Sentiment Classification on Greek Smartphone Reviews

Georgitsis Christos
cgeorgit@csd.auth.gr

Dialektakis Georgios
gdialekt@csd.auth.gr

Tsanaktsidis Georgios
gtsanakts@csd.auth.gr

ABSTRACT

Sentiment Analysis is a growing task in Natural Language Processing over the last few years, as more companies want to extract sentiment from their customers' reviews so as to obtain feedback about their products and services. In this paper, we focus on classifying smartphones reviews in Greek language into five classes based on the rating of each review. We first discuss about the data preprocessing and text vectorization we applied on our gathered dataset. Then, we apply several machine learning algorithms that have been widely used for Sentiment Classification in the past and we present extensive performance results and comparisons.

Author Keywords

Sentiment Analysis; Natural Language Processing; Text Vectorization; Machine Learning Algorithms

INTRODUCTION

One of the most familiar techniques in Natural Language Processing (NLP) is Sentiment Analysis or Sentiment Classification, which is used to detect sentiment in textual data and decide whether they are positive, negative, or neutral. Over the past few years, sentiment analysis has rapidly grown and become a fundamental medium to observe and analyze customer feedback. In this way, businesses can modify their products or services to fit their clients' needs by learning what makes them delighted or frustrated.

Sentiment analysis methods can be divided into rule-based, automatic, and hybrid. A rule-based algorithm consists of human-crafted rules such as stemming, part-of-speech tagging, tokenization, etc., in order to recognize polarity in text. However, rule-based techniques are pretty naive as they do not consider the way words are combined in a sequence, while adding more advanced rules to cover new expressions makes the system quite complex. On the other hand, instead of relying on hand crafted rules, automatic approaches utilize machine learning algorithms to model the sentiment analysis task as a classification problem. In this context, given a text or document, a classifier is trained to predict in which class between positive, negative or neutral, the text falls into. Finally, hybrid approaches combine both rule-based and machine learning algorithms maintaining the beneficial components of each one, to make more accurate predictions.

In this paper, we focus on fine-grained Sentiment Analysis, which breaks down polarity to five categories (very positive, positive, neutral, negative, very negative). Specifically, we apply different machine learning algorithms such as Naive Bayes (NB), Support Vector Machines (SVM), Multinomial Logistic Regression (MLR) and Multilayer Perceptron (MLP), on Greek reviews on smartphone devices, where we aim to detect in which of the above polarity categories a review falls into. Furthermore, we conduct various experiments based on different data pre-processing techniques and we compare the results obtained by the utilized machine learning models. Finally, significant conclusions are reached referring to which method is more suitable for the Sentiment Analysis task.

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

RELATED WORK

This section briefly reviews relevant work on Sentiment Analysis.

Akshay et al. [8] examined various machine learning classifiers for extracting sentiment from restaurant reviews, where they achieved the highest accuracy of 94.5% on this dataset. In [16], Srujan et al. proposed a random forest method to analyze the sentiment of reviews on Amazon books. In their technique, n-gram features are obtained using TF-IDF, and accuracy of roughly 90.15% is reached. Fang et al. [4] introduced a model, composed of a sentiment score formula and three different classifiers, which was used to distinguish the polarity of the text from online product reviews. In [15], a model based on Support Vector Machines is developed to detect positive and negative emotion on smartphone reviews, which attained an accuracy of 81.77%. Xu et al. [20] presented a deep learning-based method for sentiment detection on medical data, where they obtained an accuracy of 85% on emotional fatigue.

The closest related work to ours is perhaps in [11], where Bo Pang et al. used three machine learning techniques (Naive Bayes, maximum entropy, and SVM) to determine whether a movie review is positive or negative. They discover that conventional machine learning methods clearly yield better performance than rule-based and hand-crafted ones. They also examine the effect of unigrams and bigrams as features and they observe that considering only unigrams is most effective in this specific problem. Finally, they conclude that SVM outperforms Naive Bayes and Maximum Entropy, although the difference between their performance is not large.

