is to first collect a small set of opinion words manually with known orientations, and then to develop/grow this set by searching in the WordNet dictionary for their synonyms and antonyms. Hu and Liu [7] used this technique to find semantic orientation for adjectives. They used 30 adjectives as the seed list. The dictionary-based approach has a limitation which is that it can’t find opinion words with domain specific orientations [8].

Corpus based techniques depend on syntactic patterns in large corpora. Corpus-based methods produce opinion words with remarkable high accuracy. Most of these corpus-based methods need very large labeled training data. This method has an advantage that the dictionary-based approach does not have. It can help find domain specific opinion words and their orientations [3].

The most prominent work using unsupervised methods for opinion mining and sentiment detection is done by Turney et al. [8]. He used “poor” and “excellent” seed words as they appear more in social media for calculating the semantic orientation of phrases, where orientation is measured by pointwise mutual information. The sentiment of a document is then calculated as the average semantic orientation of all such phrases. Using this method an accuracy of 66% was achieved, for the movie review sector [3].

The combination of both machine learning and lexicon-based approaches can improve remarkably the sentiment classification performance. We are talking about hybrid techniques now. The main advantage of the hybrid approach using the combination of the lexicon/learning approaches, is to attain the best of both worlds-stability, as well as, readability from a carefully designed lexicon, but also high accuracy from a powerful supervised learning algorithm. This system uses a sentiment lexicon constructed using public resources for initial sentiment detection [3].

Recent work focuses also on deep learning models such as Recurrent Neural Networks (RNN) and Convolutional Neural Network (CNN) [9].

### 

### The approach followed

Neural networks have been traditionally used for sentiment analysis with the use of loops within the network architecture to model language dependencies [10]. Firstly, Convolutional Neural Networks (CNNs) learn to capture features regardless of where these might be. It makes sense to choose a CNN for classification tasks like sentiment classification since sentiment is usually determined by some key phrases. At the same time, Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP projects. A Recurrent Neural Network (RNN) creates loops between each node in the neural network. This makes it a very good candidate for sequential data, such as text. It can process sequences and create a state which contains information about what the network has seen so far. This is why RNNs are useful for natural language processing, because sentences are decoded word-by-word while keeping memory of the words that came beforehand to give better context for understanding. An RNN allows information from a previous output to be fed as input into the current state. Simply put, we can use previous information to help make a current decision.

The approach we decided to follow was to use a deep learning technique by utilizing Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNN) in order to create a statistical model.

## data

## 

Our dataset is called “[Twitter US Airline Sentiment](https://www.kaggle.com/crowdflower/twitter-airline-sentiment)” which was downloaded from Kaggle as a csv file. Its original source was from [Crowdflower’s Data for Everyone](https://www.figure-eight.com/data-for-everyone/) library.

Tweets were scraped from Twitter in February 2015 about each major US airline. Contributors then classified each tweet as either “positive”, “neutral”, or “negative” and cited the reason for a negative classification as well as a confidence score for the assigned label.

There are 14,640 rows and 15 columns. The included features are tweet id, sentiment, sentiment confidence score, negative reason, negative reason confidence, airline, sentiment gold, name, retweet count, tweet text, tweet coordinates, time of tweet, date of tweet, tweet location, and user time zone.

All our data were preprocessed in order to bring them at a certain state where they would give us the best results. The stages of this preprocessing were the following:

* Every tweet started with a reference to a specific airline company (i.e. “@AmericanAir”). We removed this as it has no semantic meaning.
* We also used regular expressions in order to remove any URL from the tweets.
* The final removals with the aid of regular expressions were all kind of symbols, as well as numbers – letters only.
* We converted all tweets to lowercase.
* We removed all common words that do not have any significant semantic meaning such as “of”, “over”, “than” etc.
* Finally, by using Wordnet, the publicly available lexicon database for the English language, we conducted lemmatization, which is the transformation of each word into a lemma. For example, transforming a word from plural to singular is one of the forms that lemmatization can take.

The form of our data that we actually fed with our model, were sequences of integers, each one representing to one word. The sequences were padded with zeros one the left, up to the length of the largest one. They can be seen below:

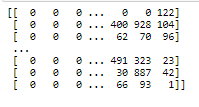


Figure 1 – The final sequences fed into our RNN model

Exploratory analysis on our data revealed that negative tweets are much more than the neutral or positive ones.

