BABEȘ-BOLYAI UNIVERSITY, CLUJ-NAPOCA
FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION
BUSINESS MODELING AND DISTRIBUTED COMPUTING

Methods in Data Science

Project Report

Sentiment Analysis in a Social Network

Advisor(s): Prof.univ.dr. Gheorghe Cosmin Silaghi
Student(s): Valentina-Ștefania Mașniță

CLUJ, JULY 2023, JULY 2023

Contents

List of Figures

1 Introduction

1.1 General context

The motivation for this domain choice relies on personal interests in the worldwide social network bubble, with one remarkable element being the fascinating interaction between human being in a virtual context. This inherent nature is best illustrated with a quote from Paul Bloom (award-winning psychologist): "Humans are social beings, and we are happier and better when connected to others".

Therefore, the connections people get to establish in a virtual environment and the liberty they have in expressing themselves (and their thoughts/opinions) in this community are aspects that highly affect and are affected by emotions. As social networks have gained immense relevance in this century and as they have become integral to the way we communicate, connect, and share information, there has emerged a fundamental need (especially relevant to network businesses) for their technologies to better understand us. Taking these into consideration, a question that arises related to this topic is **"How well can we predict the sentiments of a social network platform's users over certain subjects or products?"**. One specific application for this research question is predicting Twitter users' sentiments over various topics.

Determining user sentiments in a social network allows businesses to gain a deeper understanding of their target audience, optimize their strategies to meet customer needs and preferences, and make data-driven decisions in favor of their brand image and reputation. One other application inspired by this scope is the Social Machine Learning with H2O, Twitter, Python, created by Marios Michailidis[6], who tackles the usefulness of machine learning in urgency situations. His idea was to inspect posts from Twitter, build a classifier that differentiates between negative and positive tweets, assign them severity levels, and retrieve the ones where people are asking for help or need urgent care (one example being tweets created during the Texas hurricane).

1.2 Technologies

Sentiment Analysis is one of the most common text classification tools that analyzes incoming messages and outputs whether the underlying sentiment is positive, negative, or neutral. It represents the contextual mining of text, which first identifies and then extracts and quantifies subjective information in source material. The steps for this process start with gathering the text input, stemming, tokenization, classification, stop-word filtering, finding the sentiment class, and handling negation.[1]

Natural Language Processing (NLP) targets the branch of Artificial Intelligence (AI) concerned with computers' ability to understand text or spoken words in a natural way — the same way human beings can. NLP employs computational linguistics (rule-based modeling of human languages), statistical models, machine learning, and deep learning models and enables computers to process human language in the form of text/voice, along with the user's intent or sentiment.[2]

Deep Learning models - Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) - can be used to assist/perform sentiment analysis on social media posts and user comments over various subjects, as they learn the contextual information and then capture the complex relationships within the text - leading to sentiment classification of higher accuracy.

Long Short-Term Memory (LSTM) Networks represent a variety of Recurrent Neural Networks (RNNs), capable of learning long-term dependencies and used especially in sequence prediction problems. LSTM offers feedback connections that can process the entire sequence of data, apart from single data points such as images, with outstanding performances on a large collection of problems.[8]

The implementation environment for this project consists of Anaconda, Jupyter Notebook & Kaggle Notebook, running Python programs, and containing the necessary libraries: tensorflow, matplotlib, pandas, re, numpy, nltk, and sklearn.

2 Data

The dataset used for this project is the Sentiment140 dataset with 1.6 million tweets[3] retrieved from Kaggle. This will be used to train the model, as it's a vast dataset (containing information extracted from the Twitter API) that already presents annotations for the tweets. Sentiment140 contains six fields:

1. target – suggesting the annotations, more specifically the polarity of tweets, which are marked with some numbers between 0 and 4; 0 represents negative sentiments, 2 represents neutral ones, and 4 represents positive ones.

2. ids – marking the tweet id, used for access and identification.

3. date – representing the day a specific tweet was created.

4. flag – containing the used query or the text NO_QUERY.

5. user – meaning the author of the tweet as its Twitter username.