Overall, the task of Sentiment Analysis is one of the most researched topics in NLP, especially in the English language. However, to the best of our knowledge, limited research has

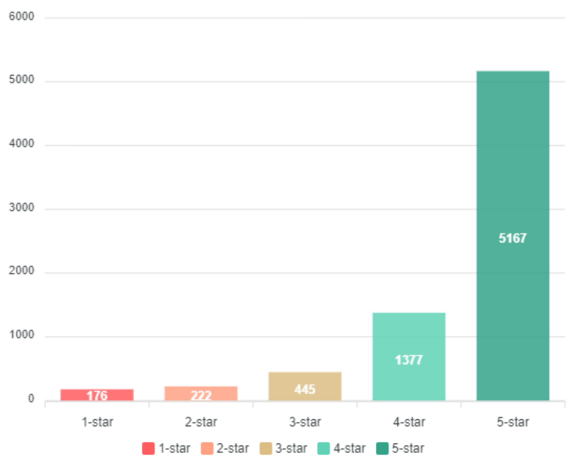

Figure 1. Number of reviews per rating

1-star	2-star	3-star	4-star	5-star
176	222	445	1377	5169

Table 1. Number of reviews per rating.

been conducted in the Greek language and no baselines exist on Greek datasets. Moreover, most Sentiment Analysis techniques divide the polarity of the text into three categories (positive, negative, and neutral). For these reasons, we propose the use of several machine learning techniques to analyze sentiment on smartphones reviews we collected in the Greek language, and we classify these reviews into five categories: very positive, positive, neutral, negative, very negative.

DATA & PRE-PROCESSING

For our experiments we gathered smartphone reviews in the Greek language from a website, named Skroutz [5], for 88 different smartphones models. The dataset consists of 7387 reviews with ratings ranging from 1-star up to 5-stars. As mentioned before we classify the reviews into 5 categories according to their star rating: 1-star is very negative, 2-stars is negative, 3-stars is neutral, 4-stars is positive and 5-stars is very positive. The number of reviews per category is shown in table 1 and visualized in figure 1. The data is saved in a JavaScript Object Notation (JSON) file and has the following structure: "phone": the model of the phone, "rating": 1-5 the rating of the phone, "review": the text of the review. The ground truth for our experiments is the "rating" and we train and test the various algorithms on the "review".

We notice that the dataset is imbalanced, having 69.94% of the reviews in the 5-stars category and only 2.38% in the 1-star, or about 1:30 ratio for 1-star to 5-stars reviews. This ratio is not extreme, but it could affect the performance of standard classifiers [18]. Training on imbalanced data can result in models that are biased towards the majority (5-stars) class and have poor prediction performance on the minority classes [12]. There are various ways to handle imbalanced data and create a balanced dataset. One technique is under-sampling the majority classes by removing instances of the data from these classes. Another approach is over-sampling the minority classes using algorithms that create duplicates or similar instances from these classes. It should be noted that under-sampling can lead to information loss, while over-sampling can lead to overfitting [21]. In this paper, we decided to train and evaluate the classifiers on the dataset (a) as-is, then (b) under-sampling with a 1:1 ratio, and (c) over-sampling. Before implementing the classifiers there is a need for preprocessing the dataset. The goal of the preprocessing step is to clean the dataset from unwanted characters, e.g. html tags (<br>, <p> etc.), remove extra white spaces, tokenize the reviews into words, remove common words (stop words) and punctuation that do not add any valuable information to sentiment analysis and normalize the tokens. To accomplish these tasks we use a Python library named spaCy [7]. Common text normalization techniques are Stemming where the word is reduced to its root form e.g. "reducing" -> "reduc" and Lemmatization where the word is reduced to is lemma e.g. "reducing" -> "reduce". The goal for both of these techniques is to decrease the inflectional and related forms of a word to a common base form, ultimately resulting in a reduction of the features. We decided to use lemmatization since it preserves the meaning of the words and their Part Of Speech (POS) tag [3].

