Comparing approaches to analyze sentiment

Luka Žontar
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
Email: lz3057@student.uni-lj.si

***Abstract*—Sentiment analysis can help us automatically determine writer's attitude towards the text on enormous datasets. In practice, it helps companies make important business decisions and evaluate public opinions on their products. This article should help readers understand different machine learning approaches to sentiment analysis. We also explain how textual data should be preprocessed generally and for different models.**

This article focuses on deep learning machine learning models. At first, we take the Naive Bayesian model to compare it with more complex machine learning learning models. In particular, we try to evaluate a deep neural network with LSTM architecture and a BERT model that was fine-tuned on data that was used in training and validation of other models. As our dataset, we used Sentiment140, which contains pre-classified Twitter posts. In the end, we combine the aforementioned models to construct a majority voting ensemble model.

We found that a fine-tuned BERT yields best results, but also overfits on training data, while dropping almost 20% on scores of all evaluation metrics on new data. On the other hand, the LSTM architecture provides us with slightly lower performance, but also lower decrease in classification scores on new data, which dropped to around 6%. While BERT in general outperforms LSTM, the latter performs better on classification of negative sentiment. We chose Naive Bayesian classifier as our baseline model and all other three models yield better results, thus, beating the baseline model. With a fine-tuned BERT model, we managed to reach almost 77.5% accuracy, while with LSTM the accuracy dropped to 74.5%.

***Index Terms*—Sentiment analysis; classification; deep learning; neural networks.**

I. Introduction

Sentiment analysis is becoming an important technique used in many companies worldwide. With it, we use natural language processing to systematically identify subjectiveness in texts. In other words, in sentiment analysis we try to determine, whether the writer is positively or negatively affected by the written content. We try to evaluate writer's inclination towards the text using natural language processing and machine learning techniques to assign sentiment scores to phrases and sentences.

In practice, sentiment analysis helps us automatically process enormous amounts of textual data that customers write on social media and in survey responses. Customer feedback analysis can help us determine whether or not customers are happy with our product. This can improve our company's strategic decisions, such as deciding in which product to invest or how to make customers happier.

Furthermore, it can help better understand public opinions about a particular subject. This can help politics predicting voting results and consequently form alliances between parties. Understanding people's opionions can also make us better investors, since stock market stability depends on other investors.

In this article, we focus on comparing different sentiment analysis techniques. The goal of this article is to help readers understand different types of sentiment analysis approaches and when they should be used. We explain different preprocessing techniques and compare a baseline Naive Bayesian model with more complex deep learning based models LSTM and BERT. In the end, we compare results with a majority voting ensemble model that is combined of all three aforementioned models.

For learning and validation we use the Sentiment140 dataset, which contains pre-classified Twitter posts. Tweets represent the general public and thus all sorts of opinions can be found in the dataset we used. Some opinions are stronger and in some it is difficult to determine the writer's inclination towards the text topic. This makes the given dataset an appropriate training and validation set.

In next section, we make an overview of related work that was relevant for our research. After that, we explain the methodology of our work, that is, how data was preprocessed and how models were evaluated. In the forth and fifth section, we present results and discuss them. Lastly, we conclude this article by presenting the main findings.

II. Related work

In this section, we present related work that was relevant for our article. Sentiment analysis was proven to be useful in different areas. For example, in medicine, sentiment analysis is used to evaluate healthcare textual data. In medical sentiment, we use medical information to achieve the best result to increase healthcare quality [1]. As mentioned before, social media holds loads of textual data of different sorts. Muzafar et al. discusesed how SARS-Cov-2 affected people on social media [2]. At first, the most commonly searched terms were Coronavirus related. And lots of negativity could be seen textual data from social media.

Feldman [3] discusses several approaches of sentiment analysis based on the basic unit that we will be classifying. We can classify whole documents, sentences or even particular aspects. Furthermore, we have to consider comparative statements such as less and more. As the most crucial field of sentiment analysis Feldman states sentiment lexicon acquisition, where words can be expanded in a set of words using corpus-based approaches or WordNet.

In past works, lots of researchers also worked on textual data preprocessing. Magliani and his colleauges [4] evaluated lots of preprocessing techniques and found that preprocessing vastly increases model's accuracy. However, they found that basic preprocessing with stemming is the best way to improve accuracy with preprocessing. They also found that processing negations, stopwords and emoticons also slightly increases accuracy.

