Fake News Detection using Snopes and Politifact Data

Vinothini Aravindan
University of Stavanger
Stavanger
Norway

Shaima Ahmad Freja
University of Stavanger
Stavanger
Norway

Yeganeh Hallaj
University of Stavanger
Stavanger
Norway

Aashish Karki
University of Stavanger
Stavanger
Norway

***Abstract*—This project experiments various machine learning models to detect fake news. We use Snopes, Politifact, and their combination to train our classifiers. We did some experiments on some classic classifiers along with LSTM and BERT. We achieved 0.9 accuracy in our best performing methods, which is Decision Tree. Our further analysis indicate that Decision Tree is a good solution to our problem.**

1. Introduction

News is a recent, interesting and significant event which affects a lot of people [1], [2]. Each person follows the recent news on the areas of his/her interest, Therefore, a fake news can influence on the different parts of a society such as economics and politics [3]. By the advent of the internet, news agencies and journalists started to use social media as a platform for publishing their news that lead to the wide spread of fake news [4], [5]. Some of the news agencies and journalists create fake news to either misinform people or to attract more attention for advertising [6]. In addition to the mentioned problems, the news agencies and the social media may be forced to publish a news or to prevent it from spreading due to the governments' pressure. Therefore, there is a need for a fake news detection tool to help people identify the fake news, and ignore them respectively.

In this project we use different classification techniques to detect the fake news. We combined a dataset from Politifact and a dataset from Snopes to train our classification methods and measure their performances. Our individual contribution is as follows:

Preprocessing. : Most parts of the preprocessing of the datasets was done by Aashish and some changes was made in collaboration with Shaima, Vinothini and Yeganah. Further, we had to come up with some changes where we changed the data according to the implementation we performed.

Feature Extraction and Classification. : Vinothini worked on the feature extraction and the classification techniques including different classification algorithms, like Random Forest, Logistic Regression, Decision Tree, Naive Bayes. The SVM was performed by Yeganah. The comparison between all of the classification algorithms were done by Yeganah. In addition, the group members were collaborating each other when implementing different algorithms.

Neural Networks. : The implementation of LSTM and the organization of the code and processes was done by Shaima. Also, the implementation of BERT was done by Aashish and major changes in the code was done in collaboration.

Report. : Report was mostly written by Yeganeh. However, everyone also contributed in writing and gathering different information.

The GitHub Repository link to the code of this project is: **https://github.com/imaashishlk/dat550**

2. Dataset and Evaluation Metrics

2.1. Dataset

We utilize 2 datasets containing news gathered from the Politifact and Snopes websites. Both of these datasets are labeled, and provide a statement and a justification. Therefore, we combined the two datasets and we conducted our experiments on the combination. In this way, we will train our classifiers on a bigger, and more general dataset.

The 2 datasets have 3 important features in common, namely *claim*, *doc*, and *label*. *claim* is the statement which is under investigation, and *doc* is the description that backs up the claim. *label* indicates whether the news is fake or real, and it has a range from True to False. As *claim* and *doc* are highly related, we concatenate them together to form only one column.

The datasets contain other features as well, which we do not consider them in this project, as we deem them irrelevant to our work. These features are as follows:

1) *Factchecker*: The person who fact checked this claim.
2) *Sources*: The original source of the news (e.g. Link to the Facebook post).
3) *Url*: Link to the published version of the news in the website.
4) *Published*: Publication date.
5) *Topic*: A list of key words as the relevant topics.

The Snopes dataset is quite small, containing only 3919 news. On the other hand, the Politifact dataset is larger, and

it contains 18379. The distribution of labels in the Snopes and Politifact dataset is illustrated in Figure 1 and Figure 2 respectively.