Text Vectorization

Before applying any machine learning classifier, there is a need to transform the cleaned text of each review to an array (or vector) of numbers. We used two methods for text vectorization: the first one is Bag of Words (BoW) and the second one is Term Frequency – Inverse Document Frequency (TF-IDF). The Bag of Words method describes the occurrence of tokens in a document. Given a dictionary (vocabulary) with all the distinct tokens from the dataset and a text, the method transforms the text into a vector that is the same size as the vocabulary [19]. Then, the method counts the number of times a word from the dictionary appears in the text and stores that number in the corresponding position of the array. An example of BoW is the following: consider the dictionary: ["οθόνη", "μπαταρία", "κακό", "καλή", κινητό"] and the text "κινητό μπαταρία καλή". BoW transforms the text to the vector (0, 1, 0, 1, 1). The reason it is called Bag of Words is because the method does not consider the position of the words in the text. It is common to create the dictionary with bigrams and of course the text is also split into bigrams, in order to capture more context from the document. The TF-IDF [19] method aims to weigh each term by its relevance in a document and consists of 2 parts, the first one is the Term Frequency which is the frequency of a word t in a document d:

$$\mathrm{tf}_{t,d} = count(t,d) \tag{1}$$

It is common to use a logarithmic scaled version of the count:

$$\mathrm{tf}_{t,d} = log_{10}(count(t,d)+1) \tag{2}$$

Using equation 2 the high frequency terms are squashed. The idea behind this is that if term t appears 100 times in document d, it does not necessarily mean that t contributes to 100 times more than the actual meaning of d. The second part is the

Inverse Document Frequency (idf):

$$\mathrm{idf}_t = \frac{N}{\mathrm{df}_t} \tag{3}$$

where N is the number of documents and df_t is the number of documents in which t occurs. If a term occurs in all documents, then the idf_t will be 1 so that rare terms get a higher value than the more common ones. It should be noted that since N may be a high value, depending on the number of documents, idf_t also gets squashed using *log*:

$$\mathrm{idf}_t = log_{10}(\frac{N}{\mathrm{df}_t}) \tag{4}$$

Then the final tf-idf weight (score) is calculated for each term in every document using the following function:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t \tag{5}$$

These weights can later be used as features for our classifiers, instead of the simple term frequency.

The last approach we apply for the task of Sentiment Classification is a Long Short-Term Memory (LSTM) network, which is a variant of Recurrent Neural Networks (RNN) [6]. Specifically, LSTM has showed great effectiveness in obtaining the long-term dependencies in a text. Moreover, we make use of Bidirectional LSTM (BiLSTM) [14], which consists of a forward LSTM layer and a backward LSTM layer, and is able to capture information from both previous and next words in the text. The architecture of our BiLSTM network is shown in figure 2.

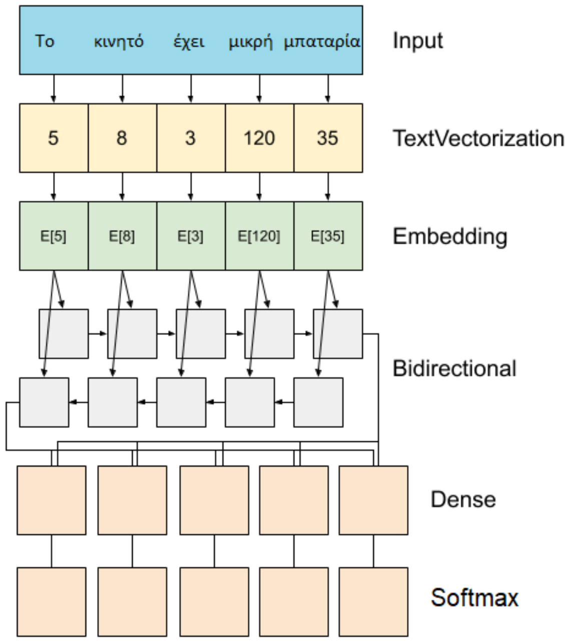

Figure 2. Architecture of Bidirectional LSTM

In this chapter, we represent the methods that we have implemented for sentiment analysis to our dataset. Specifically, we choose four standard algorithms: Multinomial Logistic Regression, Naive Bayes classification, support vector machines, and Multi-layer Perceptron.