Latha [5] focused on how to process informal social media texts and emphasized the importance of removing URLs, special characters and repeated letters from texts. Furthermore, she proposed that question words should be removed from texts. We also noticed an interesting approach by Krouska et al. [6], who focused on the length of n-grams passed to classifiers. N-grams of length greater than 1 hold a lot more information about the text polarity than a single word, which is why they are used to get better accuracy on sentiment analysis classification.

To better understand sentiment analysis, we also need to get acquainted with the top performing machine learning models. Zainudin [7] split the sentiment analysis models into three major categories, lexicon based, machine learning based and hybrid approaches. Machine learning based approaches are further devided into supervised, semi-supervised and unsupervised approaches. In this article, we will be focusing on supervised machine learning approaches.

Starting with simpler methods, Dey [8] et al. used statistical models to capture elements of subjective style and the sentence polarity. They managed to get above 80% accuracy on a movie review dataset that is comprised of 4500 samples. In this study, Naive Bayesian classifier was compared with k-NN classifier. However, k-NN classifier failed to achieve comparable results on this dataset. Researchers also noticed high discrepancies amongst accuracies on different datasets.

In recent years, there have been several breakthroughs in the area of sentiment analysis. Recurrent neural network architectures, such as long short-term memory (LSTM), have proven to be very succesful in learning to evaluate polarity of text. Besides the standard feedforward connections, LSTM adds feedback connections. Putra et al. [9] improved the basic LSTM architecture by adding exploiting the use of Word2Vec embeddings. They managed to achieve 85.96% accuracy. Word embeddings make the regular LSTM architecture more robust to overfitting. For Arabic texts, an additional improvement to LSTM architecture was proposed by Elfaik and Nfaoui [10], who used a deep bidirectional LSTM network to further increase the accuracy. While the complex model clearly overfitted on some datasets, it reached over 90% accuracy on others.

A novel approach was proposed by Devlin and colleauges [11], who used deep bidirectional transformers for language understanding. These transformers were trained on enormous datasets and finally fine-tuned on sentiment analysis tasks to produce state-of-the-art results.

III. Methods

In this section, we explain different methods that were used in this comparative study. Firstly, we describe the data that we used to get our results. Next, we explain the preprocessing methodology and finally, we describe the machine learning models that were used in this research. We used four different models: Naive Bayesian classifier, neural network with deep bidirectional LSTM architecture, BERT model fine-tuned with our data and a majority voting ensemble classifier.

A. Data

As already mentioned, we decided to use the Sentiment140 dataset [12] that contains 1,600,000 tweets extracted using Twitter API. The initial dataset hodls multiple features:

1) Target - polarity of the tweet,
2) Id - the id of the tweet,
3) Date - the date of the tweet,
4) Flag - the query (lyx). If there is no query, then this value is NO_QUERY.
5) User - the user that tweeted. Text - the text of the tweet.

We decided that only text is relevant for our study and the models we use are trained only with the text column, while we are trying to predict the target column.

In Figure 1 we can see the most common words in the wordcloud representation. Due to time complexity of our research, we only use a randomly sampled subset of 40,000 tweets. While the dataset holds multiple features, we only use the tweet text and target variable from which we filter out the neutral tweets to make our problem a binary classification problem. Also, the positive tweet polarity, which is denoted with 4 is mapped to 1 due to convenience.

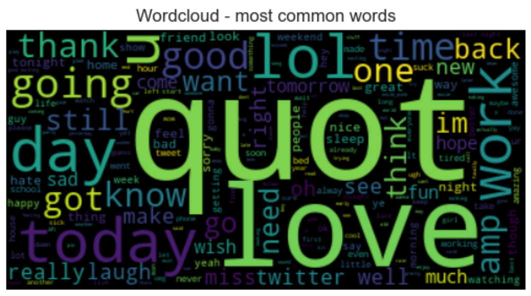

Fig. 1. **Wordcloud:** from figure we see the most common words in tweet. People often use quotes and words like day, today, love and word.

As can be seen from Figure 2, our classification problem is balanced. That is, the two classes are uniformly distributed. The next steps of preprocessing pipeline consists of the following:

1) Remove non-ASCII characters.
2) Remove links, hashtags, mentions and repetitive vowels (such as “hahaa”).
3) Replace laughing with a predefined word “laugh”.
4) Replace emojis with predefined combinations such as “smile”.
5) Switch to lower case.
6) Filter stopwords.
7) Tokenize words and turn them into appropriate form (depending on the algorithm).