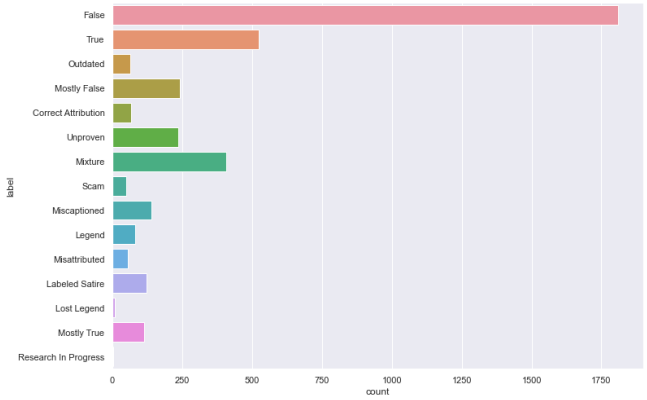

Figure 1. **The distribution of labels in the Snopes dataset.**

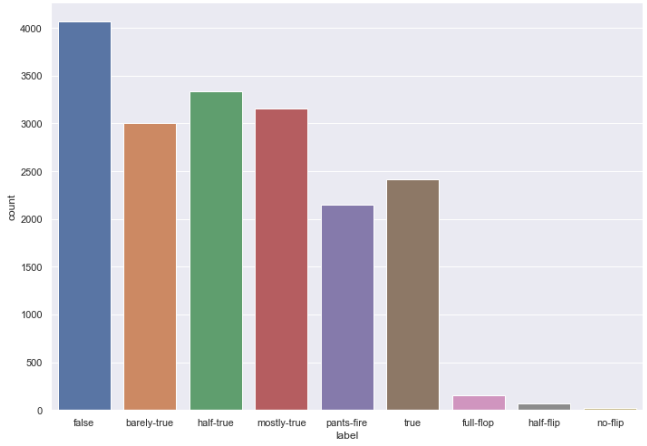

Figure 2. **The distribution of labels in the Politifact dataset.**

As Figure 1 and Figure 2 suggest, while both datasets are labeled with a range between True and False, the number of labels, and also the labels themseleves, are different. Therefore, we needed to alter the labels in a way to achieve uniform labels in the merged dataset. For that regard, as our problem is a binary classification determining the fake and real news, we considered only 2 label, and we changed the labels to only True and False. For example, we considered Both 'True' and 'Highly True' as one label. Figure 3 shows the final distribution of the labels in the merged dataset.

2.2. Experimental Metric

By changing our problem to a binary classification, we can now use the proper metrics for analyzing our results. The first calculate True Positive (TP), False Positive (FP),

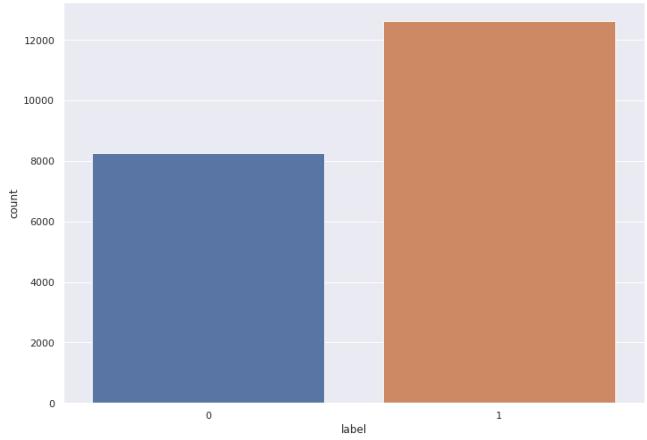

Figure 3. **The distribution of labels in the merged dataset.**

False Negative (FN), and True Negative (TN) for each method. To give a better overview of the following 4 metrics, we illustrate the confusion matrix for each method. Then, we use them to calculate different other metrics such as accuracy, which simply shows how many samples the model predicted correctly. However, sometimes the accuracy is not enough for us to do a proper analysis. For example, having an imbalanced dataset, a method can work well with the accuracy by simply classifying all the samples into a same category. Therefore, we need other metrics for further analysis. We use Precision that calculates how many of the actual positives predicted by the model are actually positive. We also use Recall that shows how many of the samples labeled positive are correctly determined by the model. For having even more analysis over the methods we calculate F1-score from precision and recall, to see how balance the model is in terms of these 2 metrics. Finally, we also plot the ROC curve, and we calculate the Area Under Curve (AUC) for each method.

3. Preprocessing

As shown in Figure 5 We do several steps to preprocess and clean the datasets. As mentioned in section 2, we first merge the Snopes and Politifact datasets. We only consider 3 columns of these datasets: *claim*, *doc*, and *label*. We concatenate the first 2 columns, and we also alter the labels to be either fake or real.

After doing the mentioned steps, we simplify and clean the dataset. Our data cleaning contains three major phases:

1) **Punctuation removal**: In this phase we basically clean the dataset through several steps. We first convert all characters to lower cases. Then, we remove the punctuation characters from the text. We then apply an extra step to make sure we only take the characters into account. Finally, we remove white spaces.