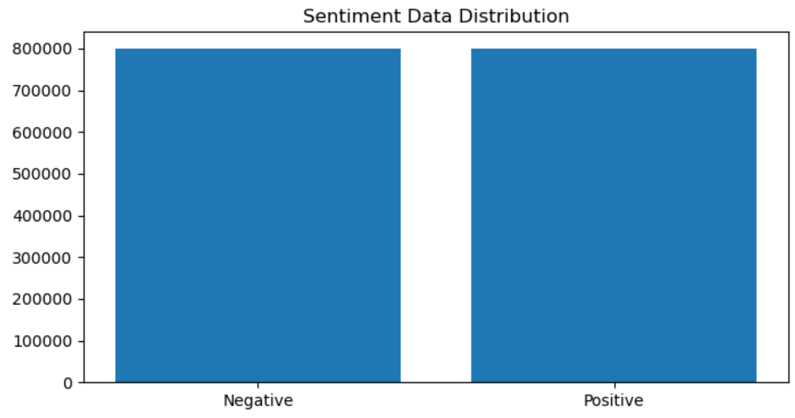


Figure 2.1: Sentiment data distribution

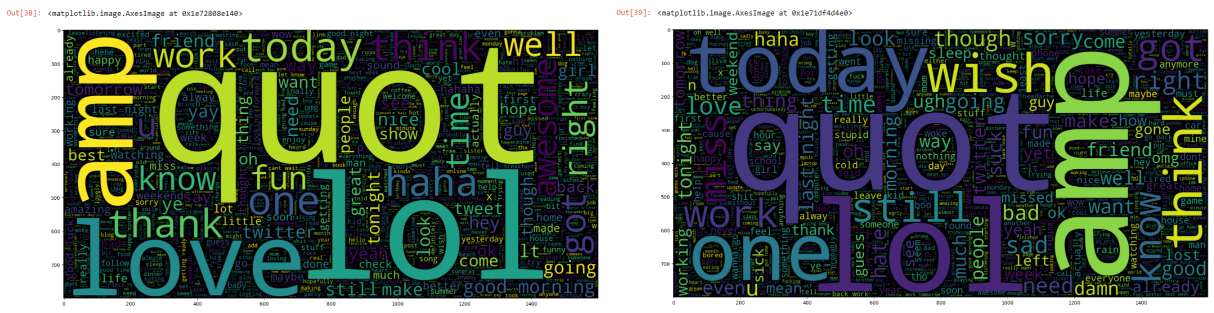
6. text – containing the actual information stored inside a tweet, the actual post.

For the scope of this application, the only needed field is target, which will be renamed accordingly, and the other fields will be dropped. Next, the labels have to be decoded in order to describe the information and its distribution, and we map 0 to negative and 1 to positive. The sentiment data distribution can be seen in Fig. 2.1.

By exploring the data and visualizing it, one decision is certain: there's a need for data clean-up. Tweet contents include other user tags, hyperlinks, emojis, punctuation, and other no-value information, which should be removed with text preprocessing (stemming, lemmatization, handling stop words) in order to result in good models.

Stemming is the practice of removing the last few characters from a word, which often results in inaccurate interpretations and spelling. Lemmatization takes the context into account and transforms the term to its meaningful base form, which is known as a Lemma. Their goal is to reduce inflectional and derivational forms of words. Stopwords are contextually meaningless words - they do not have a significant contribution to the text. After employing the re and nltk libraries, the dataset is considered cleaned.

Cloud visualization (using the wordcloud library) allows for a graphical representation of the words' frequency in the given tweets, highlighting the most common positive and negative words in Fig. 2.2.



```
Out[38]: <matplotlib.image.AxesImage at 0x1e72808e140>
Out[39]: <matplotlib.image.AxesImage at 0x1e71df4d4e0>
```

Figure 2.2: Cloud visualization of positive and negative words

3 Results and Discussion

3.1 Splitting the data and preparing the model

The dataset is divided into two subsets: training (used to train the model) and testing (used to evaluate the performance of the trained model). The resulting numbers are Train Data size: 1280000 and Test Data size 320000. Training involves feeding the model with input features and their corresponding labels (for supervised learning) or input data only (for unsupervised learning), thus, it learns to make predictions and discover patterns by adjusting its internal parameters.

The goal of the practical part is to build a sequence model, as it's particularly designed to handle sequential data (such as text) and has the ability to capture relationships and dependencies between elements. Tokenization is used to split the texts into smaller units, called tokens, with assigned meanings. In this case, the process gathers a tweet and breaks it into understandable words, mapping them to an index using dictionaries, and resulting in a vocabulary.

Another constraint added to the system (through the pad_sequence function) is the consistent shape of the sequence inputs. The next steps are label encoding and word embedding. Word embedding is a technique (used in Natural Language Processing) for representing the meaning and semantic relationships of words in a numerical format that can be understood by machine learning algorithms. Therefore, words are seen as dense vectors in a high-dimensional space, and similar meanings are mapped to nearby points.