Multinomial Logistic Regression

The first approach we have implemented for our text classification problem is a supervised learning algorithm named Multinomial Logistic Regression. We choose Multinomial logistic regression because it does not assume normality, linearity, or homoscedasticity. Instead, Multinomial logistic regression does have assumptions, such as the assumption of independence among the dependent variable choices. With this assumption, we ensure that the choice of each category is not related to others. With the use of maximum likelihood estimation, categorization is achieved [17]. We use TF-IDF to our dataset, to convert our reviews – sentences into vectors. Subsequently, we fit the vectorizer on the whole dataset. We can fit the vectorizer on just training dataset but that can have negative implications. As some words which were not present in training can be there in Test Dataset. Since the data is in a long sparse matrix, simple logistic Regression works well.

MODELS

Support Vector Machines

Support Vector Machines (SVMs) is a supervised machine learning algorithm that has proved quite effective at classification problems. SVM aims to find an optimal boundary between the possible outputs. The model does complex data transformations depending on the selected kernel function and based on that transformations, it tries to maximize the separation boundaries between the data points depending on the classes [1]. In our research, we implement One-vs-Rest (OVR) SVM, a heuristic method for applying binary classification algorithms for multi-class classification. OVR involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier creates a hyperplane in order to separate a specific class from all the other classes. Then the model is trained on each binary classification problem and predictions are made using the model that is the most confident [2].

Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) is the Naïve Bayes classifier for discrete features. These methods are based on applying Bayes theorem with the “naive” assumption of independence between every pair of features [9]. Naïve Bayes makes simple calculation and can avoid curse of dimensionality. Naïve Bayes train the parameter by combine the statistical probability on each class. The most common Bayesian model in text classification domain for large vocabularies is multinomial model that uses features bag of words and TF-IDF [10]. Moreover, we use this model due to the fact that we have documents that belong to one out of the 5 categories. In multinomial classification, the classes are mutually exclusive and each document or item appears in exactly one class. The only difference with the Naïve Bayes is that we build a separate binary classifier trained on positive examples from c and negative examples from all other classes. Now given a test document or item d, we run all the classifiers and choose the label from the classifier with the highest score. In other words, we compute the

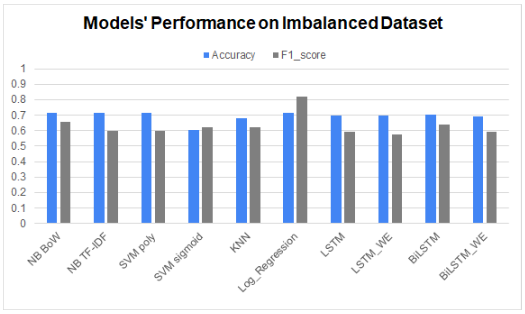

Figure 3. Performance of all models on imbalanced dataset.

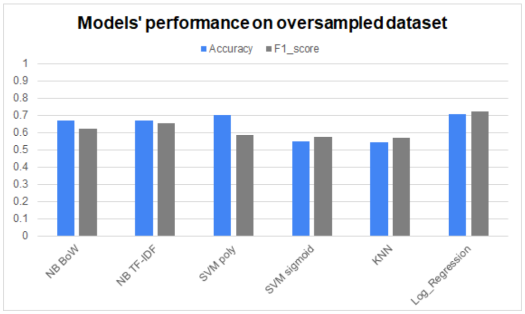

Figure 4. Performance of various models on oversampled dataset.

probability of each class label in the usual way, then we pick the class with the largest probability.

