TEXT SENTIMENT ANALYSIS USING DEEP CONVOLUTIONAL NETWORKS

Author(s): **Klevis Lika [a] , Evis Plaku [b]**

[a,b] Canadian Institute of Technology, Xhanfize Keko Street, Nr. 12, Tirana, Albania

Abstract

The ability to process language is a difficult task. A computer system that can extract meaningful information from raw text data is considered to be equipped with intelligent behaviour. Sentiment analysis is a form of information extraction of growing research and commercial interest, especially because of the exponential increase of data in recent years.

This paper presents a machine learning approach built upon Neural Networks that equips a computer system with the ability to learn how to detect sentiment in a given text and to correctly classify previously unencountered text with respect to predefined sentiment categories. The developed approach does not rely on any hard-coded parameters. Rather, a model is learned through a training procedure performed on raw data alone with the objective of searching to identify patterns and discovering feature predictors that allow the algorithm to classify text into positive, negative, or neutral sentiments, while additionally focusing on nuances such as humorous or sarcastic text.

Experimental validation shows promising results. The learning algorithm is fed with high-volume textual data extracted from the Twitter social media platform. The trained model is then tested on a separate dataset, previously unknown by the model. The classification algorithm is able to predict the sentiment associated with a given text with high accuracy.

Keywords: *text-sentiment analysis, deep learning, machine learning, convolutional neural networks*

1. Introduction

Sentiment analysis is often defined as the study of people's opinions, sentiments, or emotions through the aid of computer agents with the objective of categorizing the writer's attitude towards a particular topic. Sentiment analysis has grown into becoming one of the most active research fields in natural language processing (NLP), as it has wide applications in many different areas such as: data mining, information retrieval, product segmentation and analysis, brand monitoring and market research, among others.

Strong opinionated posts in social medias, blogs, forums, and other mediums of communication have profoundly impacted individuals and organizations, and often, have highly influenced and reshaped behaviour.

Extensive research is focused on developing intelligent computer systems that have the ability of processing large amounts of text data and categorizing the sentiment expressed as either positive, negative, or neutral.
A plethora of machine learning approaches are developed with the objective of training a computer agent in a framework that processes high-volume data with the purpose of learning to efficiently classify textual data that the machine learning framework has not encountered before.

This task poses many difficulties. Though understanding the context of a conversation is easy for humans, teaching a computer agent to achieve the same results presents many challenges, mainly associated with the various nuances of language. Grammar rules are not strict; they leave room for many possibilities of forming a sentence, or interpretations.

In addition, written text, especially on social media and forums, contains a lot of unstructured sentences, abbreviated forms, grammar nuances, and slangs which contribute to increasing the difficulty for a computer agent to understand text. Moreover, understanding sarcastic or humorous opinions expressed in a written form – without the help of voice tone or body gestures – is difficult for a computer.

To address these challenges, we have developed a machine learning framework that builds upon Convolutional Neural Networks (CNN) with the goal of equipping a computer agent with the ability to predict and classify written text according to the sentiment it expresses. The model for analysing text is not hard-coded, i.e., it is not depended on any rigid tuning of parameters. Rather, the proposed framework learns how to detect sentiment by processing high-volume raw data and searching to identify patterns and feature predictors that allow it to classify text into positive, negative, or neutral sentiments, while additionally focusing on nuances such as humorous or sarcastic text.

The learning algorithm is fed with high-volume textual data extracted from the Twitter social media platform. Twitter feeds are commonly used by individuals and organizations to express opinions.

The proposed model has two main phases. During the training procedure, the learning algorithm receives as input labelled data, i.e., twitter feeds with a categorization of sentiment. The objective is to investigate and identify hidden patterns in the data or features that can be accurate descriptors of the sentiment categorization. To achieve that, the convolutional neural network processes and transforms the input via several layers, each reinforcing a mapping from the given input to the desired output (class categorization). At each step, information back-propagates, allowing the network to adjust the weights that affect its performance, thus improving its performance as more data comes in, or stronger relations are identified in the existing data.

The second phase includes cross validating the accuracy of the learned model. For that purpose, the algorithm is tested on a separate dataset that includes only twitter feeds, but no labelled data. As discussed in more details in section 4, experimental validation demonstrates the effectiveness of this approach. Our framework is able to efficiently classify and categorize the sentiment associated with textual data that is has not encountered before.