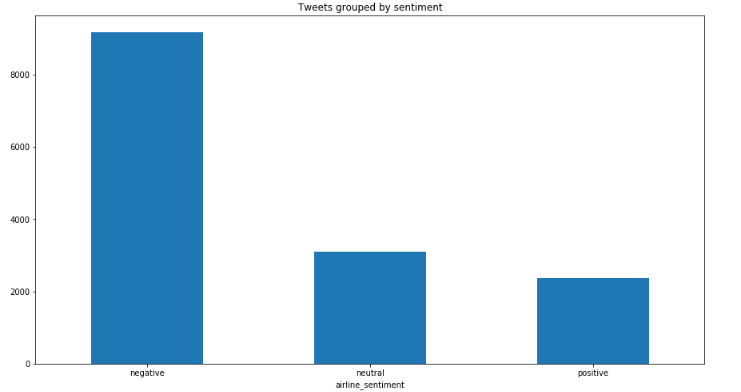


Figure 2 – Tweets grouped by sentiment

The distribution of the tweets per airline company can be found in the next Figure.

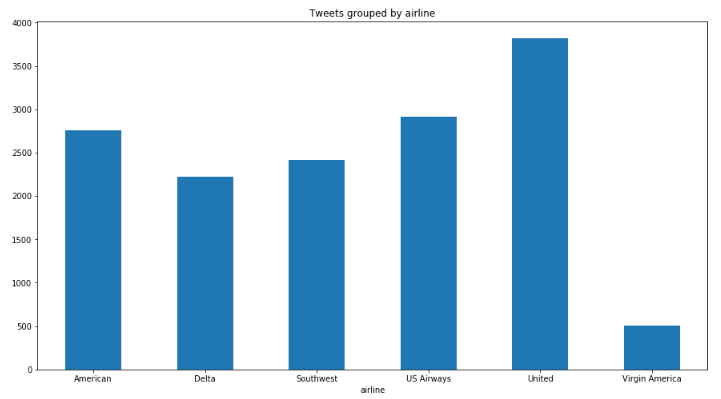


Figure 3 – Tweets grouped by airline company

If we combine these two bar-plots we created together, we can see the segmentation of tweets to positive, neutral and negative, per airline company.

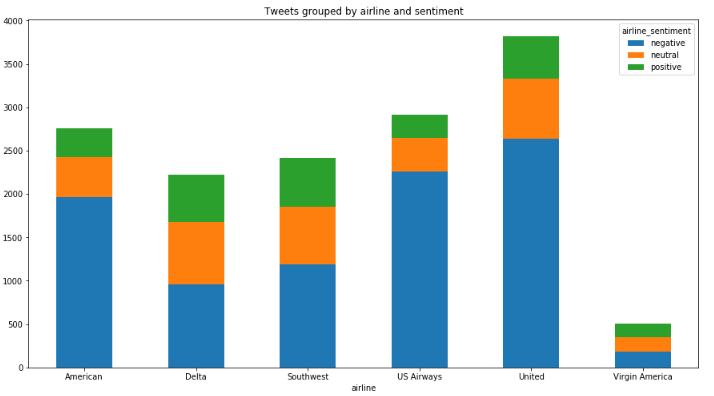


Figure 4 – Tweets grouped by airline and sentiment

We can easily see from the stacked bar-plot above that, United Airlines Company except having most of comments in total, is the one who gathered most of the negative comments when comparing to others. Virgin America Company, on the other hand, is the less popular airline company, since it gathered the least number of comments in total. Also, we can see that this company has almost equal number of positive negative and neutral tweets. Most positive comments were gathered for Southwest Airline Company, Delta Airline Company and United Airline Company, as we can also see in the plot.

Finally, in the following table, we can see the reasons that lie behind the negative tweets. “Customer Service Issues”, flight delays (“Late Flight”) and other reasons, that aren’t defined explicitly (“Can’t Tell”), have gained the most negative tweets by customers. Also, Flight Cancellations (“Cancelled Flight”), “Lost Luggage” and “Bad Flight” experiences contribute also to negative tweets.