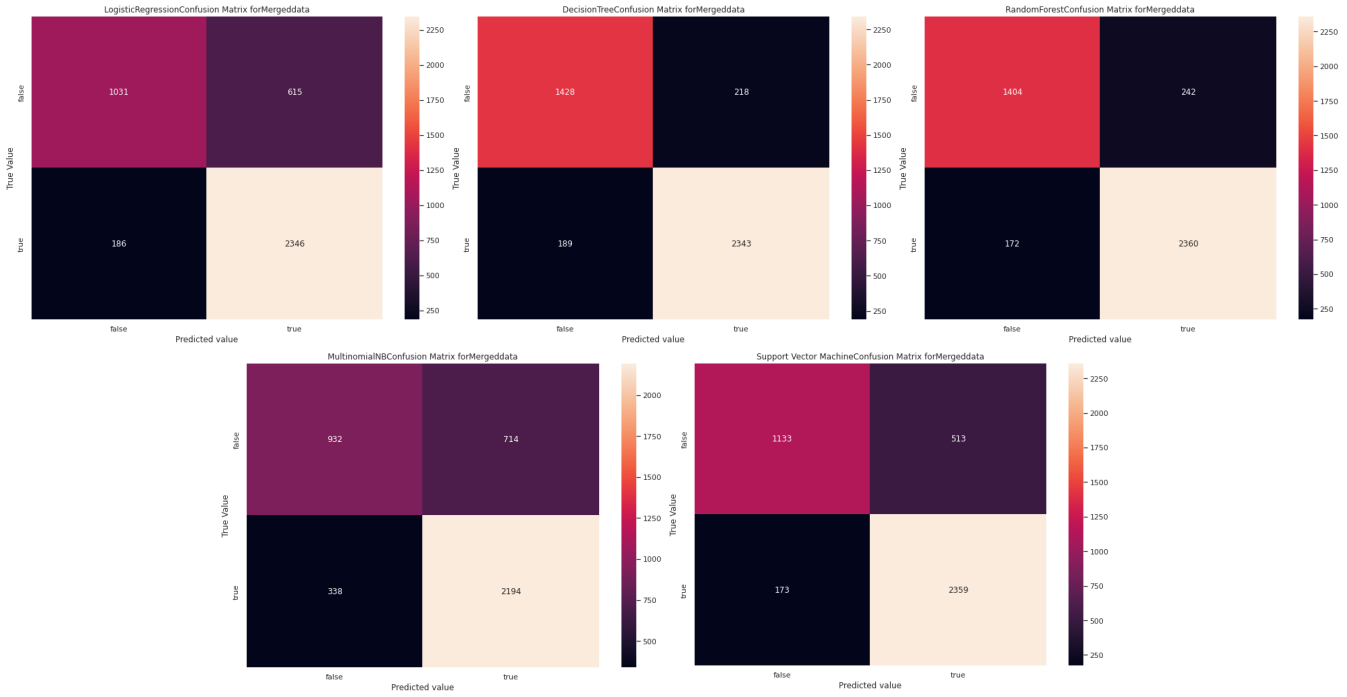

Figure 4. **The confusion matrix for 5 different methods.**

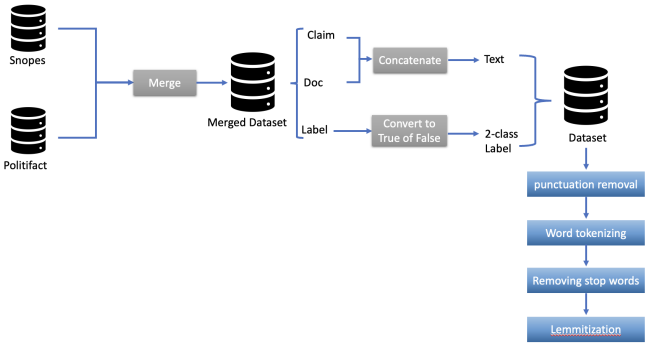

Figure 5. **The preprocessing steps.**

2) **Tokenizing**: Using NLTK library [7] we apply word tokenizer which divides the text into smaller units called tokens. These tokens are words in our case.
3) **Removing stopwords**: After converting the text into tokens, we filter out the stop words by comparing each token to the list of stop words in English. This step is neccecary as the stop words are meaningless words and do not give us any special information. Removing them from the text helps the classification methods to focus on more important features.
4) **Lemmitization**: Lemmatization is used to reduce text redundancy by converting words having the same meaning but different inflected forms to their base form. For example, Lemmatization 1)changes all forms of the verb to the root verb (e.g. changing 'is' to 'be', changing 'working' to 'work'), 2)changes all forms of noun to the base form (e.g. changing 'better' to 'well', changing 'meetings' to 'meeting') [8]. This is very useful for the classification methods as it helps creating efficient and also computationally fast models.

After cleaning and preprocessing the dataset, we split put dataset into train set (80%), and test set (20%).