K-Nearest Neighbors

K-nearest neighbors (KNN) is a supervised learning algorithm where the result is classified based on the majority vote from its K nearest neighbor category. Due to the highly accurate predictions of the algorithm, KNN is considered one of the most accurate models. In this point we should mention that the quality of the predictions depends on the distance measure. Therefore, the KNN algorithm is suitable for applications for which sufficient domain knowledge is available. This knowledge supports the selection of an appropriate measure. The KNN algorithm is a type of lazy learning, where the computation for the generation of the predictions is deferred until classification [13]

Long Short-Term Memory

The first layer of our network is a Text Vectorization layer. For this layer, we consider two different methods. The first one is simple text vectorization which applies some common text preprocessing techniques and then maps the input string to a list of integer token indices. The second method we use is loading pre-trained word embeddings in Greek from spaCy [7], as we aim to capture the semantic meaning of the smartphone reviews and make our model more effective. The second layer of our network is an Embedding layer which converts the sequences of word indices to sequences of vectors that are trainable. Next, we have one forward and one backward LSTM layer composed of 32 units each, which processes the input sequence from left to right and from right to left, respectively, and concatenates the two outputs into one final output. We also use dropout with a rate of 20% to avoid over-fitting. Next, the output of the bidirectional LSTM layer is mapped to a Dense layer composed of 5 units, as our classification task consists of 5 classes, based on the rating of the review (1 to 5). Finally, the output layer of the network is a Softmax activation layer composed of 5 units, which transforms the output of the network to a probability distribution over the five predicted output classes.

EXPERIMENTS

In figure 3, we observe the performance of our machine learning models after fitting them on the initial imbalanced dataset

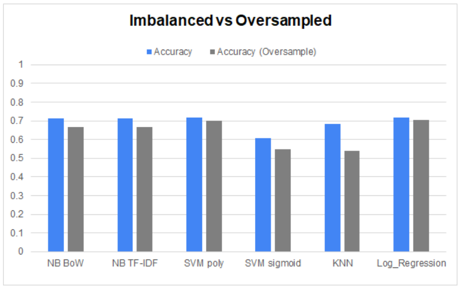

Figure 5. Performance of various models on imbalanced and oversampled dataset.

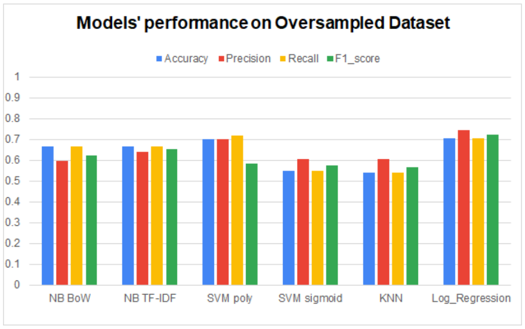

Figure 6. Performance of various models on oversampled dataset.

(classes are not represented with the same number of samples). With the use of two well-known metrics accuracy and F1_score, we depict our models' performance. Logistic regression achieves the highest rate in terms of F1_score while providing the highest accuracy. Subsequently, Naïve Bayes with bag of words achieves the same rates of accuracy as Logistic Regression, however, F1_score fluctuates at lower levels. Similar percentages of accuracy are achieved by Naïve Bayes with TF-IDF but F1_score is a lit bit lower than the previous model. A high level of accuracy is achieved by polynomial SVM with nearly the same F1_score as Naïve Bayes with TF-IDF. The linear SVM model and KNN achieves a little bit lower results than the metrics we mentioned earlier. Finally, as we can observe from the chart, LSTM models achieve almost the same results, with the BiLSTM model achieving a little bit better results.