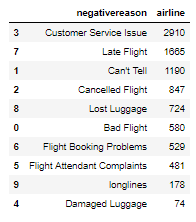


Figure 5 – Reasons of negative tweets

## Methodology

### Our First Model: Convolutional Neural Networks (CNNs)

The model type that we usε is Sequential. It allows us to build a model layer by layer. It is depicted on Figure 6.

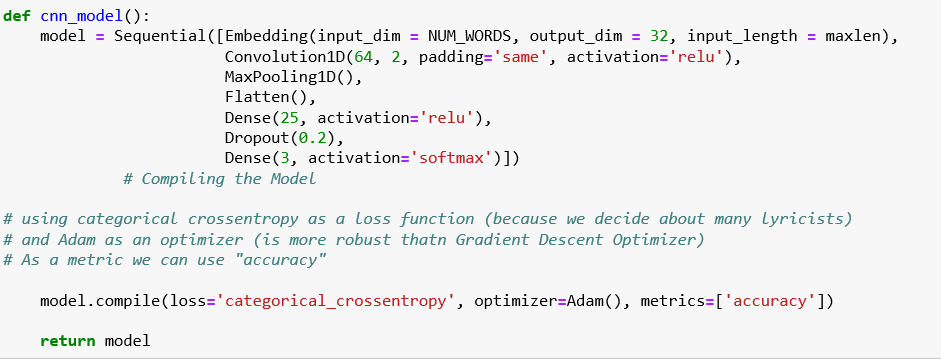


Figure 6 - Parameters of CNN model

Firstly, our CNN model consists of an initial convolution layer which receives word embeddings for each token in the tweet as inputs. The embedding layer learns to map word vectors into a lower dimensional vector space where distances between words correspond to how related they are. In our case the word embedding learned from scratch.

Ιn this layer, we passed the top 6000 most common tokens, defined the output dimension equals to 32 and maximum length of a tweet equals to 40 tokens. So each tweet is represented by a 45x32 matrix in the end. It is important to say that the maximum number of words in a sentence within the data is 33. Thus, we determined the maximum length to be a bit longer than this, like 40 (maxlen=40).

The next layer in the network performs convolutions over the ordered embedded word vectors in a tweet using multiple filter sizes. This is the equivalent of looking at all 3-grams, 4-grams and 5-grams in a sentence and will allow us to understand how words contribute to sentiment in the context of those around them.

So, we applied 1D convolutions that are useful for our text classification problem. The reason is that they can learn patterns and then recognize them at different positions in sequences. It could capture a negative phrase such as "don't like" regardless of where it happens in the tweet. The convolutions have width 32 and different height in order to learn patterns of different word length.

Firstly, we applied 64 filters with kernel size of 2, meaning we have a 2x32 filter matrix and “RELU” function for nonlinear transformation.

Looking at the summary of the model we can check the dimensions of each output layer.  The output width reflects the number of filters we apply, so the answer is we will have 40X64 dimension output. The results are shown on Figure 7

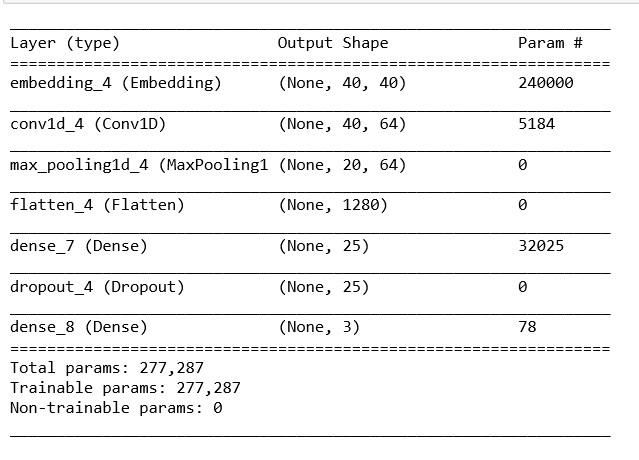


Figure 7 - Output Dimensions

Then, we added Max Pooling layer. The pooling layer will extract the maximum value from each one of 64 filters, and the output dimension will be a just 1-dimensional vector with length equals to 64.