4. Classic Models

As the first step, we trained some of the classic classifiers on our dataset, and we measured their performances. We used the scikit-learn library [9] in python for implementation of the classic models. In this project, we trained the following models on the dataset:

1) **Logistic Regressing (LR)**: We use Logistic Regressing when there are several explanatory variables. It predicts the most possible outcome by using these variables to model the conditional probabilities [10].
2) **Decision Tree (DT)**: Decision Tree is a machine learning algorithm for classification that recursively splits the dataset based on different parameters till

TABLE 1. DIFFERENT METRICS FOR ALL CLASSIFIERS IN 3 DATASETS. ALL CLASSIFIERS ARE TRAINED USING TFIDFVECTORIZER.

Metric \ Model	Politifact					Snopes					Merged				
	LR	**DT**	**RF**	**NB**	**SVM**	**LR**	**DT**	**RF**	**NB**	**SVM**	**LR**	**DT**	**RF**	**NB**	**SVM**
Accuracy	0.81	**0.94**	0.92	0.76	0.84	0.78	0.76	0.77	0.76	**0.81**	0.81	**0.90**	**0.90**	0.75	0.84
Precision	0.83	**0.93**	**0.93**	0.76	0.85	0.81	0.68	0.79	**0.88**	0.77	0.82	**0.90**	**0.90**	0.74	0.84
Recall	0.75	**0.93**	0.90	0.69	0.79	0.57	**0.67**	0.54	0.51	0.66	0.78	**0.90**	0.89	0.72	0.81
F1-Score	0.77	**0.93**	0.91	0.70	0.81	0.57	0.67	0.52	0.46	**0.69**	0.79	**0.90**	0.89	0.72	0.82

it puts each sample into a class. Decision Tree is supervised, and it needs the labels in order to train [11].

3) **Random Forest (RF)**: Random Forest is a kind of ensemble machine learning which combines different decision trees, and takes the class selected by most of them as the final result.
4) **Naive Bayes (NB)**: Naive Bayes is a kind of probabalistic classifier which is based on Bayes theorem [12].
5) **Support Vector Machine (SVM)**: Support Vector Machine is a supervised algorithm based on statistical learning. SVM is efficent and highly robust, and it is used for both classification and regression.

We trained each of the methods on the train set, and then we applied them on the test set to measure their performance. We applied tfidfVectorizer on the 3 datastets (Politifact, Snopes, Merged) first, and then we trained all models on the result of tfidfVectorizer. Figure 4 shows the confusion matrix for each method trained on the tfidfVectorizer on the merged dataset.

As we cannot compare the methods together with the confusion matrix, we used it to compute other metrics. Table 1 shows the metrics for all trained algorithms on all three datasets.

Looking at Table 1, we see that Decision Tree and Random forest had almost the best accuracy in Politifact and Merged datasets compare to other methods. their F1_score also indicate that they are the most balanced model, and they were able to get high scores in both precision and recall. It shows that these two methods are good options for solving such problem.

The other interesting thing we can see in Table 1 is the importance of calculating different methods. For example, Naive Bayes has an accuracy of 0.76 on Snopes. However, if we look into its recall and precision (0.88 and 0.51 respectively), we understand that this algorithm is only predicting one single class for all samples, and its accuracy is quite high due to the imbalancement of the dataset. When look at its F1_score, we understand that Naive Bayes is not reliable at all.

In it also shown in the Table that the size of the dataset plays an important role in the training process. We see that almost all algorithms preformed better after being trained on Politifact which is a much larger dataset compare to Snopes. The low value of metrics in training on Snopes is simple due to lack of enough data. For example, SVM performed better on Snopes compare to other algorithm, showing that in lacking enough data, SVM could be a good option. Also, we can see that after merging the datasets, the metrics have been decreased. This might be due to adding more complexity to the dataset by merging them, and lack of data for the models to learn such complexity.

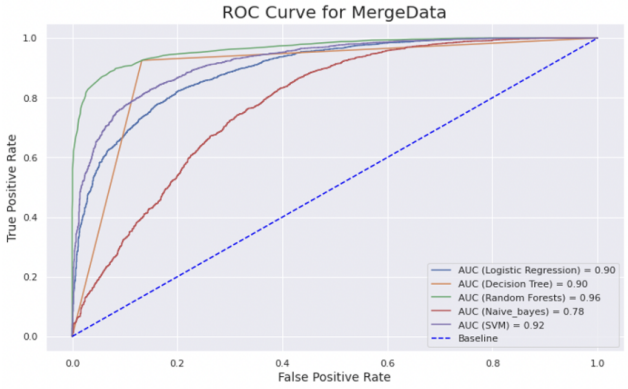

Figure 6. **ROC curve for all methods trained on the merged dataset. The Area Under Curve (AUC) is also calculated per each method.**

To further analyze the methods performances, we plot the ROC curve showed in Figure 6, and we calculated Area Under Curve (AUC). The more AUC, the better the classifier, as it could better distinguish between the classes. We can see from the figure that Random Forest has by far the most AUC compare to other methods. Therefore, we can conclude that overall, RF is the best method for this classification task.