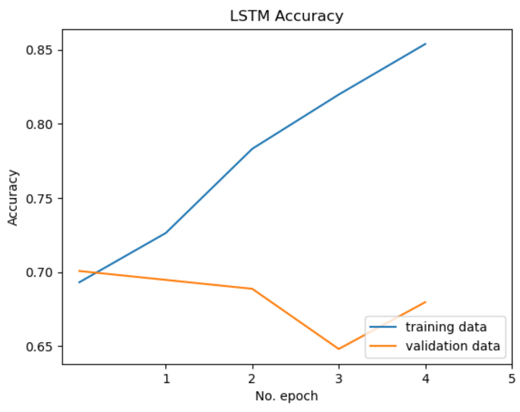

Figure 7. LSTM accuracy curve.

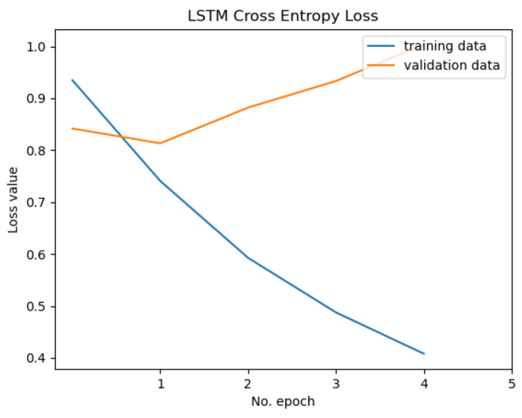

Figure 8. LSTM loss curve.

Figure 4 presents the performance of the models we employed, except for the LSTM and BiLSTM, after applying oversampling on our imbalanced dataset. We notice that Multinomial Logistic Regression provides the highest accuracy and f1_score exceeding the boundary of 70%. On the other hand, the worst results are given by the KNN and the SVM sigmoid algorithm whose accuracy and f1_score are just above 50%. In addition, the SVM polynomial achieves the same accuracy as Logistic Regression, however, its f1_score is 10% lower than that of Logistic Regression.

In figure 5, we can concede that our models achieve lower metrics when they are trained on oversampled data compared to the initial dataset. We also notice that all models, except KNN and sigmoid SVM, obtain almost the same results regardless of the method we used (oversample or initial dataset). Instead, these two models achieve better performance when the initial dataset is used. We should also note that the undersampling method was applied to our initial imbalanced dataset, however, due to the low results, we decided not to display them in our charts. In figure 6, we notice that Multinomial Logistic Regression achieves the highest performance when we use the ovesampled dataset. Moreover, Naive Bayes with TF-IDF achieves a little bit higher metrics than with the bag of words technique. Finally, KNN and sigmoid SVM achieve the lowest metrics.

Figure 11 shows the performance as achieved by the LSTM and the BiLSTM, both with and without word embeddings. We observe the accuracy and recall is roughly the same regardless of the model we used. However, BiLSTM without word embeddings outperforms all other models in terms of precision and F1_score. It is essential to mention that an important factor for which all models achieve the same accuracy is that our neural networks are overfitting on the training data, as shown in figures 7, 8, 9, and 10.

In specific, in figures 7, 8, 9, and 10, we show the accuracy and loss curves during the training of the LSTM and BiLSTM over 5 epochs, respectively. We observe that for both models, the loss for the training curve continues decreasing until the last epoch, while the validation curve increases after the second epoch. Moreover, in terms of accuracy, the training curve has

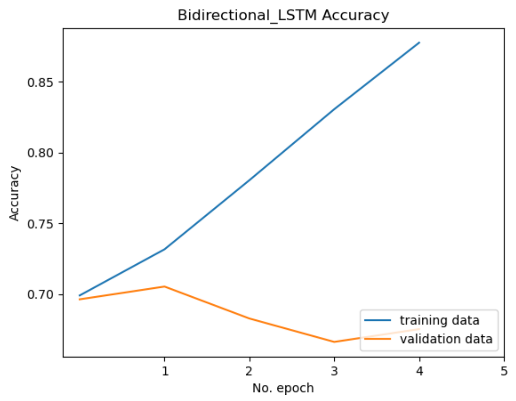

Figure 9. BiLSTM accuracy curve.