We also tried to improve evaluation metric scores with stemming words, fixing misspells, handling negations and removing retweets. However, these preprocessing techniques made the models overfit faster and effectively reduced evaluation metric scores. While, Naive Bayesian classifier was not really effected by those, LSTM and BERT performed worse. We assume that performance drop occured due to different pretraining in BERT and failing to find appropriate embeddings in LSTM with word embeddings. Rows with null values were dropped and finally, we used 80% of data for training and the rest for testing our models.

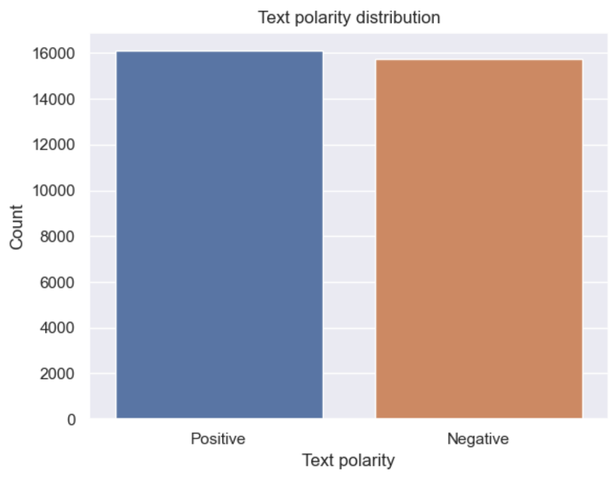

Fig. 2. **Text polarity distribution: from figure we denote that text polarity is almost uniformly distributed. Both classes are representable.**

In Figure 3, we can see the distribution of tweet word count after preprocessing. This later helpes us define parameters and model complexities in neural networks.

We also evaluated polarity of texts using the *TextBlob* library, which tokenizes words and evaluates sentiment by looking at the *SentiWordNet* [13] lexical resource. For every word in *WordNet*, the lexical resource returns positivity, negativity and neutrality score. In continuation, in Figure 4 we see how polarity varies in our texts.

B. Naive Bayes

Naive Bayesian classifier is one of the fastest and easiest methods that can also perform very well. The basic idea of this classifier is to find the probabilities of classes based on the input text. Textual data is passed to the model with word count

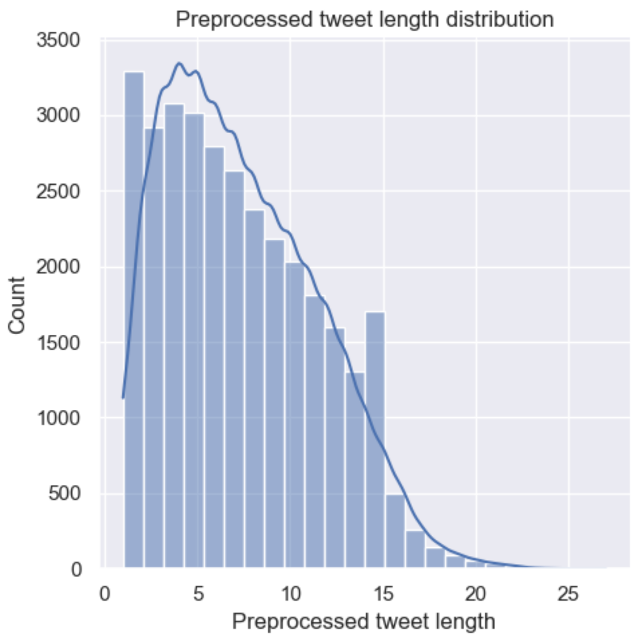

Fig. 3. **Preprocessed tweet word count distribution:** as can be seen from the figure, the distribution is right-skewed. This means that tweets are short in general. The distribution does not have a long tail, which is why, we do not have to handle outliers in this situation.