5. Neural Networks

Neural Networks (NN), are a group of machine learning algorithms in which a set of units or nodes are linked together replicating the neurons in a biological brain. Each link may send a signal to other neurons to process, and improve their learning based on the signal. In this way, the network trains on a dataset [13].

In this project, we trained two types of neural networks on our datasets: LSTM [14] and BERT [15]. The reason behind choosing the mentioned method is their capability to work with text datasets. In the following, we will discuss each method in detail.

5.1. LSTM

Long Short Term Memory (LSTM) is an extension of RNN (recurrent neural networks RNN), which handle the

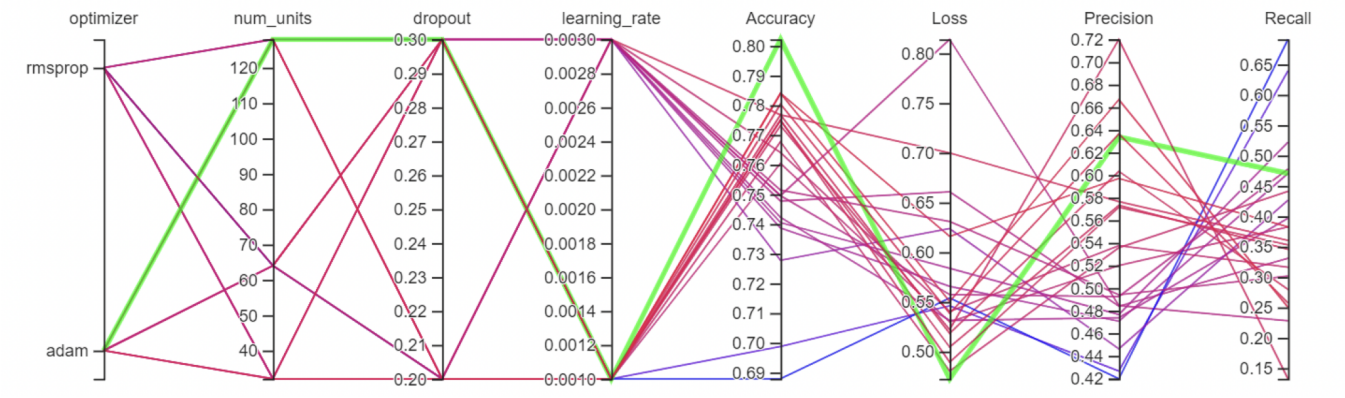

Figure 7. **A sample of using Tensorboard for training the LSTM model on snopes dataset.**

TABLE 2. DIFFERENT METRICS FOR ALL LSTM MODELS IN 3 DATASETS.

Data	Word Embedding	*HyperParameter (epochs: 8, units=64,128, Optimizer:adam, Dropout:(0.2,0.3), Learning_Rate:0.001)*							
		Training				Test			
		Accuracy	Loss	Precision	Recall	Accuracy	Loss	Precision	Recall
Snopes- Preprocessing	Un-Trained	1	0,0001216	1	1	0,7314	1,8921	0,4444	0,3529
Snopes- Preprocessing- GloVe		0,8034	0,4303	0,6509	0,5097	0,7731	0,4819	0,6122	0,2205
PolitiFact -Preprocessing- GloVe		0,7931	0,444	0,7918	0,9312	0,7675	0,4915	0,7863	0,8798
PolitiFact-GloVE	Glove	0,7523	0,5184	0,7536	0,9279	0,7502	0,5287	0,7588	0,8998
Merged-Preprocessing- GloVe		0,7753	0,4781	0,7672	0,9015	0,7517	0,5102	0,7534	0,8775
Merged-GloVe		0,7579	0,515	0,7579	0,9302	0,7435	0,539	0,7343	0,9458

long-term dependency problems (Vanishing gradient). Before feeding the train data into the LSTM model, we need to apply some preprocessing for the text and convert it to numerical to be ready to be fed into the Neural Network model. We implement LSTM Model on separate datasets (Snopes and Politifact) and their merged dataset through the following steps:

Vectorization and Word Embedding:. We convert the text columns to numbers for the model to understand them (by splitting the sentences into words and creating a dictionary of all unique words found in the text and their uniquely assigned integer). Each sentence will be converted to a sequence of word indexes (array of integers representing all the individual words). Finally, we do padding and truncating to make all sequences at the same length. As Word Embedding, we use GloVe Pre-Trained and the untrained word Embedding, which will be trained through the Keras embedding layer.