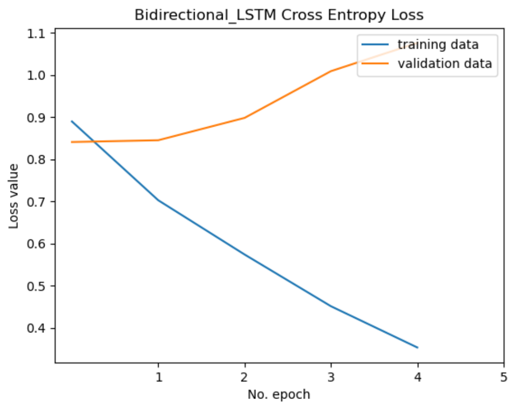

Figure 10. BiLSTM loss curve.

an upward course until the last epoch, while the validation curve has a downward course after the third epoch. This shows that, both the LSTM and the BiLSTM, are overfitting on the training data from a very early stage and are unable to make any meaningful progress regarding the validation accuracy or loss. The most important reason for that is the amount of data we have in order to train the network. Specifically, the initial dataset only consists of roughly 7000 samples, which

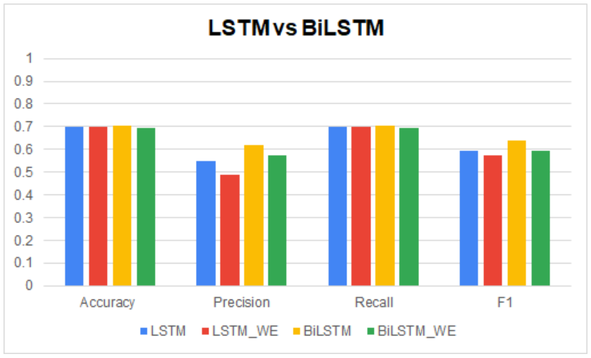

Figure 11. Performance of LSTM and BiLSTM.

Model	Accuracy	Precision	Recall	F1
NB BoW	0.7152	0.6637	0.7152	0.6600
NB TF-IDF	0.7145	0.5746	0.7145	0.5983
SVM poly	0.7165	0.6980	0.7165	0.6011
SVM sigmoid	0.6076	0.6355	0.6076	0.6202
KNN	0.6820	0.5969	0.6820	0.6218
Log_Regression	**0.7171**	**0.9675**	**0.7171**	**0.8197**
LSTM	0.6982	0.5470	0.6982	0.5927
LSTM_WE	0.6996	0.4894	0.6996	0.5759
BiLSTM	0.7050	0.6193	0.7050	0.6387
BiLSTM_WE	0.6942	0.5738	0.6942	0.5951

Table 2. Performance of all models on imbalanced dataset. (Best score in bold.)

is a quite small dataset for a LSTM to provide state-of-the-art performance. Moreover, the imbalance of the dataset plays a significant role as the network is provided with very limited samples for the classes 1, 2, and 3, which makes it difficult to learn to classify data into one of these classes. For this reason, during training, we save the weights of the neural network after every epoch only if the validation loss has decreased over the last epoch. In this way, we maintain the best viable model weights based on the validation loss. Finally, when the training completes, we load the saved weights and we make predictions on the test set using the saved model.

Finally, in table 2 we present an overall summary of the models' performance on each metric when they are trained on the initial imbalanced dataset. It is noticeable that Logistic Regression outperforms all the other models. Moreover, its precision and F1_score is much higher than those of the other models.