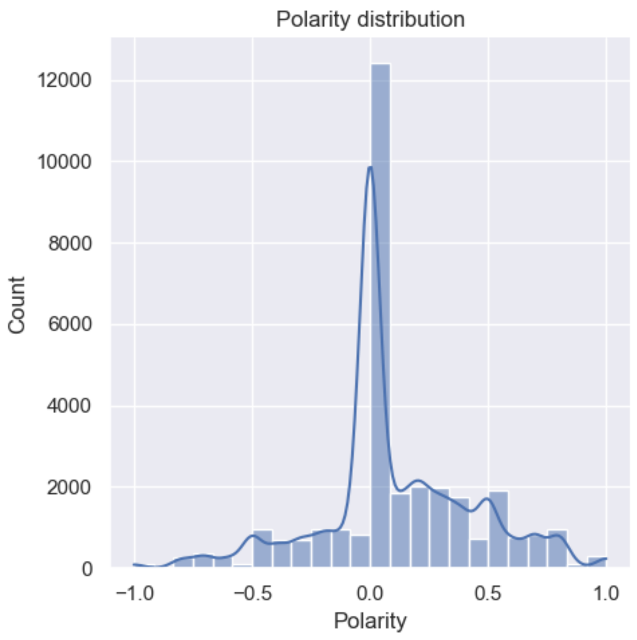

Fig. 4. **Polarity distribution:** as can be seen from the figure, the distribution peaks in the middle, meaning that the majority of tweets would be interpreted as neutral if handled with *TextBlob* library. Otherwise, it seems as though *TextBlob* would evaluate more tweets as positive than negative.

vectors. With bigger datasets this model overfits, however, it tends to be used as the baseline model.

C. LSTM

As mentioned before, the LSTM architecture was designed to grasp long term dependence between words. With it, we build a recurrent neural network that can memorize certain patterns, which general recurrent neural networks cannot.

As already mentioned, LSTM has been proven to be very succesful in sentiment analysis. We implemented the aforementioned bidirectional LSTM architecture with GloVe word embeddings instead of matrix of tokenized word encodings. We used GloVe with 200 dimensional representations. Next, we describe the architecture of the LSTM model in more detail:

1) Model accepts a matrix of tokenized tweets padded to the pre-defined maximum length. This was set to 60 due to tweet length distribution.
2) To decrease the chance of overfitting, we transform this matrix to a matrix of word embeddings and only use the top 4000 most common words.
3) Learning rate is set to $5e{-}5$ and we train the model for maximum 100 epochs. Early stopping is implemented if the model starts to overfit.
4) The model consists of two bidirectional LSTM layers with dropout (once again, to prevent overfitting).
5) In the end results are densed using ReLU activation function and on the last layer a sigmoid function.
6) We use binary crossentropy loss function and Adam optimizer.

D. BERT

BERT, which stands for bidirectional encoder representations from transformers, uses bost previous and next tokens to make predictions. It randomly masks words in a particular context and predicts them.

We used the base uncased BERT and fine-tuned it on Sentiment140 dataset. In this case, no hyperparameter tuning was done due to time complexity and since authors of BERT have already suggested the optimal parameters:

1) Tokenizer uses at most 64 words, which in our case is sufficient and also similar to the LSTM setting.
2) Batch size: 32.
3) Learning rate: 0.00002
4) Epochs in fine-tuning: 4.
5) Similarly as in LSTM, we used Adam optimizer and crossentropy loss.

E. Majority voting ensemble classifier

Lastly, we explain the ensemble classifier that exploits the aforementioned three models and predicts the class with majority voting [14]. The model takes predictions of other models and returns the class that was predicted by the most models. With such ensemble method, we should get a more robust model. Evaluation metric scores could also be increased with such methods.

IV. Results

Finally, we present results using the methodology that we described above. To better understand, how well our models generalize, we used a stratified 10-fold cross validation in all the models. Stratified k-fold cross validation folds are made by preserving the percentage of samples for each class. Using local resources with 6GB GPU, we managed to train 33 different models. The results on full training and testing datasets are further explained in Table I

When training the LSTM model, despite all the precautions we did against overfitting, the model performance still becomes too dependent on the training set after first few epochs as we can see in Figures 5, 6. However, LSTM with early stopping does not overfit as much as the other models. Figure 8 shows, how much the models' performances decrease on new data. We note that LSTM is the most robust in our case, while BERT, the least robust model, yields best scores in all the evaluation metrics. To interpret model's performance we decided to use precision, accuracy, recall and F1 score. While scores in all evaluation metrics are very similar, LSTM scores much better on classifying negative sentiment. These results are further explained in Figure 7.