Building a Machine learning model using LSTM:. We create a sequential object with the following layers:

1) *Embedding layer*: There are two options for creating this layer. We can either depend on the weights in (Glove Embedding Matrix) or form new weights by training the LSTM model from scratch.
2) *LSTM*: We use a bidirectional LSTM layer. We change the number of units as one hyper-parameter.
3) *Hidden Layer*: We use one Dense layer with the number of units and activation function detected from hyper-parameters.
4) *Output Layer*: contains 1 unit (0 or 1) for the binary classifier, and we use Sigmoid as this layer activation function.

Training the model and evaluating.:. We train the model by dividing the data into "batches," which equates to the value that we determine in batch_size, and repeatedly iterating over the entire set for a given number of epochs. Some important arguments we use in training:
1. Validation_data: Besides having a train and test set, we also divide the train set into train and validation. After training in each step, the model iterates over the validation dataset and computes the validation loss and validation metrics (accuracy, precision, recall)
2. We use Callbacks, an object that performs processes through training operations like Early Stopping, which will stop training when the chosen performance measure like 'loss_val' stops improving.
3. We also use TensorBoard callback after every batch of training to monitor hyperparameter metrics. Figure 7shows an example of the TensorBoard output in hyper-parameters when training the model on Snopes dataset. It is evident from the figure that we set several things as the hyper-parameters, and we can check which one worked better.
4. CSVLogger: We use it to save the CSV file, which contains all metrics values.
5. Evaluation: We use this module to check whether the model is the best fit for the problem statement and estimate the general accuracy of the model.

As mentioned before, we trained the model with different hyper-parameters on three different datasets, and we

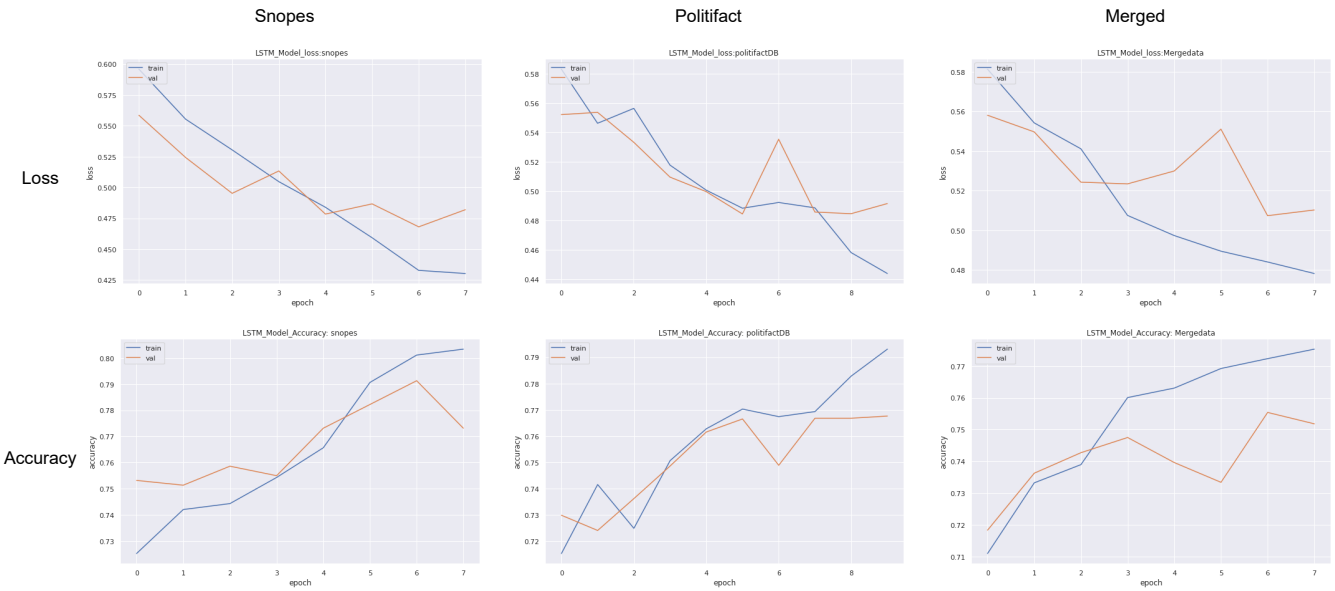

Figure 8. **Loss and accuracy plots for the LSTM model trained on 3 different datasets.**

used the early stopping technique to stop training when a monitored metric has stopped improving. Therefore, we can increase the epochs number and the fitting stops automatically when there are no improvements. The loss and accuracy functions of the model while being trained on the three datasets is shown in Figure 8. By comparing the loss and accuracy plots, we can see from the figure that the model were going to get overfit on Snopes dataset, as accuracy increased in training, but dropped in validation. This might be due to the low number of samples in Snopes. We also can see that the model achieved better accuracy in Snopes compare to other datasets. To have a better comparison, we calculate different metrics for each dataset, shown in Table 2. Also, Figure 9 compares all methods together in terms of accuracy, loss, recall, and precision.

Without using GloVe the model got the accuracy of almost 1 on the Snopes train set, but when testing it on the test set, the accuracy dropped significantly to 0.73. It shows that the model is over-fitted on the Snopes dataset. This is because the dataset is too small, and the LSTM is too deep for such small dataset. It shows the importance of using GloVe, as it solves the problem in some extent, but the problem still exsists due to the size of the dataset.

We can see that the LSTM model performed almost the same on the three datastes in terms of the accuracy. However, if we look at the precision and recall, we can see that the model is trained poorly on the Snopes, and it only predicts one class for all the samples, leading to a very low value in precision and recall. The reason again is because of the low samples and imbanalncement of the Snopes dataset.

We can also see that merging the two datasets has a good impact on the LSTM models, as it improves the recall significantly. Also, it achieves the highest balance between precision and recall, and the accuracy is acceptable.

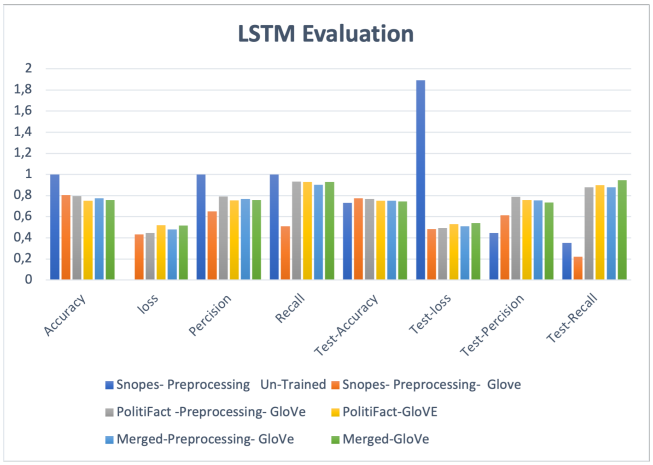

Figure 9. **Comparison of different models of LSTM trained on 3 different datasets in terms of accuracy, loss, recall, and precision.**

5.2. BERT

Bidirectional Encoder Representations from Transformers (BERT) is a NLP technique that computes vector-space representations of natural languages. It processes the each token of input text in the full context of all tokens before and after by using the transformer encoder. Two most important applications of BERT include Pre-training and fine-tuning, as BERT is already pre-trained on a large corpus of text, then fine-tuned for specific tasks. BERT main purpose is