CONCLUSION

Sentiment analysis is an important task that helps businesses understand the overall opinions of their customers. In this paper, we experimented with multiple classic machine learning algorithms, such as Naive Bayes, SVM, Logistic Regression and more, we also experimented with the LSTM and BiLSTM models. Our experiments showed that Logistic Regression outperformed all the other models for our dataset, and that both LSTM and BiLSTM models did not perform as expected. This poor performance can be explained by the small size of our dataset, the imbalance of the dataset and the lack of training our own word embeddings. In the future, we aim to obtain more data so as to increase the size of dataset, mitigate the imbalance if possible, and experiment on different types of neural networks, such as Convolutional Neural Networks (CNN) and transformers, especially Bidirectional Encoder Representations from Transformers (BERT).

REFERENCES

[1] Baeldung. 2020. Multiclass Classification Using Support Vector Machines. (2020). `https://www.baeldung.com/cs/svm-multiclass-classification`

[2] Jason Brownlee. 2020. One-vs-Rest and One-vs-One for Multi-Class Classification. (2020). `https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/`

[3] Erik Cambria, Soujanya Poria, Alexander Gelbukh, and Mike Thelwall. 2017. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems* 32, 6 (2017), 74–80.

[4] Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data* 2, 1 (2015), 5.

[5] George Hadjigeorgiou. 2005. *Skroutz*. `https://www.skroutz.gr/`

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI: `http://dx.doi.org/10.1162/neco.1997.9.8.1735`

[7] Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* 7, 1 (2017).

[8] Akshay Krishna, V Akhilesh, Animikh Aich, and Chetana Hegde. 2019. Sentiment Analysis of Restaurant Reviews Using Machine Learning Techniques. In *Emerging Research in Electronics, Computer Science and Technology*. Springer, 687–696.

[9] Akshi Kumar and Arunima Jaiswal. 2017. Empirical study of twitter and tumblr for sentiment analysis using soft computing techniques. In *Proceedings of the world congress on engineering and computer science*, Vol. 1. 1–5.

[10] Young-Man Kwon, So-Hee Jun, Won-Mo Gal, and Myung-Jae Lim. 2018. The Performance Comparison of the Classifiers According to Binary Bow, Count Bow and Tf-Idf Feature Vectors for Malware Detection. *International Journal of Engineering & Technology* 7, 3.33 (2018), 15–22.

[11] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. *arXiv preprint cs/0205070* (2002).

[12] Lizhi Peng, Hongli Zhang, Bo Yang, and Yuehui Chen. 2014. A new approach for imbalanced data classification based on data gravitation. *Information Sciences* 288 (2014), 347–373.

[13] B Gnana Priya. 2019. Emoji based sentiment analysis using KNN. *International Journal of Scientific Research and Review* 7, 4 (2019), 859–865.

[14] Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* (1997).

[15] Zeenia Singla, Sukhchandan Randhawa, and Sushma Jain. 2017. Sentiment analysis of customer product reviews using machine learning. In *2017 International Conference on Intelligent Computing and Control (I2C2)*. IEEE, 1–5.

[16] KS Srujan, SS Nikhil, H Raghav Rao, K Karthik, BS Harish, and HM Keerthi Kumar. 2018. Classification of Amazon book reviews based on sentiment analysis. In *Information Systems Design and Intelligent Applications*. Springer, 401–411.

[17] Jon Starkweather and Amanda Kay Moske. 2011. Multinomial logistic regression. *Consulted page at September 10th: http://www. unt. edu/rss/class/Jon/Benchmarks/MLR_JDS_Aug2011. pdf* 29 (2011), 2825–2830.

[18] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. 2009. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence* 23, 04 (2009), 687–719.

[19] Virginia Teller. 2000. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. *Computational Linguistics* 26, 4 (2000), 638–641.

[20] Jianqiang Xu, Zhujiao Hu, Junzhong Zou, and Anqi Bi. 2019. Intelligent Emotion Detection Method Based on Deep Learning in Medical and Health Data. *IEEE Access* 8 (2019), 3802–3811.

[21] Yilin Yan, Min Chen, Mei-Ling Shyu, and Shu-Ching Chen. 2015. Deep learning for imbalanced multimedia data classification. In *2015 IEEE international symposium on multimedia (ISM)*. IEEE, 483–488.