Unreliable News Classification Using Machine Learning Methods

A0225551L, A0131114A, A0148048B, A0194499E, A0182884N
Group 16
Mentored by Miao Yisong
{e0576185,e0696697,e0012720,e0376929,e0309679}@u.nus.edu

Abstract

The influence of fake news has spread across the globe. Fake news is frequently propagated through the use of fake news websites, which specialize in providing attention-grabbing headlines and often resemble well-known news sources in order to gain credibility. The spread of unreliable information is a well studied issue and it is associated with negative social impacts. For example, during this pandemic, fake news about the vaccination being lethal has caused adverse effects on public health. Many people chose not to be vaccinated and caused implications on country's healthcare system. Hence, having a good model to detect such fake news could help to alleviate some of it negative impact on the society.

Hence, this paper has explored the use of different existing text classification models[1], such as Linear Classifiers and Deep Learning methods to predict the reliability of news articles into four main categories - Satire, Hoax, Propaganda and Reliable News. The paper has also discovered the common properties of unreliable news such as negative bigram pair counts. With the underlying differences in the properties of text data, the paper has attempted to debunk the common myth that state-of-the-art methods (i.e. Deep Learning) is always superior to Traditional Machine Learning models in its performance for such classification task.

1 Introduction

With the prevalence of online media in this digital age, there is a rapid increase in the amount of disinformation present in the web. As online media transcends international boundaries, the negative social impact of such information have been felt across the world.

One most recent example of this is the COVID-19 pandemic. In United States[2], almost 80 percent of consumers have reported having seen fake news on the coronavirus outbreak. As the journal states, unreliable information such as the lethality of the COVID-19 vaccine has led to many rejecting vaccination and has significantly hindered public health efforts to curb the pandemic [3].

Beyond the immediate goals of completing the project, the team have recognized the importance of finding a text classification model to accurately identify such disinformation and put a stop to the spread of unreliable news. The team have experimented with different text classification models on real-world dataset [4] – from classic models (i.e. Multinomial Naive Bayes' Classifier and Linear Support Vector Classifier) to state-of-the-art models (i.e. Long short-term memory). The Multinomial Naive Bayes' classifier was selected as the baseline model and mainly used to select the appropriate pre-processing methods for the corpus. Ultimately, distinguishing characteristics from unreliable news were discovered through exploratory data analysis and the affects of such characteristics were used in measuring the comparative performance between the models.

2 Related Work / Background

There are four main approaches to classify the news which are based on existing knowledge (knowledge-based), the news' content (style-based), how it propagates (propagation-based) and the source of the news (source-based) respectively. Since only content data are available, a style-based approach was used. Typically, there are two common machine learning approaches to the fake news problem, namely using supervised traditional ML-based models and deep learning models[5]. For a traditional ML-based model, news content is represented using features at various language levels (lexicon, syntax, semantic and discourse). Latent features includes word2vec while non-latent features include BOWs, Ngram and POS tags[5]. Supervised classifiers being used frequently are SVM, Naive Bayes[6], Random Forests(RF)[7], and KNN[8]. Of all the models, SVM shines in

many aspects such as: simple structure, high adaptability, global optimization, short training time and good generalization performance[8]. First, the dataset is compressed to support vector set, then new knowledge is gained by learning to use the subset and also the rules decided by support vector machines are given. With the same experiment setup, the accuracy, recall and f1 score of SVMs outperform KNN and Naive Bayes[8].

Deep learning based neural network models have achieved great success in many NLP tasks, including learning distributed word, sentence and document representation. For deep learning based models, the content is often turned into an embedding before feeding it into the model. Common models include well-trained neural network such as VGG-16/19, text CNN and RNN such as LSTM and GRUs[5]. Among these choices, LSTM stands out because of its ability to preserve information from the older steps. LSTM can be used for different kind of text classification for example in Rao's work[9], LSTM is used to classify text as actionable/not actionable feedback from a customer service providers and political leaning which is democratic/republican. It achieved a high accuracy of 87.57 accuracy. Thus, the team used LSTM as the deep learning approach to solve the problem.

3 Corpus Analysis & Method

Before delving into experimenting with different machine learning algorithms, the team performed exploratory data analysis to understand the data better. After that, an incremental and iterative approach was used to build and test the solutions in a structured manner.

3.1 Data distribution

As shown in figure 1, the distribution of labels are Satire(Fake): 14047, Hoax: 6942, Propaganda: 17870, Trusted(Real): 9995 records respectively. Thus, the dataset is rather imbalanced and trusted news occupies only a small portion, 20.5%. There are no rows with any null value were found in the dataset.

3.2 Word count analysis

As shown in the figure 2, each category share a similar range of word counts. Thus, no useful conclusion can be deduced by solely looking at the word count. Most of the news fall under the 0-2000 words range with some outliers in each category.

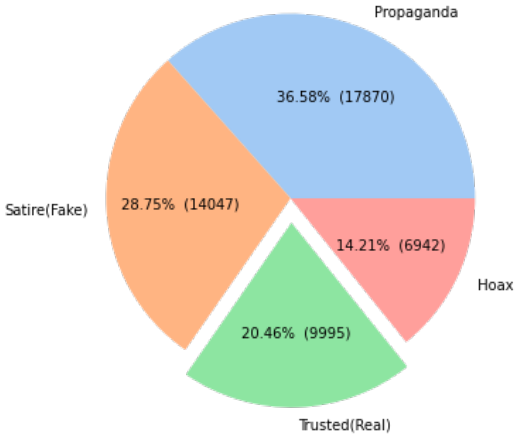
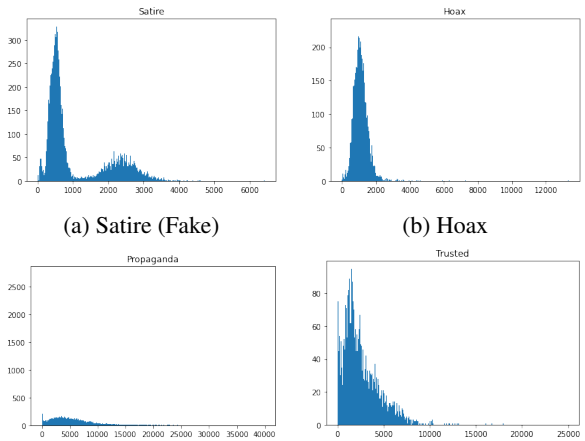


Figure 1: Data Distribution

For trusted news and propaganda, many entries are in the 2000-5000 range. One notable finding from this analysis was that propaganda news seem to have the lowest word counts overall, but this information is not conclusive enough as other categories of news can also be in the same range.



(a) Satire (Fake) (b) Hoax

(c) Propaganda (d) Trusted (Real)

Figure 2: Word Count Analysis

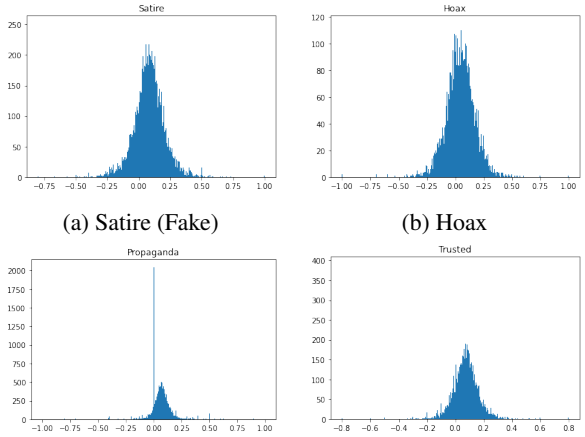
3.3 Sentiment analysis

Figure 3 shows that all classes have relatively neutral sentiment within the range of -0.1 to 0.1. This is expected because news are supposed to be fact based regardless whether it is real or fake as opposed to novel/screenplay. Thus, sentiment analysis is not very helpful in this case.

3.4 N-gram Analysis

3.4.1 Satire (14047 samples)

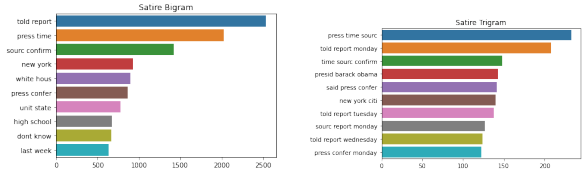
As appears in figure 4, satire news seem to be using words like 'based on recent report', 'source



(a) Satire (Fake) (b) Hoax

(c) Propaganda (d) Trusted (Real)

Figure 3: Sentiment Analysis

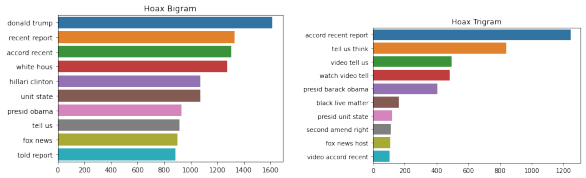


(a) Bi-gram Analysis (b) Tri-gram Analysis

Figure 4: N-gram Analysis - Satire

confirm' to reinforce the credibility. Another interesting fact is that most of the reports happen on Monday. Words like 'Time source confirm' were used to mimic that the records are from credible news agency like NY times as well.

3.4.2 Hoax (6942 samples)

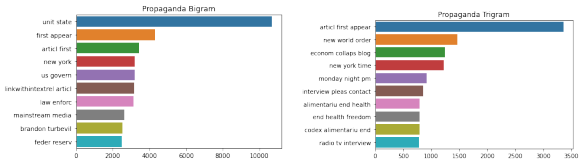


(a) Bi-gram Analysis (b) Tri-gram Analysis

Figure 5: N-gram Analysis - Hoax

Many entries in Hoax category mentioned *Obama* and *Trump*, almost every single article contains one(8645 times out of 6942 samples). Fox news seems to be the primary source of information as it appears a lot as well. Interestingly, academics, media figures, political figures, and watchdog groups has claimed Fox News to have a bias in favor of the Republican Party in its news coverage. On top of that, they have been misleading their audience in science topics, one such example is climate change.

3.4.3 Propaganda (17870 samples)

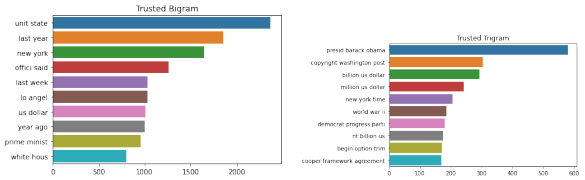


(a) Bi-gram Analysis (b) Tri-gram Analysis

Figure 6: N-gram Analysis - Propaganda

Figure 6 shows that United States is undoubtedly the most frequent two words phrase in the category. This is not surprising at all because the the intention of propaganda usually is to spread information in favour of government or political party. 58000 out of 17970 sample. Another not so obvious characteristic is the phrase new world order which is definitely a common word in typical news. It seems to be promoting a certain ideology which coincides with what propaganda is trying to do.

3.4.4 Trusted (9995 samples)



(a) Bi-gram Analysis (b) Tri-gram Analysis

Figure 7: N-gram Analysis - Trusted

The sources for trusted news seems to be from *Washington Post*, as copyright *Washington Post* appear very often. This can be very useful in deciding whether the news is trusted by simply looking at its source. On top of that, phrases like 'million/billion dollars' are very common. It can be inferred that maybe the news are talking about policy or grant amount. It might be a sign of real news because they are giving concrete numbers which is rare in fake news because they can be easily debunked.

4 Experiments

Based on the literature review in section 2, a few models which were commonly used in text classification tasks were hand-picked. The selected models were Multinomial Naive Bayes(NB) classifier, Linear Support Vector Classifier, and Neural Network model with LSTM layers.

The models were selected based on their progressive performance and complexity, starting from

the basic model (i.e. Multinomial NB) to a more complex model(i.e. LSTM). The first 2 models are linear classifiers while Model 3 involves deep learning. the team attempted to study the comparative performance between the linear classifiers and neural network. The Multinomial NB classifier was chosen as the baseline model because the model offers a rudimentary approach to text classification, which is based on the naive assumption that considers conditional independence between words.

Apart from the baseline model, the team performed a range of feature engineering steps for each individual model to attain the model with best possible performance. From there, the models were compared using their performance to discover the qualities of the best performing model that allows such accurate classification of text data. For model training, a common method used to split the data is StratifiedShuffleSplit, which is a merge of StratifiedKFold and ShuffleSplit. It returns stratified randomized folds where the folds are made by preserving the percentage of samples for each class. The data is split into training set(60%), validation set(20%) and testing set(20%). A seed is set to ensure that the data has been split the same way for all the models. F1 macro score was used as the performance metric in model training. It is considered as a more suitable metric for dataset which are imbalance. It is also a commonly used metric for such a classification task.

4.1 Multinomial Naive Bayes Classifier

Naive Bayes is a straightforward method for building classifiers, which are models that give class labels to problem cases represented as vectors of feature values, with the class labels selected from a limited set. All Naive Bayes classifiers assume that the value of one feature is independent of the value of any other feature, given the class variable. Due to that assumption, Naive Bayes classifiers perform poorly for some of the cases. However, it is highly scalable and only requires a number of parameters linear to the number of variables for training. Hence, it was selected as a baseline model.

As there were 4 classes in this problem, a random guesser should be able to achieve 25% accuracy at the very least. Thus, any machine learning approach to this problem should do better than a random guesser. As shown in table 1, a few combination of pre-processing steps were used with TF-IDF features to train a baseline model as well as selecting a most promising pre-processing steps for the future iteration.

4.2 Pre-processing

The commonly used pre-processig techniques in NLP classification tasks are using lowercasing, stop words removal, punctuation removal, contractions expansion, lemmatization, and stemming [10]. Those methods were tested on Naive Bayes, SVC and LSTM models. Based on our evaluation (macro F1 score), using lowercase can improve all three models' performance. In the original dataset, the mixture of lowercase and uppercase adds extra noise in the model training process.

However, when stop word removal and lemmatization were applied in pre-processing steps, the prediction accuracy of all three models dropped. It is likely due to the small size of training data, when the words are changed into their original forms, there are limited information models can learn from the training data set especially for those more powerful models, like SVC and LSTM [11]. This phenomenon also appeared in some other NLP classification projects. Based Pradana and Hayaty's work, stop word removal and stem also negatively affect their models' accuracy [12]. As a consequence, only lower case will be applied in pre-processing step for later experiments.

pre-processing method	accuracy	F1(macro)
No preprocessing	0.9092	0.9050
lowercase	0.9139	0.9090
lowercase stopwords removal	0.9059	0.8999
lowercase lemmatization	0.9042	0.8992
lowercase stopwords removal lemmatization	0.9054	0.8984

Table 1: Results of Naive Bayes with TF-IDF

4.3 Linear Support Vector Classifier

4.3.1 Methodology

The linear support vector classifier, which belongs to a class of Support Vector Machines is capable of performing binary and multi-class classification on a dataset. It is commonly cited in literature reviews[13] as one of the simpler yet better performing model for the text classification task. It has a remarkable property that allows them to learn independently of the feature space dimensionalities. The support vector classifier is able to measure the complexity of hypotheses based on the margin with which they separate the data, not the number of features. This means that that model can generalize even in the presence of very many features,

if the data is separable with a wide margin using functions from the hypothesis space.

To test the performance of this model, the data pre-processing steps mentioned in the earlier section were used to find the best feature engineering steps that can produce the best results (i.e. best model performance).

4.3.2 Results

		F1(macro)
Type of Vectorizers	TF-IDF Vectorizer	0.97755
	CountVectorizer	0.95249
TF-IDF with Ngrams	Unigram	0.97755
	Unigram & Bigram	0.97873

Table 2: Types of Feature Engineering with Linear SVC

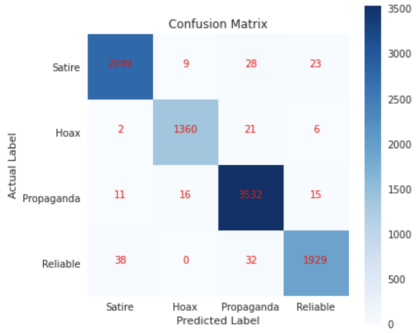


Figure 8: Confusion Matrix of SVC with TF-IDF and Ngrams(Unigram & Bigram)

The Linear SVC model was tested on each of the different feature engineering steps in Table 2. The result shows that TF-IDF with unigrams and bigrams produces the best results, with most of documents accurately categorized(Fig. 8). There are several reasons why this might be so. Firstly, TF-IDF vectorizer considers words that are in abundance in a corpus as less important and gives more weight to less occurring words. This would have captured more important features in the model that could possibly help in the classification task.

On the other hand, Count Vectorizer only considers the frequency of words, it is unable to identify their relative importance. It will simply consider words that are abundant in a corpus as the most statistically significant word even though it may not be an important feature in the model. On top of that, tokenizing the words into both unigrams and bigrams, instead of just unigrams increases the number of features in the model. Lastly, Bigrams can provide more context to a word.

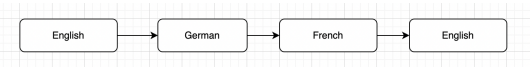


Figure 9: Generating more data by translating

With this, it can be observed that Linear SVC model does indeed perform extremely well in the text classification task. In the next section, the SVC's model's performance was compared against a state-of-the art Neural Network model.

4.4 LSTM

After introduced by Sepp Hochreiter *et al* in 1997, Long short-term memory (LSTM) has recently become a popular tool among NLP researchers for their superior ability to model and learn from sequential data [14]. LSTM aims to solve the RNN problem called gradient vanishing and exploding.

4.4.1 Methodology

The feature engineering for the LSTM model implemented was different from other models. Compared to CNN models, LSTM does not have strong feature engineering capabilities [15]. However, compared to simple linear models like the Naive Bayes and Linear Support Vector model, LSTM has a strong capability of sequential processing. Thus, the feature engineering of LSTM must be implemented carefully.

Feature Engineering

Many feature engineering techniques were tested including sampling, word vectorizer, word embedding as well as some other self-defined methods.

1. *Sampling*
   Multiple sampling methods are implemented and tested in order to tackle the imbalance of the data. Figure 1 in the *Corpus Analysis & Method* section above shows that samples in the class *Propaganda* is about twice in number of samples in the class *Real*. Upsampling and downsampling are implemented but the macro f1 score dropped by 0.05. Some other self-defined sampling methods are also implemented, such as translating the data into German, French, and then back into English. This translation pipeline is applied on minor classes. The translation pipeline is demonstrated in the below Figure 9.

   Based on the analysis, it was found that the reason for sampling did not well as intended

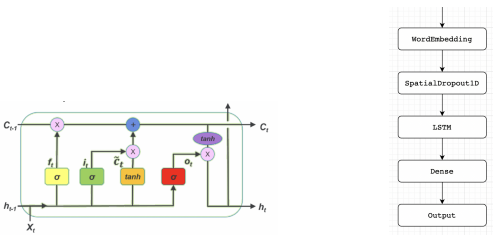
was having the bias in data. Since the data were separated in the way that maintain the original data distribution in the train set and test set, the distribution of data in test set is naturally imbalanced. This bias is utilized by the model in the form of weights in training.

2. *Word Vectorizer VS Word Embedding*
   Word embedding was chosen as the main feature engineering technique other than word vectorizer. For a small and domain specific data set with a simple linear classifier, word vectorizer may work better than word embedding. However, in this case, with powerful LSTM model, the hidden information in pre-trained word embedding models works better.

3. *Some other self-defined feature engineering techniques*
   The team also tried to manually pick some features like the length of the sample, the number of capital letters, the number of punctuation marks and etc. None of them can produce a better F1 score and they are abandoned in the final LSTM model.

LSTM replaces hidden vectors from recurrent neural networks with memory blocks equipped with gates [16]. This can maintain long-term memory in principle by practicing appropriate gating weights and has proven to be very useful in achieving state-of-the-art for various problems, including speech recognition [17]. LSTM stores separate memory cells in it which can update and display their contents only if necessary [18].

The LSTM model the team built consists of 5 layers, a Word Embedding layer, a Spatial Dropout layer, a LSTM layer, a Dense layer and lastly the output layer as Figure 10b shows.



(a) LSTM Architecture by Sari *et al* [19]

(b) The LSTM Model Structure implemented

Figure 10: The LSTM architecture and model structure

Multiple parameters sets are tested as below Figure 11 demonstrated.

Model	F1 Score	Neuron	Learning Rate	Optimizer	Loss Function	Activation Hidden	Function output	Dimension
1	10	100	0.001	Adam	Categorical Cross Entropy	Tanh	Softmax	300
2	10	100	0.0001	Adam	Categorical Cross Entropy	Tanh	Softmax	300
3	10	100	0.001	Adam	Categorical Cross Entropy	Relu	Softmax	300
4	10	100	0.0001	Adam	Categorical Cross Entropy	Relu	Softmax	300
5	10	100	0.0001	RMSProp	Categorical Cross Entropy	Tanh	Softmax	300

Figure 11: Training Models LSTM with word embedding

4.4.2 Result

Inspired by Winda Kurnia Sari *et al*'s paper Text Classification Using Long Short-Term Memory with GloVe Features [19], the hyper-parameters used are the Relu and Tanh activation functions, Adam and RMSProp optimizers will be validated with a learning rate of 0.001 and 0.0001 to minimize error.

The training and validation loss history of the model LSTM with parameters as activation=tanh and lr=1e-3 is demonstrated in Figure 12a.

Inspired by Winda Kurnia Sari *et al*'s paper [19] as well as the actual performance on our dataset, learning rate = 1e-4 was chosen as the final parameter to make the training process more smooth. The above Figure 12a and Figure 12b shows the difference.

Tanh works better than relu as activation function of the LSTM layer. Severe vanishing gradient problem was encountered when relu was used as the activation function at the LSTM layers as shown in the above Figure 12b and Figure 12c.

The macro F1 score of the LSTM model with different parameters sets are listed below in Figure 13.

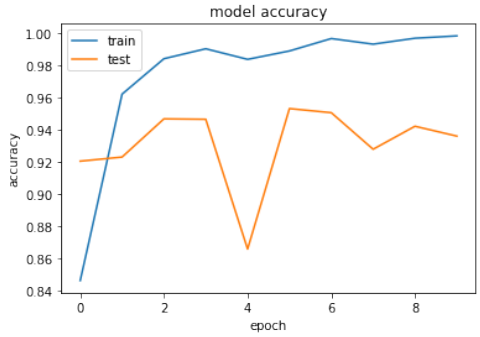
In addition to the hyper parameters, LSTM with additional drop out layers are also tested on model No. 1. The results are listed in Table 3.

		F1(macro)
Without Dropout layers	Without Dropout layers	0.94689
	With Dropout layers	0.95249

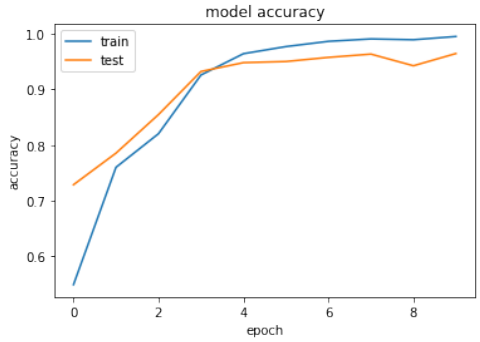
Table 3: F1-score Comparison for LSTM without/with Dropuout layer between the dense layers on Model No.1

5 Discussion

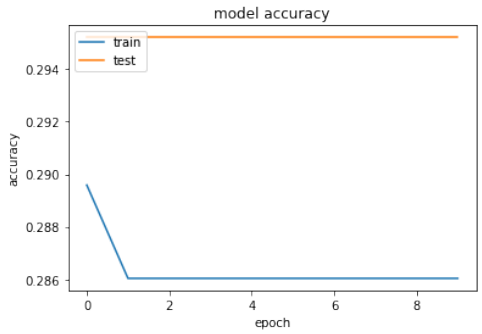
Based on the experiments from previous sections, the simple model like Linear SVC is able to outperform complex models like LSTM in different aspects. Refer to table 4.



(a) Accuracy of LSTM with tanh as activation and lr=1e-3



(b) Accuracy of LSTM with tanh as activation and lr=1e-4



(c) Vanishing gradient problem with relu as activation function at LSTM layer and lr=1e-4

Figure 12: Training and validation accuracy for different parameters sets

	SVC	**LSTM**
F1(macro) score	0.97873	0.95249

Table 4: Model Comparison

Firstly, a stark difference was observed in the runtime between Linear SVC and LSTM. A heavy duty hardware(e.g. CPU/GPU) is needed to accelerate model training for LSTM, whereas it is not required for Linear SVC model. Taking the google colab as an instance, average runtime of the LSTM model is about sixty minutes for a ten-epoch training, while the Linear SVC model took about four minutes.

Model	Test F1 Score
1	0.94689
2	0.95389
3	0.11235
4	0.17743
5	0.93531

Figure 13: F1 score of LSTM with word embedding

When the performance of the models were compared (Table 4), it was observed that the Linear Support Vector Classifier performs better in classifying the documents than LSTM. This is because SVC is able to take into account the of important properties for text[13]:

1. **High dimensional input space**: Text data usually consists of many features, resulting in a huge feature space. However, SVC uses overfitting protection, which does not necessarily depend on the number of features [20]. Therefore, they have the capability to handle the high dimensional input space.

2. **Few irrelevant features**: In text data, all the features contain useful and somewhat relevant information. It is difficult to do any aggressive feature selection on a few features and ignore others. This may potentially lead to a loss of information. Therefore, a good classifier like SVC is able to combine many features and learn a "dense" concept, retaining most of the information from the text without having to perform much feature selection [21].

3. **Document vectors are sparse**: Without performing any smoothing techniques, Documents are typically sparse, with only a few non-zero entries. Kivinen et al. [22] give both theoretical and empirical evidence for the error bound model that "additive" algorithms with similar inductive bias like SVC, are well suited for problems that involves dense concepts and sparse instances.

4. **Text categorization problems are mostly linearly separable**: Features in text data are usually linearly separable [23]. Therefore, making it easier for SVC to find linear separators.

5.1 Limitations

The hidden bias in the data and future unbiased training

Some deeper reasons why a piece of text (news) can be classified by the machine learning models were discussed by the team. Traditional news checking systems always use other additional related information to check the reliability of the news [24] [25]. Unlike the common system which crawling for additional information of the given topic on the Internet, the models were trained without any other related information, which may have violated the principal of fake news detection [26].

Initially, the team assumed that the word2vec embedding already contains the real world information and the neural network utilizes those extra information to make classification. However, the assumption was disproved after a contrast experiment was conducted on the Linear Support Vector Classifier model without any word2vec embedding. The model still can achieve about 0.98 with only simple naive unigram as input.

Inspired by the ngrams analysis mentioned above in section 3.4, the team focused on finding the idiosyncrasies in the data. More specific ngrams of each classes are examined and the truth reveals. The below Figure 14 shows how the negation word 'not' bias the four classes.

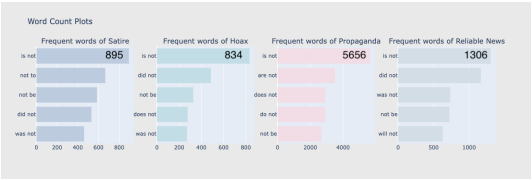


Figure 14: negation word 'not' counts of four classes

The explanation in linguistics and journalism is editors who write fake news tend to use more strong phrases like negation to emphasize the tones and appeal readers compared to editors who write real news [27] [28]. This is consistent with the previous result that removing stop words harm the performance of most of the models, because there are idiosyncrasies between different classes lying in the frequency of stop words. The models clearly take advantage of the idiosyncrasies in data, which may harm its general applicability [29].

After consulting with Guest lecturer, Dr. Chieu Hai Leong from DSO Lab about common bias in NLP, especially in fact checking field, the team followed his suggestion about generating a syntax synthetic pairing strategy to tackle this issue. The main idea was similar to Schuster *et al* in the paper [30], that for an original claim-evidence pair, manually generate a synthetic pair that holds the same relation (i.e. SUPPORTS or REFUTES) while expressing a fact that contradicts the original sentences.

Combining the ORIGINAL and GENERATED pairs, this new test set completely eliminates the ability of models to rely on cues from claims.

The below figure 15 is the example Dr. Chieu used to illustrate for us.

Source	Claim	Evidence	Label
ORIGINAL	Tim Roth is an English actor.	Timothy Simon Roth (born 14 May 1961) is an English actor and director.	SUPPORTS
GENERATED	Tim Roth is an American actor.	Timothy Simon Roth (born 14 May 1961) is an American actor and director.	SUPPORTS
ORIGINAL	Magic Johnson did not play for the Lakers.	He played point guard for the Lakers for 13 seasons.	REFUTES
GENERATED	Magic Johnson played for the Lakers.	He played for the Giants and no other team.	REFUTES

Figure 15: Synthesis pair examples by Dr. Chieu

However, due to limited time and manpower, only 5 synthesis pairs were generated. As a result of that, unbiased training was not included in this paper.

6 Conclusion

This paper introduced the popular models for text classification task and most importantly, the comparison between traditional machine learning methods and deep learning models. Despite the popularity and performance guarantees from deep learning models in other classification task (e.g. image classification), the results from this paper have shown that traditional machine learning task, such as Linear SVC model is able to outperform LSTM for this problem. The differences in performance was mainly explained by the inherent differences in the properties of text data.

A limitation of of the project [1] mainly lies in the data. Given the imbalanced data set, the team has to resort to different sampling methods to overcome this issue. Even then, as discussed in earlier sections, it was not very successful. On top of that, only the content of the news were given without the source, etc. This limited the ability to engineer other useful features. More avenue of information such as its source, year published, how is it propagated and existing knowledge bank can leveraged to build a better model. As such, for future work, more data can be mined to create a balanced data set with more information for each entry in the data set.

[1] Github Repository: https://github.com/iamthuya/cs4248-final-project-team16

Acknowledgements

Our team would like to acknowledge the guidance provided by our project mentor, Miao YiSong, guest lecturer Dr. Chieu Hai Leong, Professor Kan Min-Yen, and Professor Christian Von Der WETH. Also the teaching assistance who answered our questions on forum and in tutorials.

Statement of Independent Work

1A. Declaration of Original Work. By entering our Student IDs below, we certify that we completed our assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, we are allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify our answers as per the class policy.

Signed, [A0225551L, A0131114A, A0148048B, A0194499E, A0182884N]

Instructions to run the notebooks

All the related codes are hosted on a github repository. It can be accessed here [2].The notebooks can be run either on google colaboratory or on a local system by following the instructions below -

- Create a Python virtual environment using the provided 'requirements.txt' file
- Place the data files inside the 'notebooks' folder
- Change the input file path appropriately
- Download GloVe: Global Vectors for Word Representation
- Place the GloVe files inside the 'notebooks' folder

[2]Github Repository: https://github.com/iamthuya/cs4248-final-project-team16

References

[1] *A comprehensive guide to understand and implement text classification in Python*. July 2019. URL: https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/.

[2] Amy Watson. *Consumers who have seen fake news about the coronavirus U.S. by politics 2020*. June 2020. URL: https://www.statista.com/statistics/1105067/coronavirus-fake-news-by-politics-us/.

[3] Ilaria Montagni et al. "Acceptance of a Covid-19 vaccine is associated with ability to detect fake news and health literacy". In: *Journal of Public Health* 43.4 (2021), pp. 695–702.

[4] Bupt-Gamma. *Release dataset · Bupt-gamma/comparenet*$_{fakenewsdetection}$. URL: https://github.com/BUPT-GAMMA/CompareNet_FakeNewsDetection/releases/tag/dataset.

[5] Xinyi Zhou and Reza Rafarani. "A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities." In: (2020).

[6] Marcelo Mendoza Carlos Castillo and Barbara Poblete. "Information credibility on Twitter. In Proceedings of the 20th International Conference on World Wide Web". In: (2011), pp. 675–684.

[7] Atishay Jain Xinyi Zhou and Reza Zafarani. "Fake News Early Detection: An Interdisciplinary Study." In: (2019).

[8] K. Liu Z. Liu X. Lv and S. Shi. "Study on SVM Compared with the other Text Classification Methods". In: (2010), pp. 219–222.

[9] Nemanja Spasojevic Adithya Rao. "Actionable and Political Text Classification using Word Embeddings and LSTM". In: (2016).

[10] Alper Kursat Uysal and Serkan Gunal. "The impact of preprocessing on text classification". In: *Information processing & management* 50.1 (2014), pp. 104–112.

[11] Garrett Nicolai and Grzegorz Kondrak. "Leveraging Inflection Tables for Stemming and Lemmatization." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1138–1147.

[12] Aditya Wiha Pradana and Mardhiya Hayaty. "The effect of stemming and removal of stopwords on the accuracy of sentiment analysis on indonesian-language texts". In: *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control* (2019), pp. 375–380.

[13] Joachims Thorsten. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. URL: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf.

[14] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[15] Mainak Sarkar and Arnaud De Bruyn. "LSTM response models for direct marketing analytics: replacing feature engineering with deep learning". In: *Journal of Interactive Marketing* 53 (2021), pp. 80–95.

[16] Yong Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270.

[17] Alex Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[18] Alex Graves. "Long short-term memory". In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[19] Winda Kurnia Sari, Dian Palupi Rini, and Reza Firsandaya Malik. "Text Classification Using Long Short-Term Memory with GloVe". In: *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)* 5.2 (2019), pp. 85–100.

[20] Md Islam, Fazla Elahi Md Jubayer, Syed Ikhtiar Ahmed, et al. "A comparative study on different types of approaches to Bengali document categorization". In: *arXiv preprint arXiv:1701.08694* (2017).

[21] Varsha Pathak et al. "Kbcnmujal@ hasoc-dravidian-codemix-fire2020: Using machine learning for detection of hate speech and offensive code-mixed social media text". In: *arXiv preprint arXiv:2102.09866* (2021).

[22] M. Warmuth J. Kivinen and P. Auer. "The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant." In: *Conference on Computational Learning Theory* (1995).

[23] Abdelwadood Moh'd A Mesleh. "Chi square feature extraction based svms arabic language text categorization system". In: *Journal of Computer Science* 3.6 (2007), pp. 430–435.

[24] Inna Vogel, Jeong-Eun Choi, and Meghana Meghana. "Similarity Detection Pipeline for Crawling a Topic Related Fake News Corpus". In: *arXiv preprint arXiv:2009.13367* (2020).

[25] Inggrid Yanuar Risca Pratiwi, Rosa Andrie Asmara, and Faisal Rahutomo. "Study of hoax news detection using naıve bayes classifier in Indonesian language". In: *2017 11th International Conference on Information & Communication Technology and System (ICTS)*. IEEE. 2017, pp. 73–78.

[26] Xichen Zhang and Ali A Ghorbani. "An overview of online fake news: Characterization, detection, and discussion". In: *Information Processing & Management* 57.2 (2020), p. 102025.

[27] Carol A Watson. "Information literacy in a fake/false news world: An overview of the characteristics of fake news and its historical development". In: *International Journal of Legal Information* 46.2 (2018), pp. 93–96.

[28] Anu Shrestha and Francesca Spezzano. "Textual characteristics of news title and body to detect fake news: a reproducibility study". In: *European Conference on Information Retrieval*. Springer. 2021, pp. 120–133.

[29] Irena Spasic, Goran Nenadic, et al. "Clinical text data in machine learning: systematic review". In: *JMIR medical informatics* 8.3 (2020), e17984.

[30] Tal Schuster et al. "Towards debiasing fact verification models". In: *arXiv preprint arXiv:1908.05267* (2019).