The rest of this paper is organized as follows. Section 2 provides a discussion of fundamental background principles of machine learning and convolutional neural networks, in addition to presenting closely related work. The proposed approach is discussed in section 3, while experimental results are introduced in section 4. Conclusions and future work directions are discussed in section 5.

2. Background and Review of Literature

This section provides an overview of fundamental concepts, methodologies and approaches that focus on the task of efficiently using machine learning algorithms for text sentiment analysis.

2.1 Problem Definition

A machine learning task is often defined as: *"a computer program learns from experience E with respect to some class of task Tand performance measure P if its performance at tasks T measured by P improves with experience E."* The computer agent aims to solve the task T,which is, in turn, the learning goal of the machine learning algorithm.

Each representation of the task can be considered as a data point, commonly described by a set of features. A vector $x \in R^N$ has entries x^i representing the set of features. In our example, each twitter feed represents a single data point, described by a set of features *x*. The task of the machine learning framework is to correctly categorize previously unseen textual data and to determine to which of the $k \in N$ categories it belongs (e.g., positive, negative, neutral). To solve this task, the objective of the learning algorithm is to produce a function $f{:}R_N \rightarrow \{1,\ldots,n\}$ where f outputs a classifier distribution, i.e., it assigns a probability related to the data entry *x* belonging to the category *k.*

The success rate of the learning algorithm is evaluated by the performance measure *P* – a metric designed and used to measure the learning performance, which can be specifically related to the task *T*. For instance, in the proposed text sentiment analysis framework, the performance measure is related to a function determining the accuracy of the algorithm in correctly (or falsely) classifying the testing data, as described in more details in section 4.

Finally, given task *T* and performance measure *P* experience *E* is comprised of the training data, the extracted features, and all other relevant additional information that the algorithm will use to learn and therefore to improve its performance. In our approach, we use a convolutional neural network, i.e., a model comprised of multiple learning layers that aims to extract meaningful information from training data and be able to correctly classify previously unobserved images. A detailed description is provided in section 3.

For the learning process to take place, given the performance metric P the algorithm attempts to identify the closest distance between two functions, say $f(x)$ and $f'(x)$ which are determined by a series of weighted values as in: $f'=w^T \cdot x$, where w is a vector of parameters that the algorithm is aiming to optimize, commonly known as weights. Weights determine how the extracted features x_i for $i \in 1 \ldots N$ correlate with the output f'. The machine learning algorithm will attempt to find the closest distance between f' and f, thus aiming to predict f from the given data x.

The overall objective is to identify parameters that not only will explain the given data set, but that will also be able to accurately classify sets of data that the algorithm has not observed before. To validate the training process, the algorithm is applied to an independent validation set, say x_{val} with n_{val} data entries that the algorithm does not use for training. The algorithm iterates the training and validation process until the error is sufficiently small, a criterion specific to the task.

2.2. Convolutional Neural Networks

Neural Networks emerged as a learning paradigm that aims to mimic the interconnections of neurons in the human brain. They are composed of several connected layers forming a network. Connections are primarily determined by weights iteratively tuned during the learning phase to equip the network with the ability to correctly classify previously unseen data.

To address our sentiment analysis classification tasks, the learning algorithm is used to identify a mapping function f which maps input x to category y. To achieve that, the algorithm will derive function f' defined by weights w_i that approximates $y=f'(x,w_i)$. In general, the function f can be a composite function of any number of functions f_i defined as: $f' = f_1' \cdot f_2' \cdot f_3' \cdot \ldots \cdot f_n'$.

This model is known as a network comprised of many layers, where f_1' is the first layer, f_2' is the second layer, and similarly, until f_n' is the last layer. The first and the last layer are commonly known as input and output layers, while the rest as intermediate hidden layers. The number of layers determines the depth of the network and is normally associated with the level of detail that is presented in the extracted features and served as an input to the learning algorithm. Figure 1 provides an illustration.