The chosen solution is Stanford University's GloVe (Global Vectors for Word Representation), which is an unsupervised learning algorithm that captures aggregated global word-word co-occurrence statistics from a corpus of text in the form of a matrix[5]. The project employs Transfer Learning, as GloVe's pre-trained embedding is downloaded and used in our model.

3.2 Deep Learning model training

When building deep learning models, there are several key considerations to check:

1. Data, as Deep Learning models require large amounts of properly labeled data for training – the cleaned Sentiment140 dataset.
2. Model architecture – Sequence Model, suitable for the task and capable of handling complexities and patterns in the data.
3. Hyperparameter tuning and regularization, which can significantly impact the model's training speed and final accuracy – number of epochs, the Adam optimization algorithm for Gradient Descent.
4. Evaluation and validation, using appropriate metrics to assess the model's performance - learning curve of loss and accuracy, confusion matrix, classification scores.

The model architecture displayed in Fig. 3.1. consists of several layers, each handling a part of the process:

1. Embedding layer – generates the embedding vector with GloVe for each input sequence.
2. Conv1D layer – combine/divide data into proper feature vectors.
3. LSTM layer – learns the context of words.
4. Dense layer (fully connected layer) – captures intricate patterns in the data and is employed in the classification process.

Adam optimization[4] represents a computationally efficient stochastic gradient descent method based on adaptive estimation of first and second-order moments, which inherits the kera.Optimizer class. The project employs callbacks in order to monitor, customize and control the behavior of the training process dynamically at certain moments over an epoch. They are powerful tools for tracking important metrics, improving training efficiency, and preventing overfitting. The used callbacks are:

1. LRScheduler – controlling the learning rate at a specific epoch.
2. ModelCheckPoint– saving the model with minimum validity loss.

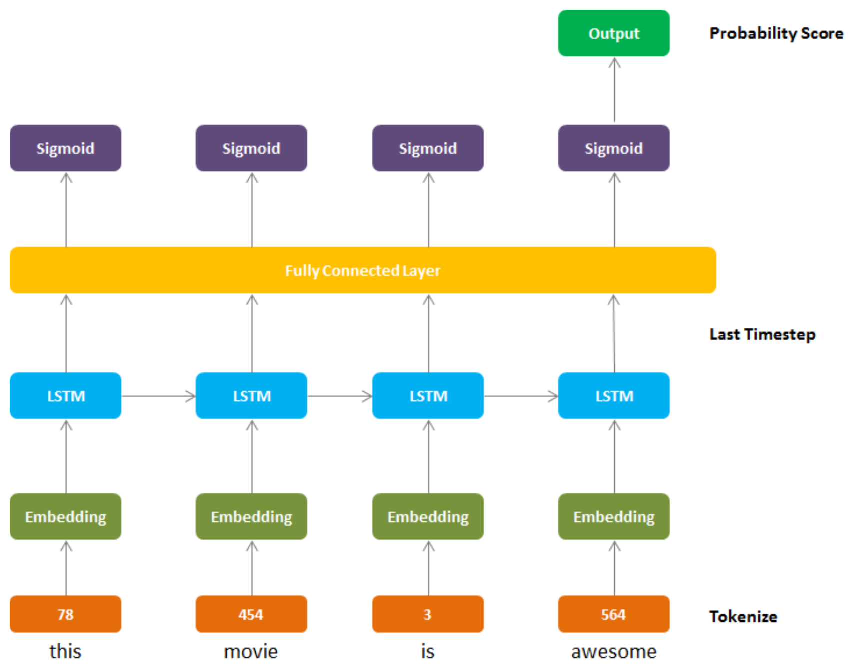


Figure 3.1: Model architecture[7]

3.3 Training results

The training is done over 1280000 samples, the validation over 320000, and the process consists of 10 epochs, which mark the models' characteristics: loss, accuracy, validity loss, validity accuracy. These aspects cam be noted in Figure 3.2.

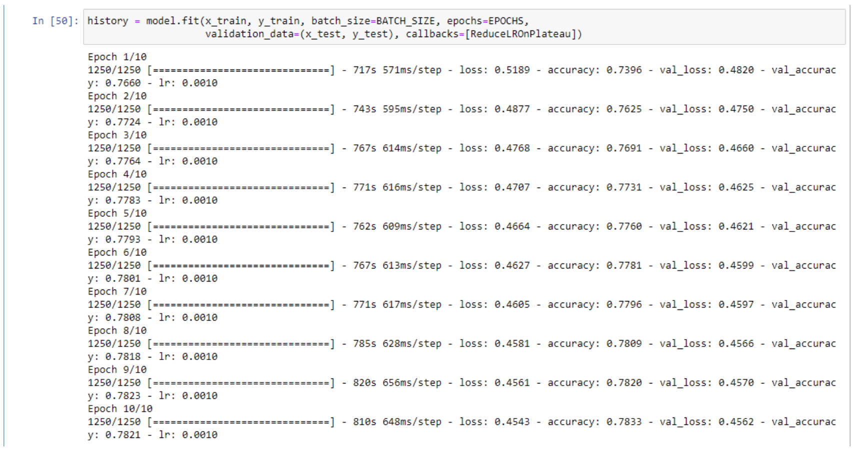
3.4 Model evaluation

Evaluation of performances starts by setting specific evaluation metrics and testing techniques.