Next, we add dropout regularization, which will randomly disable a fraction of neurons in the layer to ensure that that model does not overfit. This prevents neurons from co-adapting and forces them to learn individually useful features. In our case set it equal to 20%.

The output from global max pooling layer fed to a fully connected layer. At this layer there are 25 neurons and “relu” function to verify the nonlinearity of the model.

Finally, classified the resulting output of this layer using the “softmax” function. The output layer consists of 3 neurons and yields a result among 0 (negative sentiment), 1(neutral sentiment) and 2(positive sentiment).

Compiling the model

Next, we need to compile our model. Compiling the model takes three parameters: optimizer, loss and metrics.

The optimizer controls the learning rate. We used ‘adam’ as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

We also used ‘categorical\_crossentropy’ for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better.

To make things even easier to interpret, we used the ‘accuracy’ metric to see the accuracy score on the validation set when we train the model.

Training the model

Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters: training data (X\_train), target data (encoded\_Y), validation data, and the number of epochs. The results are shown on Figure 8.

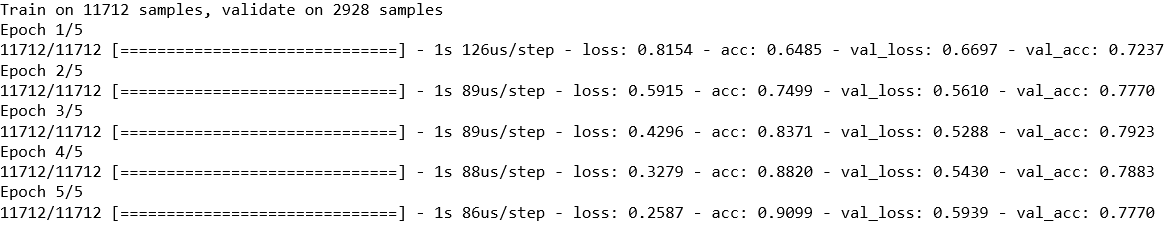


Figure 8 - Validation accuracy and loss

For our validation data, we will use the test set provided to us in our dataset, which we have split into X\_test and encoded\_Y\_test.

The number of epochs is the number of times the model will cycle through the data. As we can see from the results above, the best validation accuracy is 79.23% and epochs equal to 3. So, we can reduce the epochs to 3 in order to save calculation time.

Finally, the batch size is equal to 100. It means that tweets pass through network in smaller species of 100 tweets each time.

Plot loss function and accuracy

As we can see from the plots below, Figure 9 and 10 the loss function at validation dataset increases slightly after the epoch 3. At the same time, after the epoch 3 remains the same. So, we can conclude that epoch equal to 3 is a good value for our model.