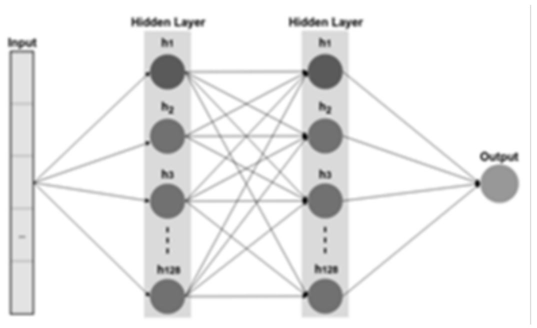

Figure 1: Representation of Layers in a Neural Network

The network will transmit messages from one layer to another through the network weights (Kuntal Kumar Pal, 2016). Each neuron is equipped with an activation potential which determines the ability of the neuron to decide whether a signal should be fired or not depending on the input signal that is received. The network then, propagates (moves forward) these signals through the whole network. The process is repeated until the classifier decides to which class the input belongs to.

The key objective of the training procedure is to minimize the difference between the expected output and the actual output (Kuntal Kumar Pal, 2016), thus minimizing the error and increasing the chance of correctly classifying input data. To achieve that, a loss function is defined and used when the training procedure takes place. This function continuously changes the weights of the network to adapt to the training data and to minimize the loss. Weights are updated because the network sends a back signal, i.e., a signal in the opposite direction (from output to input). This technique is commonly known as back-propagation.

Observe that not all neurons have the same influence over the network. Their role is dependent upon the weights they have. Strictly speaking, given *n* nodes, to compute the output as an activation function, the network sums the product of all the inputs $x_1, x_2, \dots x_n$ with their respective weights $w_1, w_2, \dots, w_n$. To model a statistically sound framework that takes into consideration unpredictable behavior, a *bias* (a value between 0 and 1) is added to each iteration step. Thus, the activation function of a neuron can be computed as:

$$f(b + \Sigma_{i=1}^{n} x_i \cdot w_i)$$

The bias *b* is an additional input to the neuron structure that makes possible to transition in the graph of the network, depending on the data that are being trained, thus enabling the neuron structure to have more flexibility in creating the required output value. An activation function determines if the neuron should be activated. It introduces non-linearity to the output of a neuron, thus making the overall structure of the network more adaptable and able to respond to complex inputs.

A Convolutional Neural Network is a special case of the generic network presented above. Figure 2 provides an illustration. It consists of at least one *convolutional* layer, in addition to other layers known as *subsampling* and *fully-connected* layers. Each feature of a layer receives inputs from previous layers and can extract visual features information that is used during the training phase. Note that the process of extracting the features is not hard-coded, rather the network updates the neuron weights and the transmission of messages between layers based upon the training data.

2.3. Sentiment Analysis Related Work

Sentiment analysis deals with the problem of extracting information and categorizing it to determine if the writer conveys a positive, negative, or neutral opinion. Extensive research is focused on analyzing text and estimating the sentiments associated with it. Most approaches fall into three main categories: 1) lexicon-based techniques, 2) machine learning techniques, and 3) hybrid approaches.

Lexicon-based techniques aimed to address the task of categorizing sentiment relying on a dictionary or corpus (Salas-Zárate, Medina-Moreira, Lagos-Ortiz, Luna-Aveiga, & Rodriguez-Garcia, 2017). Though statistical analysis is often performed through various clustering algorithms

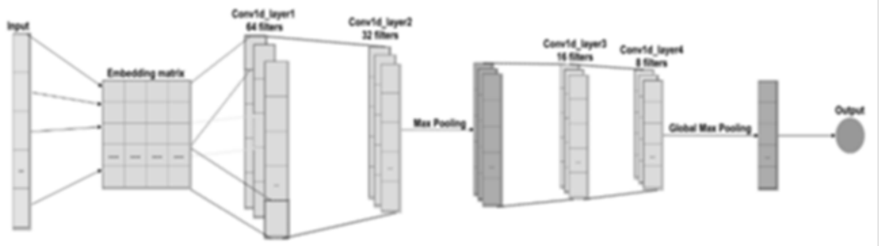

Figure 2: Convolutional Neural Network Architecture

(Huq, Ali, & Rahman, 2017), (Pinto, McCallum, Wei, & Croft, 2003), and (Soni & A, 2015), this family of approaches have not able to adequately adapt to different scenarios.