The learning curve displays the model accuracy and model loss at each epoch, as in Fig. 3.3. A threshold with a value of 0.5 is defined to differentiate between the positive and negative classes, as the model outputs a prediction score for these characteristics between 0 and 1.

The confusion matrix from Fig. 3.4. provides an overview of the model's performance in the classification task by comparing the true label with the predicted label.

A simple visualization of the classification scores is displayed in list form in Fig 3.5. A deduction made after evaluating these numbers is that the accuracy of the classification is around 78%. The limitations of these results are caused by possible inconsistencies in the dataset that were not handled and the performances of the chosen algorithms.



```
In [50]: history = model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS,
                             validation_data=(x_test, y_test), callbacks=[ReduceLROnPlateau])

         Epoch 1/10
         1250/1250 [==============================] - 717s 571ms/step - loss: 0.5189 - accuracy: 0.7396 - val_loss: 0.4820 - val_accurac
         y: 0.7660 - lr: 0.0010
         Epoch 2/10
         1250/1250 [==============================] - 743s 595ms/step - loss: 0.4877 - accuracy: 0.7625 - val_loss: 0.4750 - val_accurac
         y: 0.7724 - lr: 0.0010
         Epoch 3/10
         1250/1250 [==============================] - 767s 614ms/step - loss: 0.4768 - accuracy: 0.7691 - val_loss: 0.4660 - val_accurac
         y: 0.7764 - lr: 0.0010
         Epoch 4/10
         1250/1250 [==============================] - 771s 616ms/step - loss: 0.4707 - accuracy: 0.7731 - val_loss: 0.4625 - val_accurac
         y: 0.7783 - lr: 0.0010
         Epoch 5/10
         1250/1250 [==============================] - 762s 609ms/step - loss: 0.4664 - accuracy: 0.7760 - val_loss: 0.4621 - val_accurac
         y: 0.7793 - lr: 0.0010
         Epoch 6/10
         1250/1250 [==============================] - 767s 613ms/step - loss: 0.4627 - accuracy: 0.7781 - val_loss: 0.4599 - val_accurac
         y: 0.7801 - lr: 0.0010
         Epoch 7/10
         1250/1250 [==============================] - 771s 617ms/step - loss: 0.4605 - accuracy: 0.7796 - val_loss: 0.4597 - val_accurac
         y: 0.7808 - lr: 0.0010
         Epoch 8/10
         1250/1250 [==============================] - 785s 628ms/step - loss: 0.4581 - accuracy: 0.7809 - val_loss: 0.4566 - val_accurac
         y: 0.7818 - lr: 0.0010
         Epoch 9/10
         1250/1250 [==============================] - 820s 656ms/step - loss: 0.4561 - accuracy: 0.7820 - val_loss: 0.4570 - val_accurac
         y: 0.7823 - lr: 0.0010
         Epoch 10/10
         1250/1250 [==============================] - 810s 648ms/step - loss: 0.4543 - accuracy: 0.7833 - val_loss: 0.4562 - val_accurac
         y: 0.7821 - lr: 0.0010
```