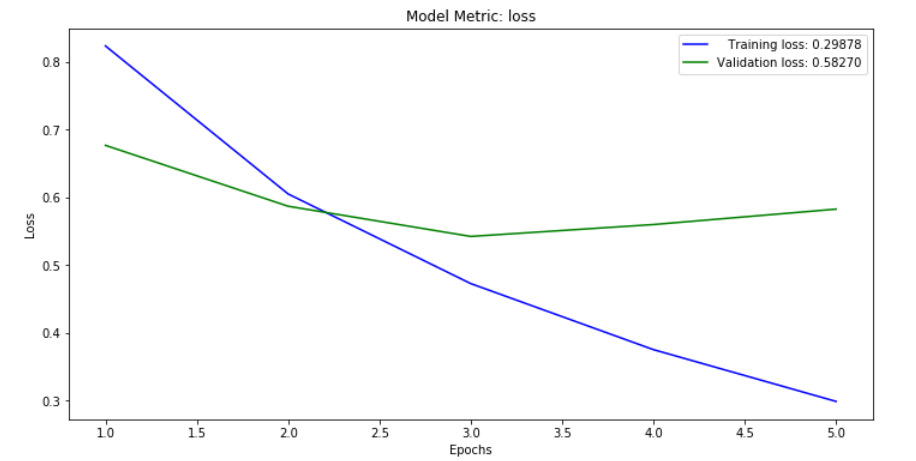


Figure 9 - Loss function for CNN model

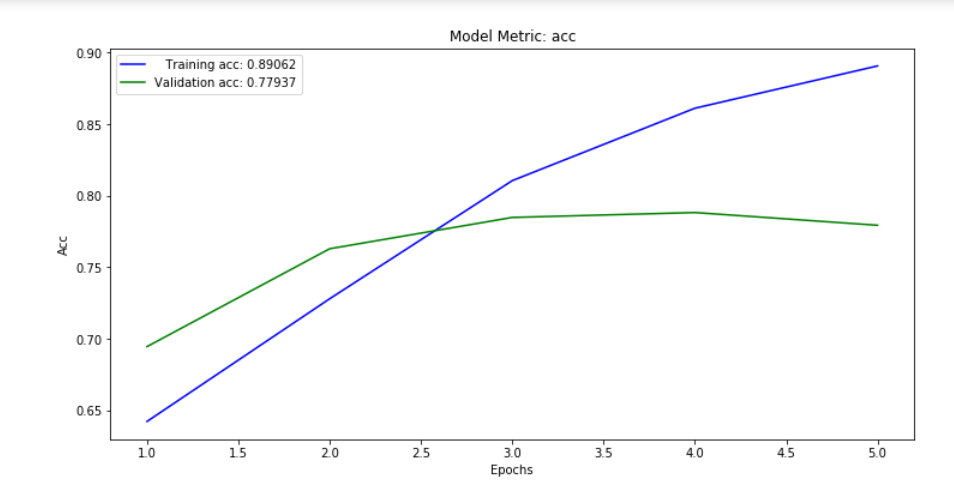


Figure 10 - Accuracy for CNN

Using our model to make predictions

At the final stage, we used the predict () function to make predictions using the test dataset (X\_test). The predict function will give an array with 3 numbers. These numbers are the probabilities that the input tweet represents each sentiment. The array index with the highest number represents the model prediction. The sum of each array equals 1 (since each number is a probability).

In conclusion, the accuracy of CNN model is equal to 77.9%. It is a good value, but we can try for a better value.

Improving CNN

Taking into account the above insights we created a new CNN model with the same number of inputs, output dimension and maximum length. But we applied 128 filters of kernel size 3 and run it for 3 epochs.

Eventually, the accuracy on validation dataset is smaller in this case. So, we remain with the initial CNN model.

### Our Second Model: Recurrent Neural Networks (RNNs)

One of the typical uses of the recurrent neural networks, is on sentiment analysis. There was really no point in trying to challenge this fact, so this was the neural network we chose to construct our model. The reason RNNs is essentially a textbook case when someone is dealing with a sentiment analysis problem is, that this is a problem of sequence prediction. And RNNs are designed in such way that they take under consideration the previous observations, unlike for example conventional feed-forward neural networks. Along with the improvement provided by LSTM networks which offer a mechanism to remember or forget things, they are the state of art solution in handling sentiment analysis problems.

Our first layer is an embedding layer. This is a transformation layer used to turn the indexes we provide into dense vectors of fixed size [11]. Word embeddings is actually a very popular technique to represent(embed) words in a continuous vector space where semantically similar words are mapped to nearby points [12]. The embedding space we chose had 96 dimensions, because, as can be seen by the diagrams below, this value gave us the better results. There is not any rule for choosing some specific dimension value and this value does not appear to have a significant effect on the accuracy of our model.