A wide range of machine learning algorithms have been developed to build flexible and efficient sentiment classifiers. Naïve Bayes classifiers (Zhang & Zheng, 2016), maximum-entropy classifiers (Malik & Kumar, 2018), or support vector machines (Firmino Alves, Baptista, Firmino, Oliveira, & Paiva, 2014) have been effectively used. However, these algorithms rely heavily on an explicit set of features related to text-description, such as lexicons, adjectives, adverbs, and grammar rules. The performance of the algorithms is greatly affected by the quality of features.

To overcome these challenges, deep learning models have emerged as an approach that does not explicitly models features. Rather, it aims to identify hidden patterns from the raw data alone and construct hyper-parameter models that make sense of the (often unknown) features.

For instance, Cai et.al have found that integrating together text and image data can allow for building models that more accurately classify sentiment expressed in tweets. (Cai, 2015). In their work, the authors propose an approach that is built upon the usage of convolutional neural networks and uses to individual architectures for learning textual features and visual features. The two, are then combined as a single input for another neural network that exploits and further investigates the relation between text and image to increase the accuracy of the classifier.

In other work, the task of generating an effective model that captures both the syntactic and the semantic relations between sentences in a text is achieved through the usage of deep convolution neural network (A. Chachra, 2017). The aim is to make better sentiment classifiers by automating the whole process, which otherwise would have been addressed by using natural language processing techniques. This approach analyzes text by considering its context and aiming to identify relations from word level to a phrase level to a document layer. Carefully taking these layers into consideration yields an effective classifier.

The idea of using an integration of neural networks for achieving a higher accuracy has been also successfully implemented by Tang et al. In their work, the authors use two neural networks: one for prediction and another one for ranking the predicted word (Tang, 2016). Their study used a dataset containing twitter feeds. The effectiveness of the approach was based on three levels: word, sentence and lexical.

Another approach was proposed by Chen et.al., to address the same problem (Chen, 2016). It uses the entire document as sentiment level information rather than dealing with local text information. To achieve this task, the proposed approach is built upon a hierarchical neural network modeled to generate sentence and document embedding.

In another work, the authors propose the Recursive Neural Tensor Network (RNTN) architecture, which represents a phrase through word vectors and a parse tree and then compute vectors for higher nodes in the tree using the same tensor-based composition function (Minh-Thang Luong, 2013).

3. Text Sentiment Analysis

The key objective of the proposed framework is to train a neural network with the ability of categorizing as positive, negative, or neutral the sentiment associated with the text of the training data, so that it can be used to efficiently estimate sentiment of text not encountered before. The framework consists of two main phases: a learning phase, and a testing phase. An illustration is provided in figure 3.

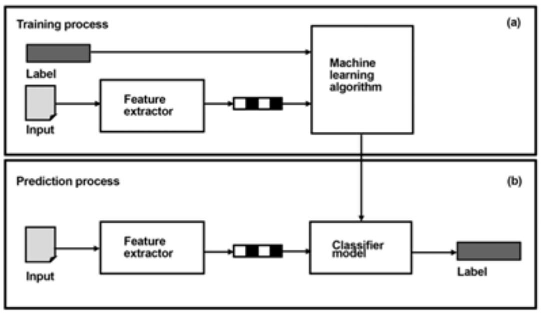

Figure 3: The sentiment analysis framework.
(top) Features are extracted from the raw input and fed to the machine learning algorithm which produces a classification model. (bottom) The classifier is used on new data. It extracts features and runs the model to categorize sentiment, i.e., predict the true labels

During the learning phase raw data consisting of twitter feeds are provided as input. The deep learning model extracts features that can be useful predictors of the sentiment associated with that text. The algorithm is trained using this data with the overall objective of identifying patterns between the extracted features and the text categorization. A classifier model is produced as output. To validate the effectiveness of the proposed approach, this classifier is tested on a separate dataset, previously unknown for the learning algorithm. The model takes as input raw data, and after implicitly extracting features, it estimates the sentiment associated with a given text, i.e., it predicts the true labels of the data.

This work performs sentiment analysis on a sentence level. We consider a three-class classification problem, that is, to classify a sentence as positive, negative, or neutral.