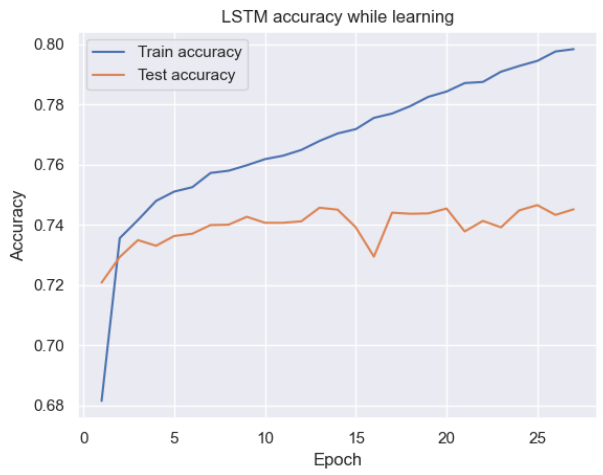

Fig. 5. **LSTM accuracy:** in first few epochs accuracy is rising quickly. However, it becomes more or less constant in median after 5 or 10 epochs, while varying up and down.

V. Discussion

In this section, we will try to interpret results and reason, why such results occur. In the Results section, we noticed that deep neural networks are much more succesful in sentiment analysis in comparison to the baseline model. On the other hand, we need to take into account time complexity of training such models. On a local computer with a 6GB GPU, we managed to gather results in the span of several days. Since in our case, we only needed to train the models once, this issue can be neglected. However, if everyday retraining would be required, we would either need more resources or we would need to use simpler models such as Naive Bayesian classifier.

Algorithm	**Accuracy**		**Precision**		**Recall**		**F1 score**	
	Train	Test	Train	Test	Train	Test	Train	Test
Naive Bayes	0.855	0.733	0.855	0.739	0.857	0.725	0.856	0.732
LSTM	0.810	0.745	0.790	0.730	0.848	0.781	0.818	0.755
BERT	0.951	0.775	0.953	0.783	0.951	0.763	0.952	0.773
Majority classifier	0.896	0.768	0.890	0.765	0.905	0.776	0.897	0.771

TABLE I

RESULTS: THE UPPER TABLE REPRESENTS THE RESULTS OF OUR RESEARCH. FOR ALL THE MODELS, WE ESTIMATED ACCURACY, PRECISION, RECALL AND F1 SCORE BOTH ON TRAINING AND TEST SETS. AS WE CAN SEE, THE BERT MODEL YIELDS BEST RESULTS, BEATING LSTM AND NAIVE BAYES BY QUITE A FEW PERCENTS. THE MAJORITY VOTING ENSEMBLE CLASSIFIER YIELDS COMPARABLE RESULTS. FURTHERMORE, THE LATTER MODEL IS ALSO MUCH MORE ROBUST TO NEW, UNSEEN DATA. COMPARING RESULTS ON THE TRAINING AND TEST DATASETS PROVIDES US WITH AN ESTIMATION OF HOW MUCH THE MODEL OVERFITS.

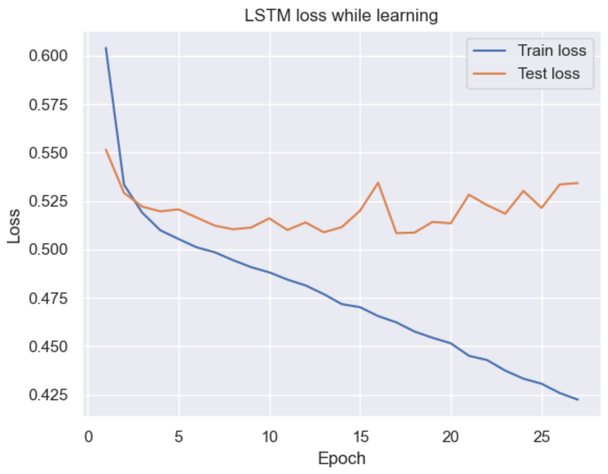

Fig. 6. **LSTM loss:** in first few epochs loss is dropping quickly. However, it becomes more or less constant in median after 5 or 10 epochs, while varying up and down.