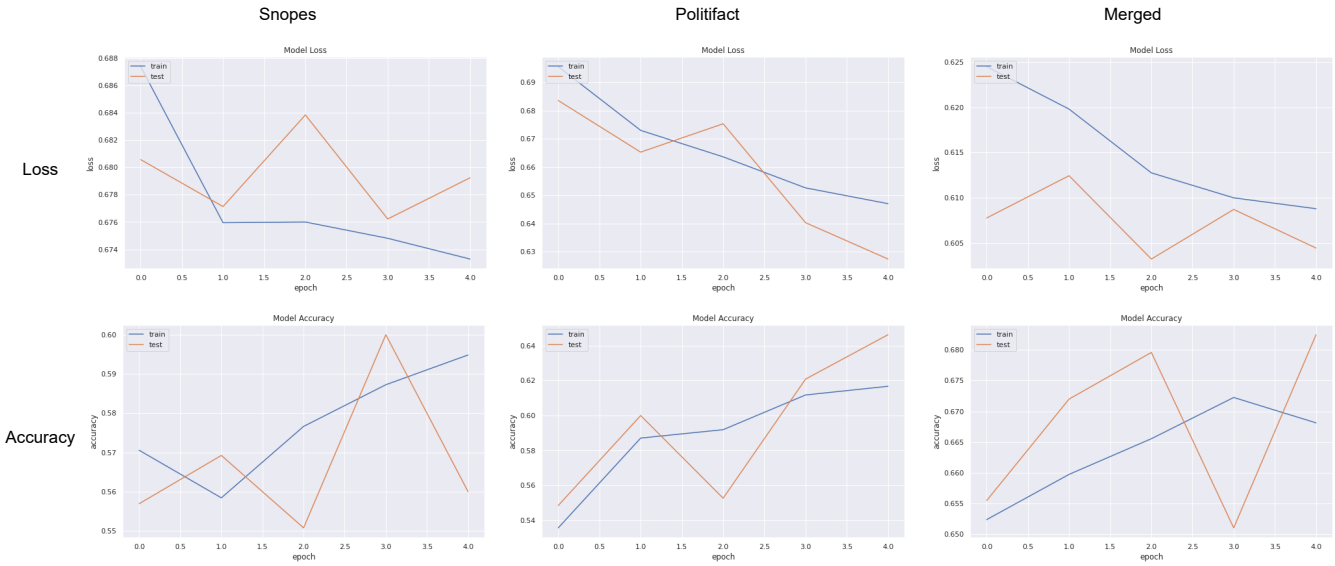

Figure 10. **Loss and accuracy plots for the BERT model trained on 3 different datasets.**

TABLE 3. DIFFERENT METRICS FOR BERT TRAINED ON 3 DATASETS.

Metric / Model	Accuracy	Precision	Recall	F1-score
BERT on Politifact	0.65	0.70	0.65	0.63
BERT on Snopes	0.57	0.61	0.57	0.53
BERT on Merged	0.70	0.72	0.70	0.69

to build a vector representation from the text, and it only employs an encoder for this purpose.

We used L-12_H-768 version for BERT. For using BERT, we first balanced the data by sampling an equal number from both the positive and negative labels. Then we trained BERT on 5 epochs. Figure 10 shows the loss and accuracy function of BERT trained on 3 datasets.

To see the performance of BERT models trained on each dataset, we measure different parameters shown in Table 3. We can see that merging the datasets improved all the metrics significantly. The reason is that BERT needs a large dataset to be trained on, and merging the datasets increased the number of samples, and helped BERT to train better. This is also the reason that BERT works poorly on the Snopes dataset.

While being only trained for 5 epochs, BERT achieved a decent accuracy of 0.70 on the merged dataset, and the F1-score shows that it is quite balanced in terms of both recall and precision.

6. Conclusion

In this project we developed and trained different classifiers on the news dataset to detect fake news. We conducted our experiments on 3 different datasets: Snopes, Politifact, and their combination. We first tried using some classic classifiers such as Logistic Regression, Decision Tree, Random Forest, Naive Bayes, and SVM. Then we tried two well-known methods for text learning: LSTM and BERT.

Overall we can see that Decision Tree and Random Forest are the best solution to this problem, and they achieved a great performance. We can conclude that as the dataset is small, deep neural networks are not a good solution, as they need a huge amount of data to be trained on. We see that LSTM and specially BERT does not have a good performance on these datasets, because they both are complicated networks needing large datasets to learn useful information. Classic classifiers such as Decision Tree can perform well on small and simple datasets.

As the first step in our future work, we need to gather more samples to add to our datasets. One way to do it is to have a crawler to gather information from the news websites. Other thing we can do in the future is to train our implemented BERT for more epochs. Currently, due to lack of enough time, we trained BERT for only 5 epochs on each dataset, as it requires a lot of time to get trained. This can be a reason for its low performance. In addition, we will investigate other methods such as Convolutional Neural Networks (CNN), and will compare them to LSTM.

References

[1] J. W. Kershner, *The elements of news writing*. Pearson Allyn and Bacon Boston, 2005.

[2] B. Richardson, *The process of writing news: From information to story*. Allyn & Bacon, 2007.

[3] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[4] A. Hermida, F. Fletcher, D. Korell, and D. Logan, "Share, like, recommend: Decoding the social media news consumer," *Journalism studies*, vol. 13, no. 5-6, pp. 815–824, 2012.

[5] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[6] E. C. Tandoc Jr, Z. W. Lim, and R. Ling, "Defining "fake news" a typology of scholarly definitions," *Digital journalism*, vol. 6, no. 2, pp. 137–153, 2018.

[7] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[8] B. Fardeen, By, , A. F. i. a. fardeen, A. Fardeen, and T. i. a. fardeen, "Learn lemmatization in ntlk with examples," Apr 2021. [Online]. Available: https://machinelearningknowledge.ai/learn-lemmatization-in-ntlk-with-examples/Stemming_vs_Lemmatization_Example

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[10] S. Sperandei, "Understanding logistic regression analysis," *Biochemia medica*, vol. 24, no. 1, pp. 12–18, 2014.

[11] M. Kulkarni, "Decision trees for classification: A machine learning algorithm," *The Xoriant*, 2017.

[12] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[13] C. M. Bishop, "Neural networks and their applications," *Review of scientific instruments*, vol. 65, no. 6, pp. 1803–1832, 1994.

[14] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.