To achieve this task, we perform a pre-processing step of cleaning the dataset before we can feed it to the learning algorithm. Recall that our dataset is composed of twitter feeds. Tweets usually contain punctuation marks, large number of white spaces, non-characters, tags, and other characters that not only do not contain any valuable information for sentiment analysis, but that can affect on the opposite the performance of the algorithm. Our dataset was cleaned using the following steps:

- Cleaning tweets from retweets, tags (such as #) and removing links associated with sentences
- Cleaning all the non-letter characters
- Removing extra white spaces, punctuation, and stop words
- Converting all text to lowercase to avoid categorizing the same word differently
- Converting HTML to plain text

After cleaning the dataset, we apply state-of-the-art techniques such as *word embedding* before feeding the data to the learning algorithm. Word embedding is the process that maps each word to a vector of real values with the objective of representing similarly words that have a similar meaning.

The objective of the learning phase is to minimize the error between the output produced by the network (i.e., the predicted sentiment categorization) and the desired output (i.e., the true categorization of sentiment). More formally, this error can be expressed as follows:

$$E(w) = \frac{1}{K \cdot N} \Sigma_{k=1}^{K} \Sigma_{N=1}^{N_L} (y_n^k - d_n^k)^2$$

In the above equation, y_n^k represents the output produced by the classification algorithm, where K is the number of the input images and the desired output vectors. For each k not larger than the size of the dataset, x_k is the k–th entry, and d_k is the desired output vector.
For the learning to take place, we are interested in computing the gradient of the defined error term by calculating the partial derivatives of the error function with respect to the weighted sums of the input. Algorithm 1 presents the main steps to follow

Algorithm 1: Training algorithm update of weights

```
Initialize weights W and biases b of the neural network N
do
  for each training example $(x_i,y_i)$
    $p_i$ = neural-network-prediction $(N,x_i)$
    Calculate gradients of loss function $(p_i,y_i)$ with respect
    to $w^2$
    get $\delta w^2$ for all weights from hidden to output layers
    calculate gradient w.r.t $w^1$ by chain rule at hidden layer
    get $\delta w^1$ for all weights from input to hidden layers
    update $(w^1,w^2)$
until training examples are classified correctly or stopping
criteria met
return the trained neural network
```

As one can observe, the purpose of the algorithm is to continuously update the weights associated with the extracted features used in the training procedure while aiming to minimize the error function. For that reason, gradients of the loss function are calculated from hidden layers to output layers and then back-propagated recursively to adjust all weights between layers accordingly. The algorithm stops if it converges, i.e., the neural network is trained accordingly, or some other stopping criteria is met.

3.1. Layers of the Network

The neural network processes the input data by efficiently applying several steps. First it approaches the input using convolutional layers, and then decreases the dimensions of the input by down-sampling it through a process known as max pooling. The number of features is increased during this step. Finally, the fully connected layers allow for the network to operate by firing the activation and loss functions. In particular:

Convolutional layer's role is to extract features from the input data that will be used for classification. **Pooling/subsampling layers** are commonly inserted after convolutional layers with the purpose of reducing the dimensions and the spatial size of the input data. This process helps the network to reduce the number of parameters needed to explain the model, thus reducing computational complexity, and decreasing the chances of overfitting (i.e., the learning algorithm performs well on the training data, but poorly on testing data). During this phase, the extracted features are evaluated and checked for patterns, hierarchies or interconnections that can affect the classification. Noise reduction is also an output of this process.

Fully connected layers are often used as the final layers of the network. Their key objective is to validate the model and to ensure that the network will be able to efficiently perform the classification task, at hand. The output of the previous layers is used to determine the relations between parameters and their role in the learning model. Previous layers and their associated features are summed accordingly to determine the class of the target output. The network backpropagates this information and ensures that as more input data arrives, it is considered to enhance the learning experience. An intuitive illustration of these layers is provided in figure 4.