Figure 3.2: History

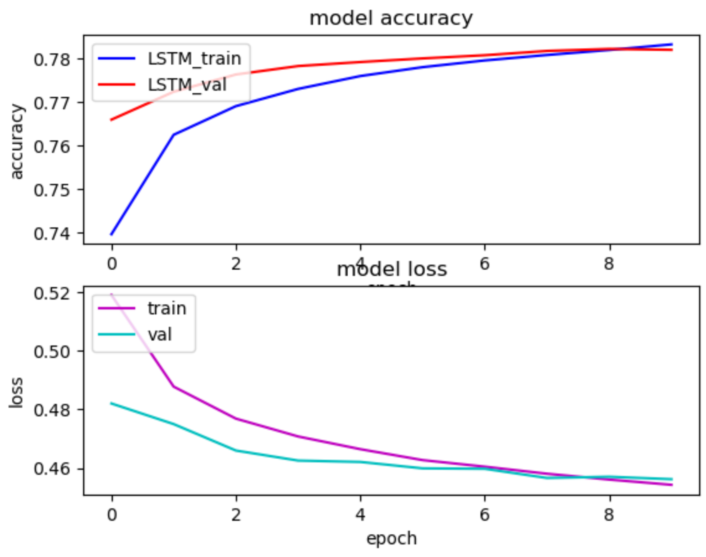


Figure 3.3: Learning curve

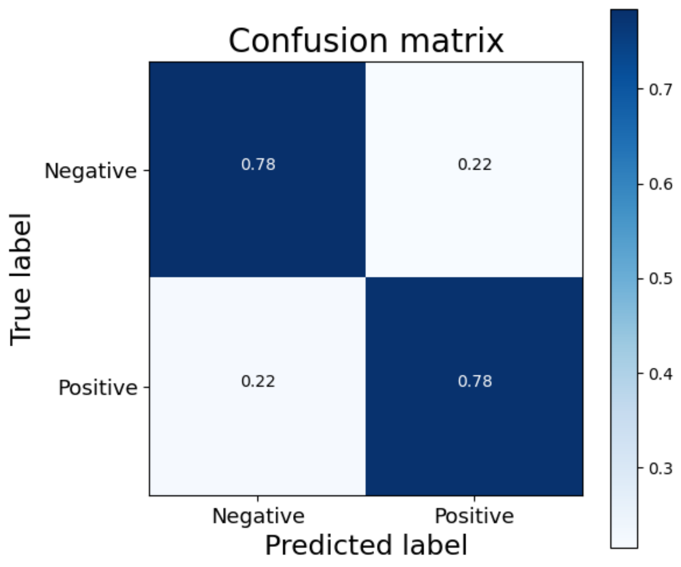
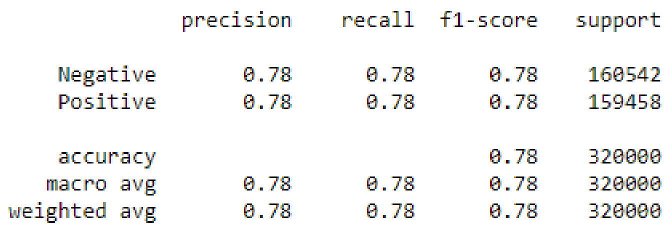


Figure 3.4: Confusion matrix



```
              precision    recall  f1-score   support

    Negative       0.78      0.78      0.78    160542
    Positive       0.78      0.78      0.78    159458

    accuracy                           0.78    320000
   macro avg       0.78      0.78      0.78    320000
weighted avg       0.78      0.78      0.78    320000
```

Figure 3.5: Classification scores

4 Conclusions

To answer the research question "How well can we predict the sentiments of a social network platform's users over certain subjects or products", a Natural Language Processing application was conceived using the Sentiment140 dataset, python code, and specific additional algorithms and libraries (GloVo, Adam, Python libraries: tensorflow, matplotlib, pandas, numpy, re, nltk, sklearn). The Deep Learning model constructed and employed in the training and testing of data was a sequence model, being the best choice for handling large amounts of text information.

The characteristics that support the answer to the aforementioned question are the models' accuracy and validity loss in label classification and prediction. The final model is represented by the model with the minimum validity loss, which demonstrates an accuracy of 78%, a decent result when it comes to sentiment analysis.

References

[1] Karsten Hamborg, Felix; Donnay. *NewsMTSC: A Dataset for (Multi-)Target-dependent Sentiment Classification in Political News Articles.* Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, 2021.

[2] IBM. What is natural language processing (nlp)? *https://www.ibm.com/topics/natural-language-processing*, 2023.

[3] KazAnova. Sentiment140 dataset with 1.6 million tweets. *https://www.kaggle.com/datasets/kazanova/sentiment140*, 2017.

[4] Keras. $https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam$, 2023.

[5] Jeffrey Pennington; Richard Socher; Christopher D. Manning. Glove: Global vectors for word representation. *https://nlp.stanford.edu/projects/glove/*, 2014.

[6] Marios Michailidis. Social machine learning with h2o, twitter, python. *https://www.linkedin.com/pulse/social-machine-learning-h2o-twitter-python-marios-michailidis/*, 2017.

[7] Arun Pandian R. Nlp beginner - text classification using lstm. *https://miro.medium.com/v2/resize:fit:1458/1*SICYykT7ybua1gVJDNlajw.png*, 2014.

[8] Sepp Hochreiter; Jürgen Schmidhuber. *Long short-term memory.* Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014, 1997.