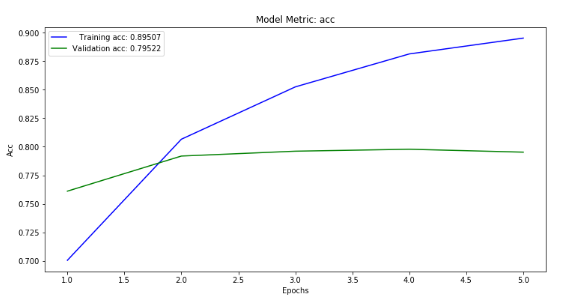


Figure 11: Embedding space of 64 Dimensions

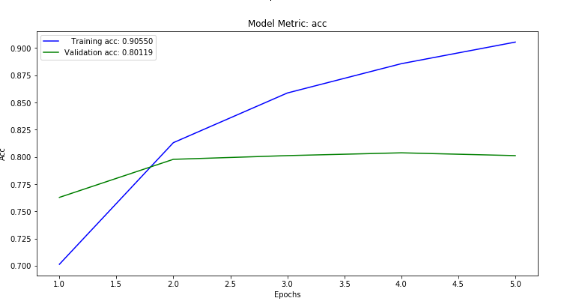


Figure 12: Embedding space of 96 Dimensions

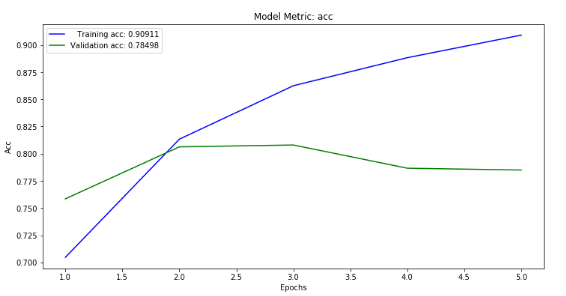


Figure 13: Embedding space of 128 Dimensions

The next layer is an LSTM (Long Short-Term Memory) layer with 96 memory units and some dropout values to avoid overfitting. We used LSTM as a special form of RNNs, as they are especially powerful when it comes to finding the right features when the chain of input-chunks becomes longer. In Keras the LSTMs neural networks are implemented by adding the relative layer, which we have.

RNNs are powerful models but overfit quickly [13]. In order to avoid overfitting we should accompany our LSTM layer by a Dropout layer. This will ignore randomly selected neurons during the training phase, therefore reducing the sensitivity to specific weights. Keras provides us with the capability to incorporate this layer into the input or the output units of the LSTM or even at the connections between the units of the LSTM, by using “recurrent dropout” option. After many trials we decided to apply a dropout of 20% to the inputs and connections and not on the outputs which would require adding one more output layer.

Finally, we added a Dense layer which we used to change the dimensions of our vector from 96 to just 3, as many as our label categories are.

### Model Input

In order to bring our data into a proper form to feed them in our model, we pass them through a tokenizing procedure. This consists of choosing the 2000 most used words and building an internal dictionary that contains all of them by assigning to each one an index. Next, we replace each word with its index. We have now transformed the sequences of our words into sequences of integers. Next, as our sequences need to have the same length we add some zeros at the left of each one as padding and the length of all of them becomes the length of the largest one. Finally, we split our data into training, validation and testing ones. The ratio is **0.68 : 0.07 : 0.25.**

### Loss Function, Evaluation Measures, Optimizers

We are essentially dealing with a multi-class classification problem, so the loss functions among which we can choose are [14]:

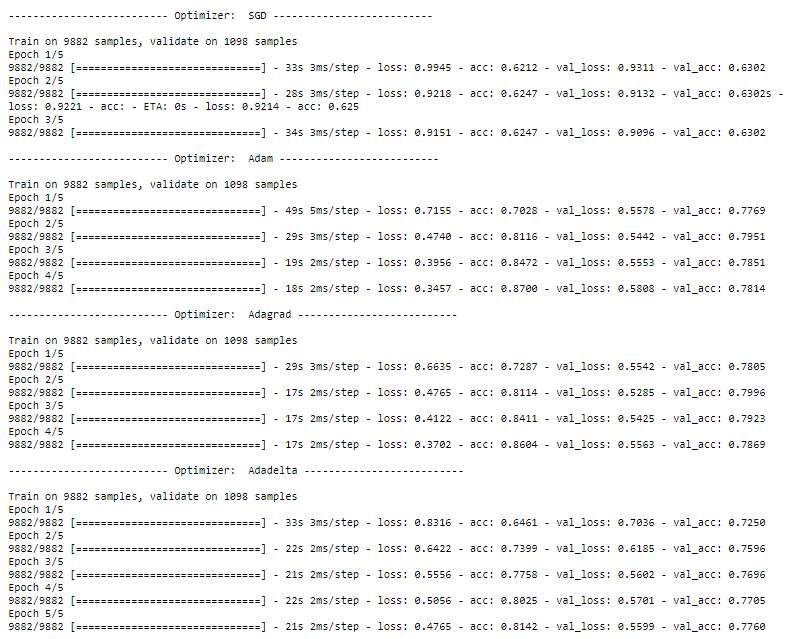
* Multi-Class cross entropy (Keras: categorical\_crossentropy)
* Sparse Multiclass Cross entropy (Keras: *sparse\_categorical\_crossentropy)*
* Kullback Leibler Divergence Loss (Keras: *kullback\_leibler\_divergence)*

Cross entropy is the default loss function to use for multi-class classification problems which we eventually used during our model’s calculations.

We would use sparse multiclass cross entropy in case we had a large number of labels. In our problem the labels are only three so there is no need to use this loss function. With regards to Kullback Leibler Divergence Loss function, it is a way to measure the difference between the two distribution probabilities. In case of a multi-class classification problem, it is functionally equivalent to multi-class cross-entropy. So, we have no reason to avoid using the default loss function, which is cross entropy.

The same loss function we chose to use as the metrics parameter. We input ‘accuracy’ which makes Keras automatically infer the appropriate accuracy function out of the loss function we have used.

Finally, in order to choose the optimizer, we tried most of the available optimizers in Keras and from the results, which can be seen below, we chose the one with the best ones. This is the Adagrad optimizer which is the only one that gave us accuracy in the validation dataset just below 80%.



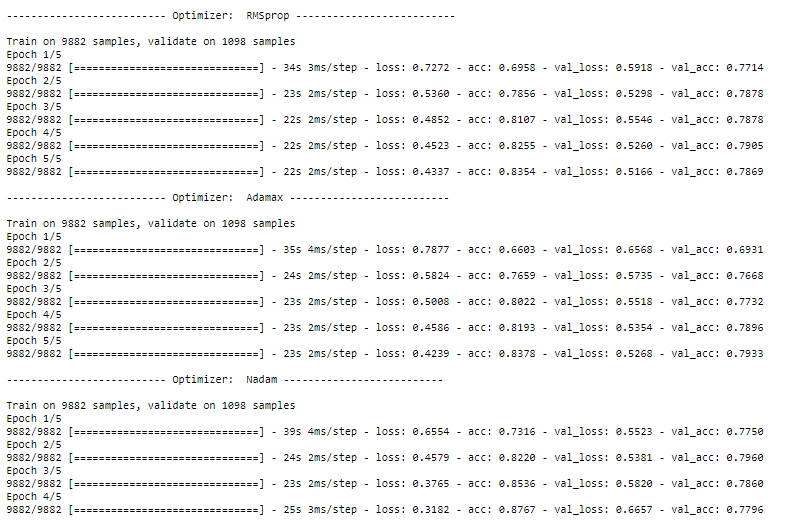


Figure 14 – Trainings to define the best optimizer in RNN

## Results

We can see below the learning curves of our model. We used 10 epochs but posed a trigger when two consecutive validation accuracy values are below the maximum, up to that point, value, then the procedure stops.

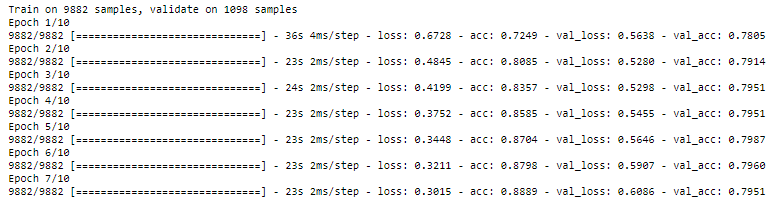


Figure 15 - Training of our model

The maximum value we get on the validation set is 79.87%. We can see it more clearly below:

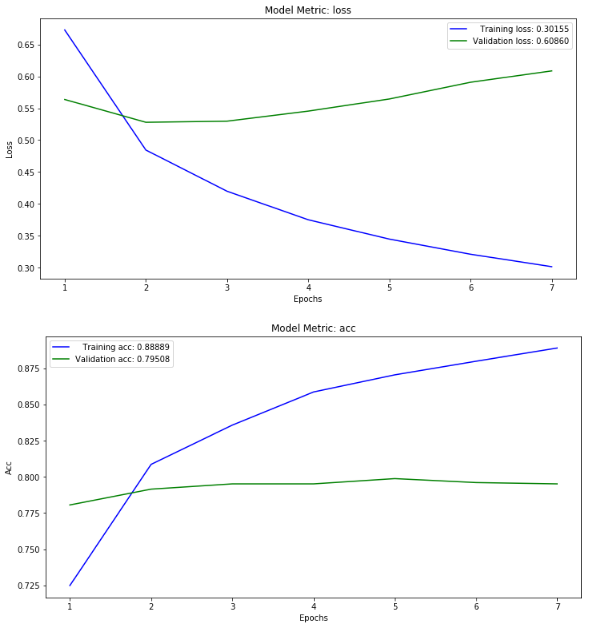


Figure 16 - Learning Curves of our RNN model

From epoch 5 onward we have the problem of overfitting which gives us lower correct predictions for our validation set in each consecutive epoch.

The time required for our model to train was 175 sec on an Intel Core i3 2100 machine with 12 GBytes of RAM.

1. MEMBERS/ROLES

Our group consists of 3 members: Gialama Niovi, Dagre Adriani and Kondyli Afroditi. We tried to divide the needs for this project into three major categories:

1) Developers and Software engineering

Kondyli Afroditi: Graduate student of Department of Physics at University of Athens. Further specialization in Meteorology (MSc UOA) and in Business Analytics (MSc AUEB) sector, granted from different departments. Currently, working as AI Specialist at Accenture Company.

2) Statistical Analysis and Models Implementation

Dagre Adriani: Graduate student of University of Piraeus with specialization in Economics, granted from the department. Currently, studied Business Analytics (MSc AUEB) and working as Business Analyst at Deloitte Company.

3) Business Development and Project Manager

Gialama Niovi: Graduate student of Department of Physics at University of Crete. Further specialization in Microsystems and Nanodevices (MSc NTUA) and in Business Analytics (MSc AUEB) sector, granted from different departments. Currently, working as Management Consultant at ICAP Company.

# Bibliography

|  |  |
| --- | --- |
| [1] | E. Cambria, D. Das, S. Bandyopadhyay and A. Feraco, A Practical Guide to Sentiment Analysis, Springer. |
| [2] | E. M. E. F. F. A. P. Bing Liu, Sentiment Analysis in Social Networks, 2016. |
| [3] | V. S and J. Teraiya, "A comparative study of sentiment analysis techniques," *Journal JIKRCE, 2(2), 313-317,* 2013. |
| [4] | P. B., L. Lee and S. Vaithyanathan, "Thumbs up?:sentiment classification using machine learning techniques," in *ACL-02 conference on Empirical methods in natural language processing*, 2002. |
| [5] | A. A., H. Chen and A. Salem, "Sentiment Analysis in multiple languages: Feature selection for opinion classification in web forums," *ACM Transactions on Information Systems(TOIS), 26(3), 12,* 2008. |
| [6] | T. P.D., "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews.," in *40th annual meeting on association for computational linguistics(pp.417-424). Association for Computational Linguistics*, 2002, July. |
| [7] | M. H. &. B. Liu, "Mining and summarizing customer reviews," in *Tenth ACM SIGKDD international conference on Knowledge*, 2004, August. |
| [8] | A. M. and G. Kondrak, "A comparison of sentiment analysis techniques: Polarizing movie blogs.," in *Canadian Society for Computational Studies of Intelligence*, Berlin, Heidelberg, 2008. |
| [9] | Y. Goldberg, "Neural network methods for natural language processing.," *Synthesis Lectures on Human Language Technologies,* pp. 10(1), 1-399, 2017. |
| [10] | C. C. Aggarwal, Machine Learning for Text, Springer, 2018. |
| [11] | "www.keras.io," [Online]. Available: https://keras.io/layers/embeddings/. |
| [12] | "www.tensorflow.org," [Online]. Available: https://www.tensorflow.org/tutorials/representation/word2vec. |
| [13] | Y. Gal and Zoubin Ghahramani, "A Theoretically Grounded Application of Droupout in Recurrent Neural Networks". |
| [14] | J. Brownlee, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/. |
| [15] | "www.figure-eight.com," [Online]. Available: https://www.figure-eight.com/data-for-everyone/. |