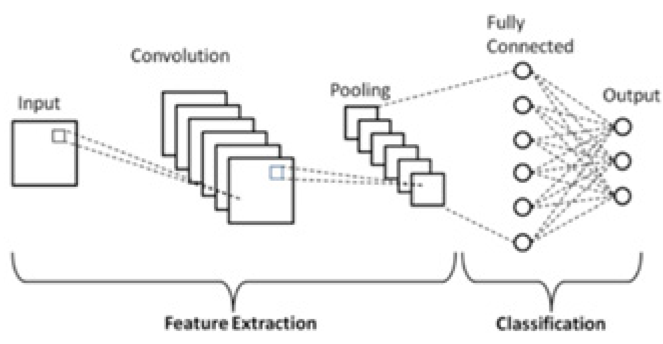

Figure 4: Convolutional neural network layers

Our framework is implemented using the Python programming language and several state-of-the-art libraries that efficiently deal with machine learning tasks, such as Keras (Chollet, 2015) and TensorFlow (Abadi, 2015). Keras is effective in the designing process of the neural network, allowing for smooth implementation of convolutional layers, while working well with raw data. TensorFlow is mainly used during the processes of training the neural network to automatically detect and classify our data. Other libraries such as Pandas (Team, 2020), Numpy (Harris, 2020), and Matplotlib (Hunter, 2007) are used during the pre-processing and visualization steps.

4. Experimental Results

4.1 Testing Environment and Dataset Description

A series of experiments is performed to evaluate the effectiveness of the proposed approach. The learning algorithm is fed a large dataset containing approximately 1.6 million labelled records, where each record is a twitter feed (Go, Bhayani, & Huang, 2019). In addition to that, another dataset (Misra & Arora, 2019) is used to train the model on sarcastic or humorous text. The training data, as described in section 3, are pre-processed, and cleaned from unnecessary parts that do not convey any meaning regarding the sentiment analysis. Expression of emotions through emojis helps to categorize sentiment. Table 1 provides an illustration.

Table 1 – Emoticons labeling and sentiment

Positive emoticons	Meaning	Negative emoticons	Meaning
:)	Smile	:(	Frown
:D	Green	:'(	Crying
:P	Tongue out	:@	Angry

4.2. Performance Measures

To evaluate the performance of the proposed approach, we rely on the standard measures of **precision, recall** and **f-measure** (K.M., 2011). F-measure is a score of a test's accuracy. It is calculated from the precision and recall of the test, where precision is defined as the number of true positive results divided by the number of all positive results, including those not identified correctly.

Recall is defined as the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value while recall is known as sensitivity in diagnostic classification. The F-score is the harmonic mean of precision and recall. In particular:

$$P = \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FP}} \qquad R = \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}} \qquad F = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

where *P,R,F* denote respectively precision, recall and f-measure, while *#TP,#FP,#FN* represent the number of true positive, false positives and false negatives. A combination of these three measures provides a standard and accurate description of the accuracy of the learning algorithm.

4.3. Results

Table 2 provides a summary of the precision, recall and f-score the dataset used in this work, while figure 5 demonstrates a visual representation of the loss and accuracy functions in terms of epochs, i.e., the number times that the learning algorithm will work through the entire training dataset.

Table 2 – Performance of the learning algorithm

Dataset	Precision	Recall	F-Score
1.6 million twitter feeds	94%	85%	89%

As once can observe, the learning algorithm performs with high accuracy both on the training and, most importantly, on the testing data. An increase in the number of echos allows the algorithm to refine further the connections

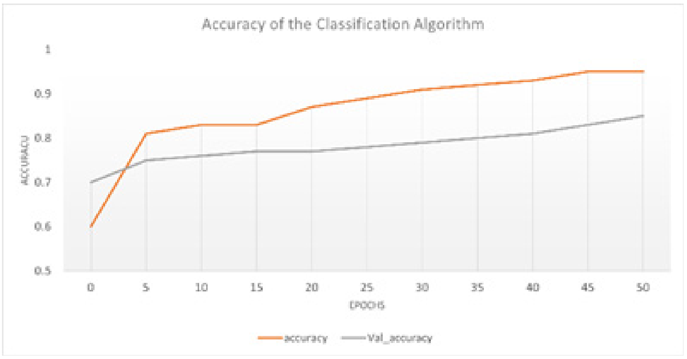

Figure 5: A visual representation of the accuracy function in the training and testing dataset, with respect of the number of steps (echos)

between features and the desired output, thus yielding even more accurate results.

The accuracy of the learning framework increases significantly with the number of iterations, as shown in figure 5. The proposed neural network framework is able to extract the necessary features that serve as predictors of the labelled data and efficiently predicts the sentiment associated with them. The data used during the testing procedure, i.e., for cross validation has a lower accuracy, in comparison. These results are expected considering the difficulties of text sentiment analysis presented earlier, but especially related to sarcastic or humorous text, nuances in written expressions, or the usage of slang-language. To address these challenges, we can further investigate the role that text processing plays in the learning framework, in addition to acquiring other datasets that contain more data in the above-mentioned categories, or to explore the role that various learning models and algorithms can play to achieve higher validation accuracy.