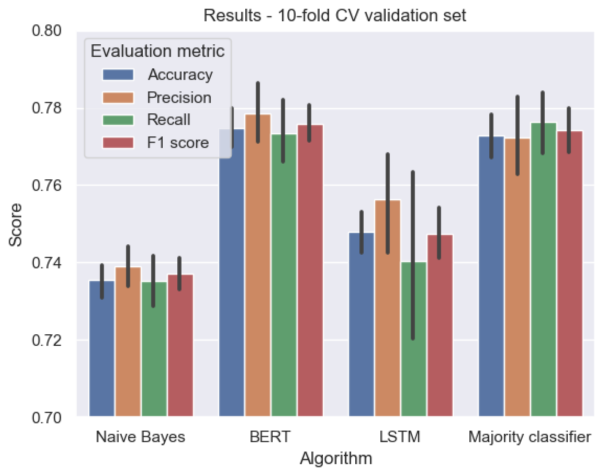

Fig. 7. **Results (10-fold CV - validation sets):** cross validation gives us a better idea of how well the model generalizes. As we can see BERT and majority voting ensemble classifier yield comparable results. LSTM results are slightly better than those of Naive Bayesian classifier. From the figure, we also denote that accuracy, precision, recall and F1 score are almost the same. This makes sense, since the problem is balanced, that is, classes are uniformly distributed. However, we see that the LSTM model returns a gap between precision and recall. Recall is much lower than precision, which means that our LSTM model is weaker in predicting positive sentiment in comparison to predicting negative sentiment. Also note that we can use cross validation to evaluate uncertainty of the model's performance. On barplot, the uncertainty is shown with the error line in the middle of the plots.

Our models could also be further improved by including more data and increasing model complexity. This was not feasible due to available resources. However, this probably affected our neural network models. Since neural networks assume that they will have a representable data subset , we should not be surprised with overfitting. Another reason for overfitting is that tweets are short and they are less likely to differ between samples, since people often copy others and talk about similar stuff. Thus, optimizers can fail to find proper solutions, since a solution can work for a subset of samples and fail for the rest.

As we could see in the Results section, majority of models score similarly well in accuracy, precision, recall and F1 score. LSTM stands out, since recall is much lower than precision and precision is higher than accuracy and F1 score. This means that the LSTM model is much more succesful in evaluating negative context than positive. Recall is also lower in general, which might be due to language used in texts with negative sentiment. Perhaps, evaluating negative sentiment is slightly better. This might be a consequence of complex compounds such as sarcasm and irony.

In comparison to results of related works, we achieve worse results, however, we did not expect the same results, since we used a different dataset. Firstly, tweets are short and thus, more complex models can have problems with overfitting. Secondly, majority of researches were done on movie and product review datasets. The scope of used language used in these texts is much more narrow. Movie and product reviews can also be found on Twitter, thus, datasets used in related works could be interpreted as a sort of subset of Sentiment140 dataset.

As mentioned before, majority voting ensemble classifier

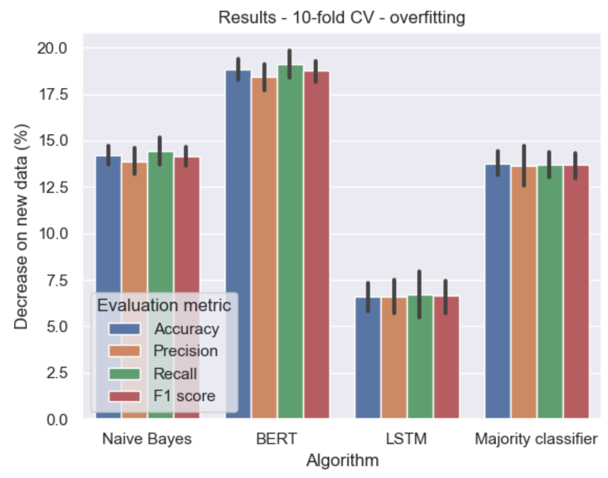

Fig. 8. **Results (10-fold CV - overfitting):** this figure represents an estimation of model's overfitting. We see that BERT overfits the most, dropping almost 20% on all evaluation metric scores on new data. Majority voting classifier increases model's robustness and is comparable to Naive Bayes. The latter, baseline model also drops 15% on all evaluation metric scores on new data. With several precautions, we managed to drop this drop of the LSTM performance on new data to just 6%.

provides us with a combination of results of LSTM and BERT models. Comparing with BERT, it scores comparable in all evaluation metric scores, while being more robust to new data.

To conclude, when choosing the algorithm for sentiment analysis, we would need to take into account the majority class in the training dataset. LSTM performs better on negative sentiment, while worse on positive sentiment. On the other hand, BERT performs better in sentiment analysis classification metrics such as accuracy, precision, recall and F1 score. If we want our model to be more robust, we should take appropriate precautions, when fitting our models. We should use word embeddings, such as GloVe, we should try to get the best complexity-performance ratio, we should lower the learning rate and stop learning the model when we notice overfitting.

VI. Conclusion

To conclude, we found that working with textual data and classifying sentiment is not an easy task. Firstly, we tried to get the best preprocessing methodology, where we tried many combinations. We found that more complex preprocessing methods often yield worse results. Furthermore, we learned that sentiment analysis techniques are very slow, especially when we try to use lots of data, which is required in complex tasks such as this one.

We compared 4 different sentiment analysis approaches and found that neural networks have truly dominated in this area in past years. A fine-tuned BERT and the LSTM architecture both yield better performances than Naive Bayesian classifier. However, as expected the majority voting ensemble classifier returns comparable performance on all evaluation metrics and model's robustness. While the BERT model yields better evaluation metric scores on validation set, it overfits. The model drops almost 20% on scores of all evaluation metrics on new data. On the other hand, we used many precautions on LSTM to prevent overfitting, however, we lower model performance with this. We also noticed differences in performances on classification of negative sentiment versus classification on positive sentiment. With a fine-tuned BERT model, we managed to reach almost 77.5% accuracy, while with LSTM the accuracy dropped to 74.5%.

Finally, we found that neural networks perform better than statistical methods in sentiment analysis classification. In future works, we could try fine-tuning BERT with the LSTM architecture and thus further improving accuracy. Additionally, we could also try to fit the BERT model using the precautions against overfitting that we used on the LSTM model. Another improvement of performance could be done by taking into account a bigger subset of data.

References

[1] K. Denecke and Y. Deng, "Sentiment analysis in medical settings: New opportunities and challenges," *Artificial Intelligence in Medicine*, vol. 64, no. 1, pp. 17–27, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0933365715000299

[2] M. Bhat, M. Qadiri, N.-u.-A. Beg, M. Kundroo, A. N. Ahanger, and B. Agarwal, "Sentiment analysis of social media response on the covid19 outbreak," *Brain, Behavior, and Immunity*, vol. 87, 05 2020.

[3] R. Feldman, "Techniques and applications for sentiment analysis," *Commun. ACM*, vol. 56, p. 82–89, 04 2013.

[4] F. Magliani, T. Fontanini, P. Fornacciari, S. Manicardi, and E. Iotti, "A comparison between preprocessing techniques for sentiment analysis in twitter," 12 2016.

[5] I. Latha, D. Varma, and D. Govardhan, "Preprocessing the informal text for efficient sentiment analysis," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 1, pp. 58–61, 07 2012.

[6] A. Krouska, C. Troussas, and M. Virvou, "The effect of preprocessing techniques on twitter sentiment analysis," 07 2016, pp. 1–5.

[7] S. Zainudin, "Sentiment analysis techniques in recent works," 07 2015.

[8] L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari, "Sentiment analysis of review datasets using naïve bayes' and k-nn classifier," *International Journal of Information Engineering and Electronic Business*, vol. 8, no. 4, p. 54–62, Jul 2016. [Online]. Available: http://dx.doi.org/10.5815/ijieeb.2016.04.07

[9] P. Muhammad, R. Kusumaningrum, and A. Wibowo, "Sentiment analysis using word2vec and long short-term memory (lstm) for indonesian hotel reviews," *Procedia Computer Science*, vol. 179, pp. 728–735, 01 2021.

[10] H. Elfaik and E. H. Nfaoui, "Deep bidirectional lstm network learning-based sentiment analysis for arabic text," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 395–412, 2021. [Online]. Available: https://doi.org/10.1515/jisys-2020-0021

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[12] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *Processing*, vol. 150, 01 2009.

[13] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining." vol. 10, 01 2010.

[14] R. Rodrigues do Carmo, A. M. Lacerda, and D. H. Dalip, "A majority voting approach for sentiment analysis in short texts using topic models," in *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, ser. WebMedia '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 449–455. [Online]. Available: https://doi.org/10.1145/3126858.3126861