5. Conclusions

This paper presents a framework that addresses the problem of performing text sentiment analysis through the usage of convolutional neural networks. A high-volume dataset containing twitter feeds is fed as input to the neural network which then learns how to classify these feeds according to the sentiment associated with them. This is a difficult task that presents numerous challenges, as discussed through this thesis.

Instead of relying on hard-tuned parameters, in our approach we learn the underlying model through processing it in a neural network with multiple layers. The end-result of the learning phase is to extract features that can predict the labelled training data and to identify patterns that facilitate accurate classification. This model is then evaluated on a separate dataset, so that it can also perform well on previously unencountered environments.

Experimental validation shows the effectiveness of our approach into classifying twitter feeds with high accuracy.

This work opens several possibilities for improvements. In particular, the importance of an accurate, general, updated dataset is crucial for the accuracy of the trained model. We aim to enrich our dataset by including data from other sources as well and use the trained model to predict the sentiment of text. Another direction to extend the application is the usage of a database to save a history of how a particular word is used over time and what sentiment it is associated. This can be helpful to understand how language, words, and their meaning change overtime as well. The learning framework can benefit from integrating other sources of opinions, such as images, or texts from different sources. In general, enriching the set of features that are used to describe the text, or further investigating the correlations between different parts of the text from a semantic point of view can be another good direction for future work.

REFERENCES

A. Chachra, P. M. (2017). Sentiment analysis of text using deep convolution neural networks. Tenth International Conference on Contemporary Computing.

Abadi, M. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from https://www.tensorflow.org/
Cai, G. (2015). Convolutional Neural Networks for Multimedia Sentiment Analysis. Natural Language Processing and Chinese Computing.

Chen, H. S. (2016). Neural sentiment classification with user and product attention. Proceedings of the 2016 conference on empirical methods in natural language processing.

Chollet, F. (2015). Keras. Retrieved from https://keras.io

Firmino Alves, A., Baptista, C., Firmino, A., Oliveira, M., & Paiva. (2014). A Comparison of SVM versus naive-bayes techniques for sentiment analysis in tweets: A case study with the 2013 FIFA confederations cup. In Proceedings of the 20th Brazilian Symposium on Multimedia and theWeb.

Go, A., Bhayani, R., & Huang, L. (2019). Sentiment140. Retrieved from http://help.sentiment140.com/home
Harris, C. R. (2020). Array programming with NumPy. Nature.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment . Computing in Science and Engineering.

Huq, M., Ali, A., & Rahman. (2017). A. Sentiment analysis on Twitter data using KNN and SVM. IJACSA Int. J. Adv.

K.M., T. (2011). Precision and Recall. Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning.
Malik, V., & Kumar, A. (2018). Communication. Sentiment Analysis of Twitter Data Using Naive Bayes Algorithm. Int. J. Recent Innov. Trends Comput. Commun.

Minh-Thang Luong, R. S. (2013). Better word representations with recursive neural networks for morphology. Proceedings of the Conference on Computational Natural Language.

Misra, R., & Arora, P. (2019). Sarcasm Detection using Hybrid Neural Network. arXiv preprint arXiv:1908.07414.

Pinto, D., McCallum, A., Wei, X., & Croft, W. (2003). Table extraction using conditional random fields. 26th Annual International ACM SIGIR Conference on Research and Development in Informaion.

Salas-Zárate, M., Medina-Moreira, J., Lagos-Ortiz, K., Luna-Aveiga, H., & Rodriguez-Garcia, M. (2017). Sentiment analysis on tweets about diabetes: An aspect-level approach. Comput. Math. Methods Med. .

Soni, S., & A, S. (2015). Sentiment analysis of customer reviews based on hidden markov model. In Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology.

Tang, D. W. (2016). Sentiment embeddings with applications to sentiment analysis. IEEE Transactions on Knowledge and Data Engineering.
Team, P. D. (2020). Retrieved from https://doi.org/10.5281/zenodo.3509134

Zhang, X., & Zheng, X. (2016). Comparison of Text Sentiment Analysis Based on Machine Learning. In Proceedings of the 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC).