Final Project Report

BT4222: Mining Web Data for Business Insights

Prepared By: Group 13

Lee Sheng Hao Darren (A0201940U)

Loh Hong Tak Edmund (A0199943H)

Tan Yi Bing (A0204181U)

Teo Zhi Feng (A0203989N)

Yap Kai Herng (A0199729A)

Contents Page

1. Background and Overview

The COVID-19 Pandemic has proven to be a devastating crisis that has reached and affected every corner of the world. This pandemic has resulted in serious health, social, economic and geopolitical challenges in my countries around the world (KPMG, 2020). In the midst of the crisis, there lies an issue that has generated a significant amount of social tension that slowed down and hindered recovery in many areas - fake news (Nyilasy, 2021).

With the exponential rise of internet penetration globally, people now are having unprecedented access to online news sites, social media and forums. While the Internet has facilitated the transmission of important information regarding the pandemic, the difficulty of regulating and verifying information has allowed for the spreading of fake, untruthful and even malicious content (Lazer et al., 2018). In Singapore, it is reported that at least 6 in 10 people have received fake Covid-19 news, mostly from social media (Chew, 2020). Such fake information has caused panic and anxiety in people who are unable to discern this information (Depoux et al., 2020) and increased propensity to reject information from verified experts (Uscinski et al., 2020) . Furthermore, fake news has enabled people to resist and undermine the country's pandemic recovery process (Freeman et al., 2020).

Evidently, there is an alarming need to tackle the issue of fake news, especially during the pandemic where it is a matter of life and death. While there are existing systems to detect fake news in today's social media sites, fake news seems to spread significantly faster (Vosoughi et al., 2018). Interestingly, despite the increasing interest in this area, more research is warranted to develop a unified, effective solution to counter this problem of detecting fake news accurately and swiftly (Vosoughi et al., 2018). Furthermore, as the syntax and semantics of information can differ across cultures and context, these layers of nuances have added complexities to discerning between fake and real news (Stewart, 2021).

2. Project Objective and Methodology

Objective

This project aims to create a classification model that effectively predicts Covid-19 Fake News given a piece of text.

Research Question

Which text vectorisation method and machine learning model can best predict Covid-19 Fake News across classification performance and time?

Methodology

Our group will explore different text vectorisation methods and machine learning models to determine which vectorisation method-machine learning model pair has the best classification performance.

Vectorisation Methods	**Models**
Count Vectorisation	TextCNN
TF-IDF Vectorisation	LSTM (Unidirectional)
Word2Vec	LSTM (Bidirectional)
	LSTM + CNN
	BiLSTM + CNN
	XGBoost Classifier
	RandomForest Classifier
	Gaussian Naive Bayes Classifier
	Multinomial Naive Bayes Classifier
	PassiveAggressive Classifier

Table 1: Vectorisation methods and models used

This study will be conducted with the following methodology:

1. Data Preparation and Pre-Processing
2. Exploratory Data Analysis

3. Text Vectorisation
4. Model Building and Validation
5. Model Evaluation and Comparison
6. Insights Mining
7. Limitations and Further Extensions

Metrics for Evaluation - SPACE

There are two dimensions that we will be considering in our evaluation of each model: Classification Performance and Training Time.

In terms of classification performance, we will be considering loss, accuracy, precision, recall, F1 score and ROC-AUC, with particular emphasis on the latter two.

The reason why we would consider training time an important aspect is due to the nature of the problem at hand. Fake news classification over social media entails a large and constant stream of data, and fake news trends change all the time. In addition, predictions must be made promptly and in real time. This requires constant updates of models to account for drift, which means constantly retaining. Models that take a long time to train are thus unsuited for the fast-moving nature of the problem at hand.

Hence, we propose a Speed and Performance Adjusted Combined Evaluation, SPACE, as a way to score models. SPACE(n) is evaluated as:

$$tanh(F1^{n} \ / \ tanh(time))$$

where *n* is an indicated preference of classification performance over training speed.

We chose F1 as the main metric as our dataset is balanced and F1 provides a good blend of precision and recall. We used the hyperbolic tangent function to squash all our values for F1 and training time between 0 and 1. Essentially this metric tries to look at performance per time, but since the range of training time spans more magnitudes, we needed to increase the weight of F1. Our choice of *n* was bound by 2 criteria: First, *n* cannot be overly large - that would cause the relative ranking of the models using this metric to simply tend towards the F1-score ranking. Next, *n* should still be large enough, in order to reflect the increased importance a model's performance has, as compared to the training time and thus we chose SPACE(24) as our evaluation metric.

3. About the Dataset

For this project, we will be using the COVID19 Fake News Dataset NLP from Kaggle.[1] It contains a column comprising the news content and a column comprising the label (real or fake) for each news content.

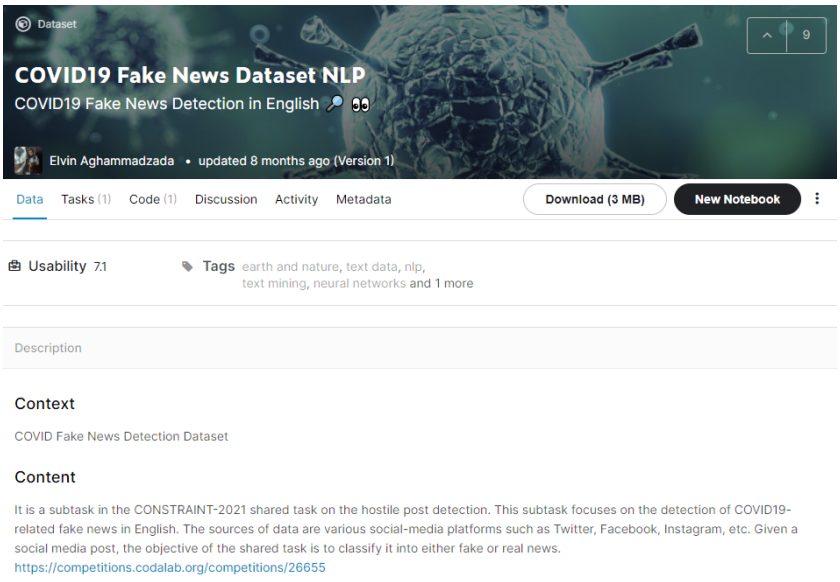

Figure 1: Kaggle dataset

[1] https://www.kaggle.com/elvinagammed/covid19-fake-news-dataset-nlp?select=Constraint_Train.csv

4. Data Preparation and Pre-Processing

Sanity Check

After loading the data, we first performed a sanity check on the data. No null inputs and duplicated rows were found.

Data Balance

Next, we inspected the distribution of class labels to check for data balance. The data is appropriately balanced across “real” and “fake” classes (Figure 2).

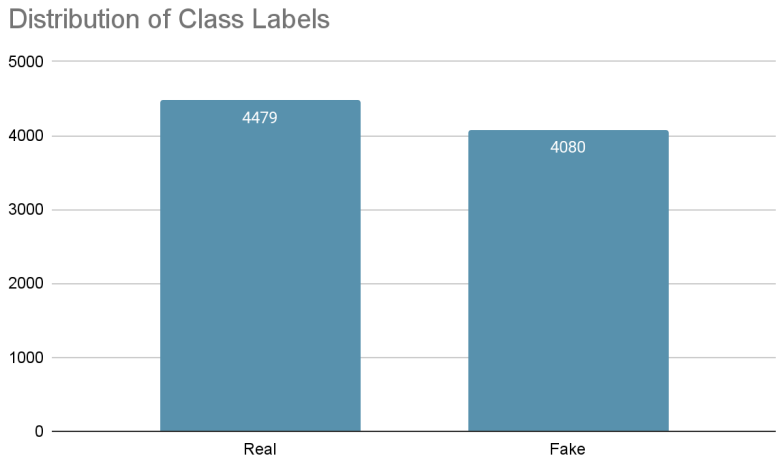

Figure 2: Distribution of class labels

Data Pre-Processing

To better understand our raw text data and identify potential outliers, we examine the word counts for each document. The following figure summarises the distribution of word counts.

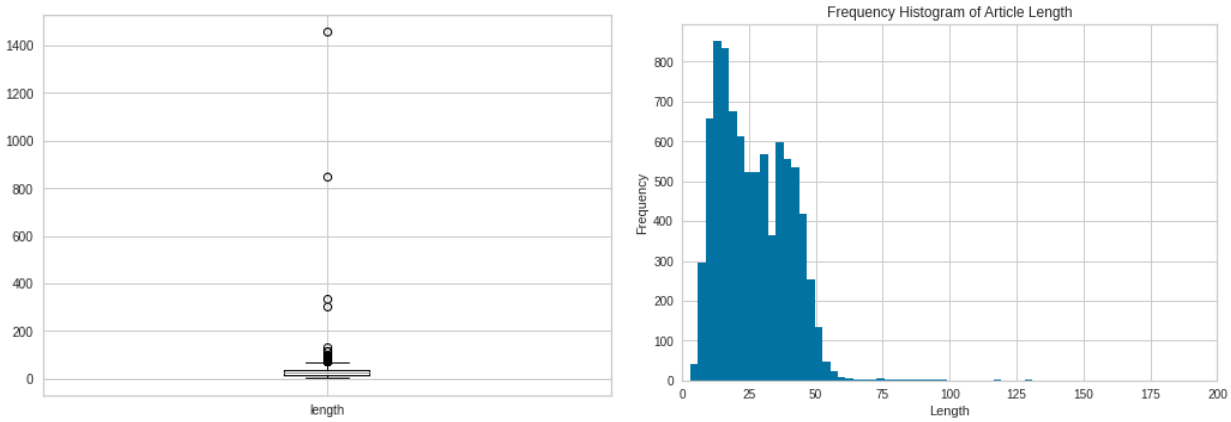

Figure 3: Box plot (left) and histogram (right) for distribution of word counts

From Figure 3, we notice several outlier documents with higher word counts from the box plot. We remove the outlier in the frequency histogram. We also notice that most of the documents contain about 20 words. To decide if we should retain or remove the outliers, we will first read them to identify any potential patterns (see Table 2).

Article	Label	Length
Amoxi" Capsule made in US is the only medicine...	fake	1456
Actors" applauded Macron when he visited the s...	fake	847
Man visited Albany N.Y. days before dying from...	fake	336
I'm so happy that we’re able to do something v...	real	304

Table 2: First four sample documents

The outlier articles are long and will potentially introduce noise in our model, hence we will remove them in our dataset to be used for model building and evaluation.

In this stage, we proceed to clean the raw text data to increase interpretability of the text data by our machine learning models. We performed the following transformations to the text:

1. Transform text to lowercase
2. Remove trailing white spaces
3. Remove stopwords
4. Stem and lemmatise words to their root forms

Previewing a sample of 15 documents returns the following:

```
 'broad though lag data thing trend right direct true even number hope
date like current hospit statist state report peterj walker'
 'bon jovi song contest corona virus meet aggress onlin humor writer
covid19'
 'burger king offer free design coronavirus mask purchas coronavirus
burgerk sofiavergara'
 'drink ayurved decoct 6 thousand corona infect patient cure'
 'sinc august 11 contact trace team identifi 4014 close contact case 4006
contact self isol complet self isol process contact rest'
 'methodolog note say continu revis pin way code hospit data state report
two differ number current hospit cumul hospit unlik number'
 '4 4 aid develop state intervent plan follow icmr survey state also
conduct zone citi specif survey indiafightscovid19'
 'dublin canal run clear amid covid lockdown reveal beauti shop trolley
handgun'
 'cdc director robert redfield say healthi peopl wear mask'
 'everi worker front line fight covid 19 deserv protect accommod keep
famili safe glad see state step support folk employ must'
 'state report 1192 death 7 day averag went 1000 first time week'
 'devbhatt moca goi 1st juli 2020 moca goi safe oper 785 domest flight
handl 71471 passeng'
 'latest updat ministri health manat hauora today new case covid 19 report
new zealand mean new zealand combin total confirm probabl case remain 1498
1148 confirm case covid 19'
 'thus enhanc time test keep posit rate low also fatal rate low'
 'updat covid19 impact hiv tb amp malaria relat death 5 year could increas
10 36 high burden set maintain critic prevent amp healthcar servic
signific reduc overal impact covid 19 epidem report'
```

Train-Test Split

Following this, we split the data into training and testing sets in the ratio 0.8 for training and 0.2 for testing (Figure 4).

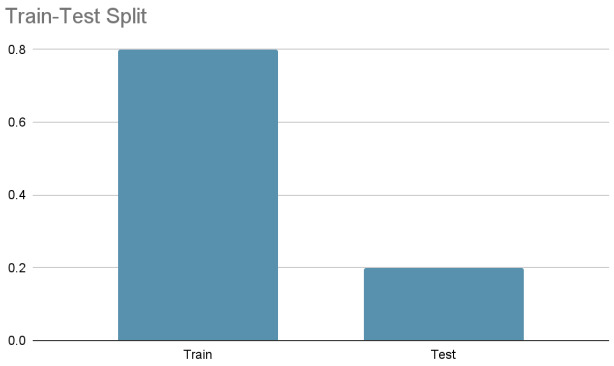

Figure 4: Train Test Split

5. Exploratory Data Analysis

Length Analysis

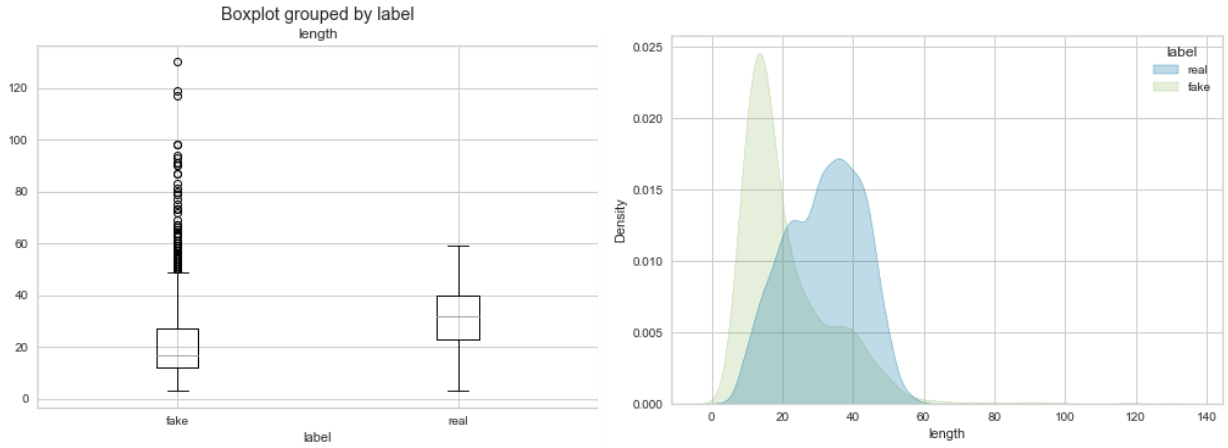

Figure 5: Distribution of article length by label

	Real News Length	**Fake News Length**
Mean	31.684974324626033	21.048074564630856
Skew	-0.1808558590317423	1.759677011339499
Kurtosis	-0.8023275139487924	5.620895453834617
Shapiro-Wilk P-Value	1.3716072864927914e-23	0.0
Bartlett's Test of Homogeneity of Variances P-Value	1.0500508393955644e-27	
Mann-Whiteney U Test P-Value	0.0	

Table 3: Statistical values of lengths

We first conducted a length analysis to explore if the length of the article differs between real and fake news. We plotted a box plot and density curves to visualise the distribution of article lengths between real and fake news. From Figure 5, we notice that the length of real news tends to be greater than that of fake news. We notice that the length of fake news is highly skewed right, with a high skewness value of 1.7597. The length of fake news also has a very narrow and sharp peak, with a kurtosis value of 5.6209. This indicates that fake news tends to be of shorter lengths with lower variances as compared to real news.

To strengthen the hypothesis that the length of real news is longer than that of fake news, we conduct a one-sided hypothesis test between the lengths of real and fake news from our dataset. Upon conducting Sharpiro-Wilk tests, we obtain very small P-values for both the real and fake news, indicating high departure from the normal distribution. Bartlett's test of homogeneity of variances also produced a very small P-value, indicating high heterogeneity of variances between the mean lengths of fake and real news. We hence perform a Mann-Whiteney U test to determine if the mean lengths of real and fake news are different. The test produced a P-value of 0.0, indicating that it is impossible that both real and fake news have equal mean lengths.

This indicates with strong statistical significance that fake news is of shorter length than real news. In practical applications, shorter news articles within the scope of Covid-19 will more likely be fake news.

WordCloud

To understand the frequency of different terms and words in the dataset, we decided to visualize the word frequencies in a WordCloud. However, to account for variations of the same word, we feed the word cloud lemmatized data, which is why word stems instead of full words are observed. In addition, symbols words referring directly to Covid-19, and the word ‘people’ added into the stop word list as they do not contribute meaningfully to the understanding of the dataset despite appearing frequently.

Figure 6: WordCloud over ALL samples

Looking at the WordCloud over the whole dataset, we can see that ‘test’, ‘new’ and ‘say’ are the most common terms, with topics like vaccines and hospitals being popular.

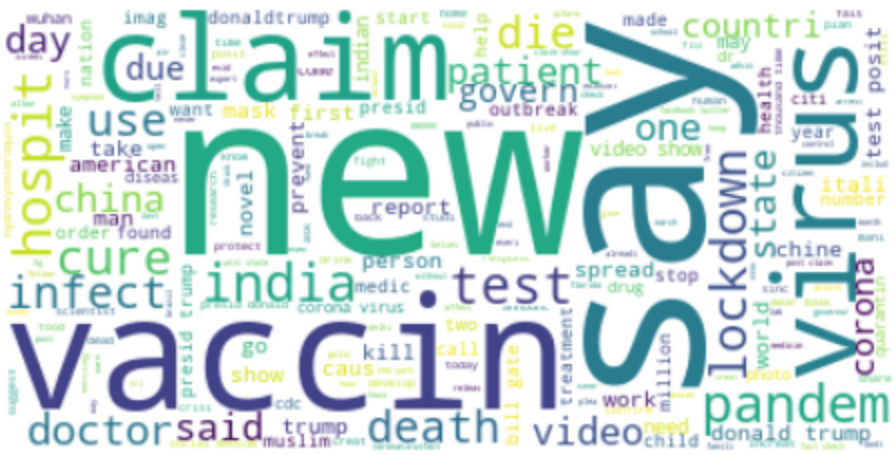

Figure 7: WordCloud over FAKE samples

However, more interesting is the WordCloud over fake news samples. Vaccine-related terms come up a lot more often, and more anecdotal terms like ‘say’/’said’ and ‘claim’ increase in frequency.

Figure 8: WordCloud over REAL samples

In contrast, we see that the WordCloud over real samples are a lot more focused on reporting and facts. With terms like ‘data’/’number’/’test’ being more prominent in real samples compared to the fake samples, and anecdotal terms like ‘say’ being much less prominent, overall the samples in the ‘real’ WordCloud tend to have numbers/data to backup their statements while ‘fake’ samples tend to use anecdotes.

Also, an interesting distinction between real and fake samples is how prominent the fake samples feature ‘cures’. The Covid-19 pandemic has seen many dubious healthcare professionals, religious leaders, social media influencers and even politicians hawking ‘cures’ for the virus (BBC News, 2020). More often than

not, these bogus cures are attempts at unscrupulous profiteering and do not have any efficacy in Covid-19 treatment. In contrast, the medical community's discourse has been less likely to tout 'cures'. In fact, established methods of treating Covid-19 are not so much singular cures but rather courses of treatment that are unlikely to be characterised as 'cures'. In addition, the Covid-19 pandemic has spawned a new wave of anti-vaccination misinformation. This phenomena has been prominently observed across the globe where a deluge of fake news centered around vaccine effectiveness or mandates has flooded social media (Surowiecki, 2021). It is interesting to note that the fake samples contribute to most of the mentions of vaccines in the dataset.

Given how the real and fake WordClouds are noticeably different from one another, it supports a belief that real and fake tweets are meaningfully different from each other and are separable, which means that training classifiers over the dataset could yield useful and accurate models.

K-means

Given the differences we observed between real and fake WordClouds, an additional area we wanted to explore was whether this would be reflected as clusters of topics. As such, we ran a k-means clustering on the word embeddings we obtained, which will be explained in further detail in the next section on Vectorization methods.

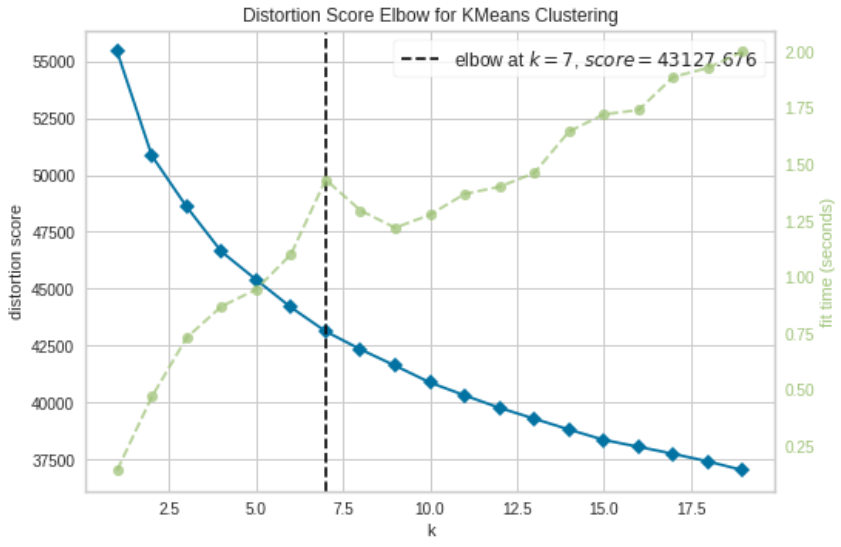

Figure 9: K-means clustering distortion plot (Word2Vec embedding)

In order to find the optimal number of clusters, we attempt to use the elbow method and plot out the distortion score for varying numbers of clusters. However, we see from Figure 9 that while the algorithm

identifies k = 7 as the optimum number of clusters, there is no obvious elbow here. The downward-sloping curve is fairly smooth, with no obvious kinks.

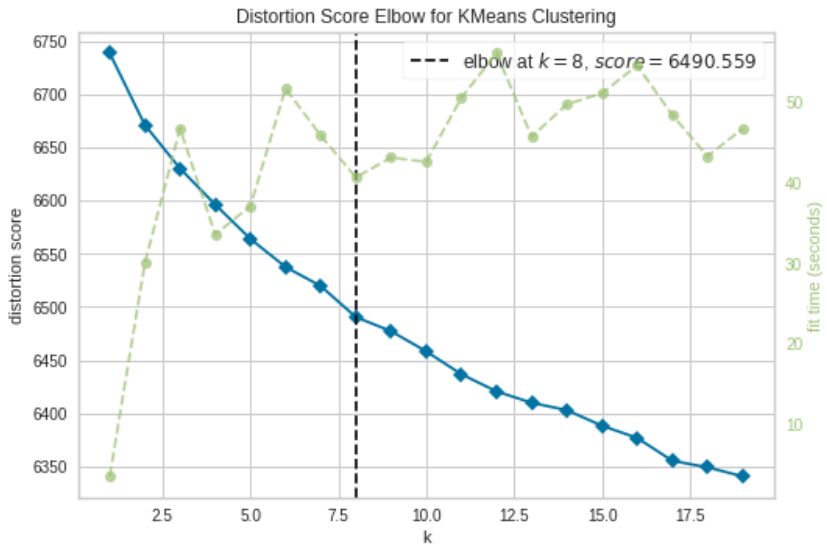

Figure 10: K-means clustering distortion plot (TF-IDF embedding)

Similar to the previous Word2Vec plot, we see that k-means clustering for the TF-IDF embedding does not result in an obvious elbow visually, with a fairly smooth downward-sloping curve.

The lack of an obvious elbow for both plots is explainable - while the dataset was only classified into real or fake news, in actuality the news articles cover a wide range of topics in addition to whatever they mention about Covid-19. With the large variety of words used for the different topics covered, this leads to a large variation in the semantic meanings for each document. Thus with each successive 'split' when the number of clusters increases by one, the improvement in distortion score is on average fairly similar. As such, we chose not to proceed further with the actual clustering, since we felt that we were not going to be able to obtain clusters that would be particularly meaningful nor helpful to our analysis.

6. Vectorisation Methods

Count Vectorisation

The first vectorisation method we explore is Count Vectorisation. Count Vectorisation converts a collection of text documents to a vector containing the overall frequencies of each word with respect to all documents. This basic form of vectorisation serves as the baseline vectorisation method for our models.

TF-IDF Vectorisation

The second vectorisation method we explore is TF-IDF Vectorisation. TF-IDF Vectorisation represents each word as a product of the word's term frequency within the document and the inverse document frequency of the word across all the documents. This vectorisation technique not only considers the frequency of a word, but also evaluates the relevance of a word to a document in a collection of documents.

Word2Vec Model

The third vectorisation method we explore is the Word2Vec Model. Word2Vec models embed words in lower-dimensional vector spaces by training on a shallow neural network. This results in a set of word-vectors where semantic relationships are mapped by spatial proximity within the vector space. Word-vectors closer together in the vector space have similar meanings based on context, while word-vectors distant to each other have differing meanings (Mujtaba, 2020). We implement the Word2Vec model using the Gensim package.

After training the Word2Vec model, we construct an Embedding Matrix containing the weights of individual words that will be trained by our models later.

7. Model Building and Validation

Overview

In this section, we will detail the different models that we have built, trained and evaluated. The models were trained using an Intel i7-8750 CPU @ 2.20GHz with 16GB of RAM and a NVIDIA GeForce GTX 1050 Ti with Max-Q Design graphics card (CUDA version 11.4).

Deep Learning: TextCNN

TextCNN is a convolutional neural network model for text data. It is a useful deep learning algorithm for sentence classification tasks such as sentiment analysis and question classification (Gong & Ji, 2018). We decided to utilise a simple model with one CNN layer due to its excellent model performance despite its simplicity achieved by various researchers (Yoon, 2014).

Hence, we built the TextCNN model with the following architecture:

Embedding	Conv1D(128)	GlobalMaxPooling	Dense(10)	Dense(1) output

Table 4: Model architecture for TextCNN (left to right)

As the TF-IDF and Count Vector are sparse matrices, an additional step is required to build a custom data generator to fit the respective vectors with our TextCNN model.

For the TextCNN model, the results are summarised in the table below. TextCNN with word2vec outperformed the other vectorisation methods across all matrices and is faster by almost eight times as compared to other vectorisation methods.

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**	**F1**	**Time (sec)**	**SPACE (24)**
CountVectorizer	0.5933	0.7068	0.7238	0.6873	0.7388	0.7051	208.7419	0.000228
TF-IDF Vectorizer	0.6576	0.6139	0.5987	0.7365	0.6559	0.6605	195.9644	0.000048
Word2Vec	**0.2431**	**0.9217**	**0.9138**	**0.9347**	**0.9715**	**0.9241**	**26.4311**	**0.149364**

Table 5: Results for TextCNN

Deep Learning: Long Short-Term Memory (LSTM)

LSTM is a type of artificial recurrent neural network architecture which recognises patterns across time and is ideal for modelling sequential data. It captures long-term dependencies between word sequences by maintaining an internal state that encodes information from previous timesteps. about the timesteps it has seen so far (TensorFlow, n.d.). LSTM is one of the most effective types of Recurrent Neural Network (RNN) architecture because it mitigates the vanishing and exploding gradient problem in traditional RNNs where the repeated multiplication across a deep RNN network can cause values to vanish or explode (Goldberg, 2017). However, the order of words can lead to bias.

We attempted two types of LSTM: unidirectional and bidirectional. For the unidirectional LSTM model, we built it with the following architecture:

Embedding	LSTM(64)	GlobalMaxPooling	Dense(10)	Dense(1) output

Table 6: Model architecture for unidirectional LSTM (left to right)

Similar to TextCNN, our LSTM model performed the best with word2vec across all metrics except recall. It also outperformed the TextCNN model across most metrics but at the expense of longer training time.

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**	**F1**	**Time (sec)**	**SPACE (24)**
CountVectorizer	0.5997	0.7050	0.7244	0.6804	0.7316	0.7017	956.0869	0.000203
TF-IDF Vectorizer	0.6975	0.5099	0.5099	**1.0000**	0.6174	0.6754	1045.5528	0.000081
Word2Vec	**0.2031**	**0.9305**	**0.9415**	0.9210	**0.9768**	**0.9311**	**37.3143**	**0.178312**

Table 7: Results for unidirectional LSTM

For the bidirectional LSTM model, its upd The first model is trained to recognise sequential patterns while the second model learns from the reverse of that sequence. Both models will then be combined. We built this model with the following architecture:

Embedding	Bidirectional (LSTM(64))	GlobalMaxPooling	Dense(10)	Dense(1) output

Table 8: Model architecture for bidirectional LSTM (left to right)

Similarly, our bidirectional LSTM model performed the best with word2vec across all metrics. Compared to TextCNN and unidirectional LSTM models, this model performed the best across most matrices and

achieved the lowest loss of 0.1916. It also achieved the highest AUC compared to all other models. However, this comes at the expense of a slower training time.

	Loss	Accuracy	Precision	Recall	AUC	F1	Time (sec)	SPACE (24)
CountVectorizer	0.5936	0.7056	0.7183	0.6953	0.7588	0.7066	2058.7930	0.000240
TF-IDF Vectorizer	0.6531	0.6209	0.6055	0.7365	0.6634	0.6646	1997.6188	0.000055
Word2Vec	**0.1916**	**0.9328**	**0.9448**	**0.9221**	**0.9808**	**0.9333**	**67.0973**	**0.188647**

Table 9: Results for bidirectional LSTM

Deep Learning: CNN and LSTM Hybrid Models

Hybrid models are used extensively in the field of fake news detection, where researchers derived promising results from different CNN and LSTM combinations (Nasir et al., 2021). By combining CNN's ability to extract local features and LSTM's ability to learn long-term dependencies, CNN and LSTM hybrid models can perform better because they utilise both local and sequential information of the dataset (Zhou et al., 2015). In our paper, we built two different hybrid models: LSTM-CNN and Bidirectional LSTM-CNN.

For our LSTM-CNN model, we connected a LSTM layer to the embedding layer, followed by a one-dimensional CNN layer. Max pooling layers are added in between to reduce dimensionality and reduce the likelihood of overfitting. Hence, we built this model with the following architecture:

Embedding	Conv1D (128)	MaxPooling1D	LSTM(64)	GlobalMaxPooling	Dense(10)	Dense(1) output

Table 10: Model architecture for LSTM-CNN (left to right)

Similar to the CNN and LSTM models, the LSTM-CNN model performed the best with word2vec. Its results are comparable to that of LSTM, achieving similar accuracy and F1-score and AUC, and are equally time efficient.

	Loss	Accuracy	Precision	Recall	AUC	F1	Time (sec)	SPACE (24)
CountVectorizer	0.5791	0.7120	0.7661	0.6266	0.7690	0.6894	1120.5744	0.000133
TF-IDF	0.6291	0.6525	0.6466	0.7022	0.7024	0.6733	1122.7999	0.000075
Word2Vec	**0.2677**	**0.9311**	**0.9227**	**0.9439**	**0.9724**	**0.9332**	**44.0183**	**0.187933**

Table 11: Results for LSTM-CNN

For our Bidirectional LSTM-CNN model, the model architecture is similar to that of the LSTM-CNN model, with the exception that a bidirectional LSTM layer is used instead of a LSTM layer.

Embedding	Bidirectional (LSTM(64))	MaxPooling1D	Conv1D(128)	GlobalMaxPooling	Dense(10)	Dense(1) output

Table 12: Model architecture for bidirectional LSTM-CNN (left to right)

Similar to all other deep learning models, our Bidirectional LSTM-CNN model with word2vec performed the best across all matrices. In fact, this model achieved the highest accuracy (0.9357) and F1-score (0.9374) amongst all other models.

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**	**F1**	**Time (sec)**	**SPACE (24)**
CountVectorizer	0.5784	0.7103	0.7860	0.5934	0.7705	0.6762	2272.5826	0.000084
TF-IDF	0.6296	0.6548	0.6466	0.7125	0.7015	0.6779	2225.3081	0.000089
Word2Vec	**0.2218**	**0.9357**	**0.9320**	**0.9427**	**0.9769**	**0.9374**	**74.7573**	**0.208596**

Table 13: Results for bidirectional LSTM-CNN

Ensemble: XGBoost Classifier

XGBoost is a state-of-the-art gradient boosting library that implements a gradient boosting decision tree algorithm, while being scalable and efficient (Chen & Guestrin, 2016). Gradient boosting is an ensembling approach where new models are added to predict the residuals of prior models, and are then all combined together for a final prediction. In XGBoost, the base model used is a decision tree. In general, XGBoost tends to perform better with structured/tabular data, as opposed to unstructured data.

For our XGBoost model, the results are summarised as follows:

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**	**F1**	**Time (sec)**	**SPACE (24)**
CountVectorizer	**0.2153**	**0.9106**	**0.9176**	0.9061	**0.9729**	0.9118	**1.4559**	**0.121036**
TF-IDF	0.2199	**0.9106**	0.9128	**0.9118**	0.9709	**0.9123**	1.9851	0.114297
Word2Vec	0.3904	0.8364	0.8233	0.8648	0.9106	0.8436	2.2011	0.017281

Table 14: Results for XGBoost

Here, we see that XGBoost performs better with the Count and TF-IDF Vectorizer, as compared to Word2Vec.

Ensemble: Random Forest Classifier

Random decision forest is an ensemble learning method where a multitude of decision trees are trained on the training dataset, and is an example of bagging. For classification tasks, the output of the random forest is the class that is selected by the majority of the trees in the ensemble. This helps to improve prediction accuracy, and combat overfitting.

	Loss	Accuracy	Precision	Recall	AUC	F1	Time (sec)	SPACE (24)
CountVectorizer	**0.2406**	0.9235	**0.9314**	0.9175	0.9762	0.9244	3.0639	0.151109
TF-IDF	0.2431	**0.9246**	0.9256	**0.9267**	**0.9771**	**0.9262**	2.7267	**0.158668**
Word2Vec	0.4418	0.7891	0.7777	0.8213	0.8808	0.7989	**1.1470**	0.005592

Table 15: Results for random forest

Similar to XGBoost, the Random Forest classifier appears to perform much better with the Count and TF-IDF Vectorizer, as opposed to Word2Vec.

Probabilistic Models: Gaussian Naive Bayes Classifier

Gaussian Naive Bayes is a probabilistic model that assumes the data from each label is drawn from a simple Gaussian distribution. Gaussian Naive Bayes is the easiest to work with as only estimation of the mean and the standard deviation from data is required.

	Loss	Accuracy	Precision	Recall	AUC	F1	Time (sec)	SPACE (24)
CountVectorizer	5.8709	0.8300	0.7704	0.9496	0.8273	0.8507	1.1801	0.024928
TF-IDF	**5.4070**	**0.8435**	**0.7969**	0.9301	**0.8424**	**0.8584**	1.1759	**0.030957**
Word2Vec	17.0936	0.5047	0.5073	**0.9897**	0.4943	0.6708	**0.0259**	0.002659

Table 16: Results for Gaussian Naive Bayes

We observed that our Gaussian Naive Bayes Classifier appears to perform better with TF-IDF amongst the three vectorization methods. The model with word2vec has the fastest train time but performed poorly in terms of classification, hence achieving an extremely low SPACE score of 0.002659.

However, it is noted that this model works best when working with continuous data. Thus, it is not surprising that the results below are poorer than the models since our dataset is not continuous. This model used to show the baseline performance of a naive bayes classifier.

Probabilistic Models: Multinomial Naive Bayes Classifier

Multinomial Naive Bayes Classifier is a probabilistic learning method and it is mostly used in Natural Language Processing (NLP). The algorithm uses the Bayes Theorem and calculates the probability of each tag of a text for a given sample and outputs the highest probability. The Naive assumption assumes that every word in a sentence is independent of the other ones, thus it does not need to look at the full sentences to predict. As such, this model is robust against irrelevant features while emphasizing important features. This model is used as a baseline for text classifications and will be compared with other better performing models.

	Loss	Accuracy	Precision	Recall	AUC	F1	Time (sec)	SPACE (24)
CountVectorizer	0.2969	**0.9188**	**0.9180**	0.9233	0.9766	**0.9206**	0.1696	0.673736
TF-IDF	**0.2393**	0.9141	0.8862	**0.9542**	**0.9788**	0.9189	0.1616	0.675313
Word2Vec	11.6501	0.6606	0.7433	0.5109	0.7049	0.6056	**0.0110**	0.000539

Table 17: Results for Multinomial Naive Bayes

Here, we observe that our Multinomial Naive Bayes Classifier performed well with TF-IDF and CountVectoriser, achieving SPACE scores of around 0.67. Despite having the fastest training time as compared to all other models, the model with word2vec performed poorly, with a SPACE score of 0.000539.

Online Learning Algorithm: Passive Aggressive Classifier

Passive Aggressive Classifier (PAC) was designed as an online learning algorithm. It excels at reading in large streams of sequential data and does one-by-one updates to the model for each observation, discarding observations once it has trained on it. This makes PACs good when there is a large amount of data to be considered and is infeasible to train on the whole dataset at the same time. This makes it a good algorithm to learn from large online datasets like Twitter. However, our dataset is not large enough to make use of its memory and time saving techniques, but could be a reasonable model to be deployed in a live environment, so we decided to test its performance on our dataset as well.

The main learning algorithm behind PACs has two components. Given an observation, it will make a prediction. If the prediction is correct, it will not update (passive). If the prediction is wrong, it will update the decision boundary to the extent such that the current observation is exactly on the correct side of the decision boundary (aggressive).

Two different methods, TF-IDF and Word2Vec, were used to transform the textual data into valid inputs for the PAC. Running the PAC with cross validation enabled, the following results were obtained:

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**	**F1**	**Time (sec)**	**SPACE (24)**
CountVectorizer	2.7841	0.9194	**0.9268**	0.9141	~~0.9195~~	0.9204	0.0271	0.999918
TF-IDF	**2.4210**	**0.9299**	0.9226	**0.9416**	~~0.9297~~	**0.9320**	**0.0133**	**1.000000**
Word2Vec	16.1398	0.5327	0.5356	0.6289	~~0.5308~~	0.5785	0.0953	0.000021

Table 18: Results for PAC

Minor caveat with this model is that it does not produce prediction probabilities, only hard labels of 0 or 1. Hence, the AUC score is not an appropriate way to evaluate this model, hence the discussion on this model will focus more on the F1 Score.

It is clear that the Word2Vec version of the model is significantly worse than the TF-IDF or CountVectorizer versions.

8. Overall Evaluation of Models

Appendix 1 lists the full results from our experiments. Using our SPACE(24) metric, our PAC model with TF-IDF gave us the best blended performance and has an exceptional blend of performance and speed, with a SPACE(24) score of 1.00000. Hence, it would be our pick to achieve our project objective of predicting Covid-19 Fake News given our selected dataset. Notably, it has been designed as an online learning algorithm that accounts for the memory and speed requirements that learning from an large, unending, online stream of data entails.

Analysing model performance using the other metrics, the Bidirectional LSTM-CNN with Word2Vec is our best performer in terms of classification performance, achieving an F1 score of 0.9374 and has the potential to be increased with further hyperparameter tuning. Our fastest model is Multinomial Naive Bayes with Word2Vec, training in 0.0110s. However, it must be noted that this model performed terribly in classifying fake news, achieving 0.6056 for F1 score and 0.000539 for our SPACE metric.

9. Insights

Performance-Time Tradeoff

As elaborated in Section 1, detecting fake news is particularly challenging given that it is extremely pervasive and it spreads rapidly. Consequently, the effectiveness of any fake news detection model relies heavily on two attributes: performance and time efficiency. This is exactly why we developed our SPACE metric because it is essential to evaluate models more robustly based on their classification performance as well as time efficiency.

In terms of performance, we observed models with LSTM layers performed the best. Our Bidirectional LSTM-CNN with Word2Vec and Bidirectional LSTM models with Word2Vec are the top two best performers in terms of accuracy and F1 score. This can be attributed to LSTM's ability to identify sequential patterns in making predictions very effectively and accurately.

Deep-diving into our results, we observe something interesting about time efficiency. For models which performed well in terms of F1 score (i.e. above 0.9), we observe that achieving a higher F1 score comes at the cost of longer training time (see Figure 11).

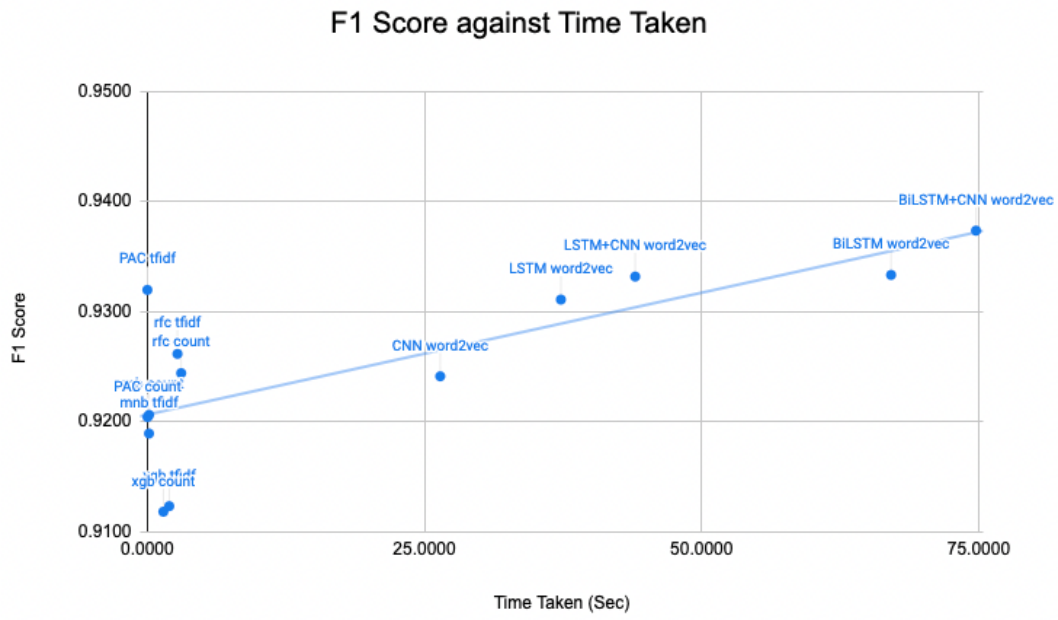

Figure 11: F1-score against time taken for model training for F1-score above 0.9

However, it is noted that this relationship is not linear. From Figure 12, we observed that models with long training time did not perform significantly better. For models with poor classification performance (i.e. below 0.9), this tradeoff does not seem to hold.

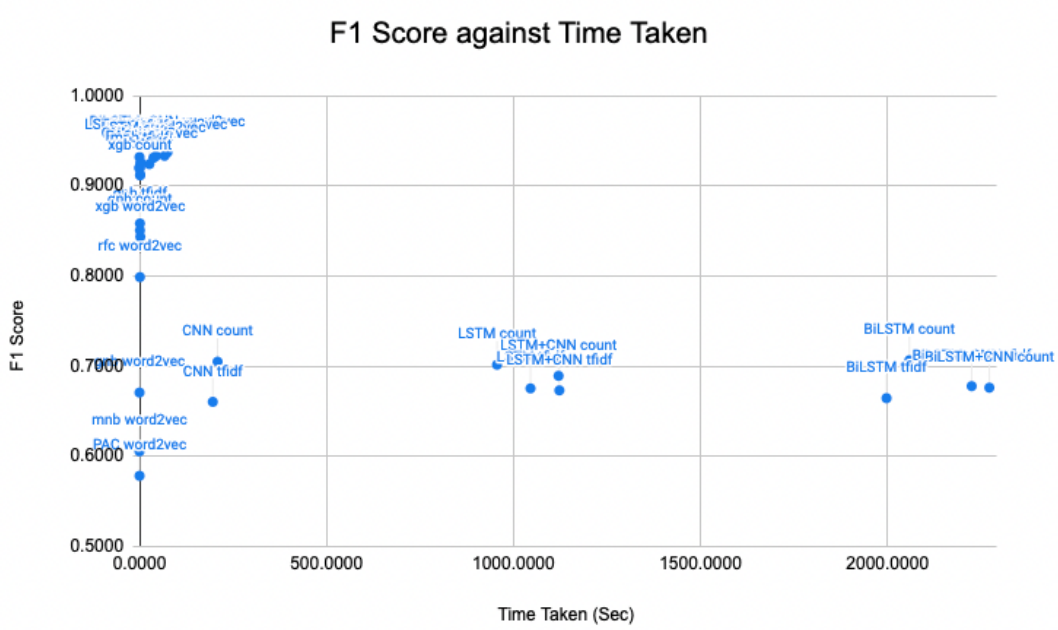

Figure 12: F1-score against time taken for model training for all models

Evidently, the performance-time tradeoff is a significant consideration when developing a model to detect fake news. If the desired objective is to achieve high classification performance regardless of time taken to train the model, then using Bidirectional LSTM-CNN using Word2Vec will be the better model. Else, if being able to predict fake news accurately yet swiftly is vital, then the PAC model on TF-IDF might be the better model using our SPACE(24) metric as it balances classification performance and time efficiency.

SPACE Metric Analysis

In order to properly account for the performance-time tradeoff, we employed the SPACE metric described in Section 2.

$$tanh(F1^{n} \ / \ tanh(time))$$

SPACE requires manual input: *n*, which determines the relative importance of F1 Score vs speed. However, an appropriate value for *n* is harder to determine as it is a subjective preference of F1 score over

speed and could change based on the practical real-world constraints when building the model. When *n* vs SPACE(n) is plotted for each of our models, we see that it is possible that the rankings change over *n*. This means that depending on subjective preferences, the relative rankings will shift, which is to be expected. As *n* tends to ∞, the SPACE(n) ranking will converge to the F1 Score ranking.

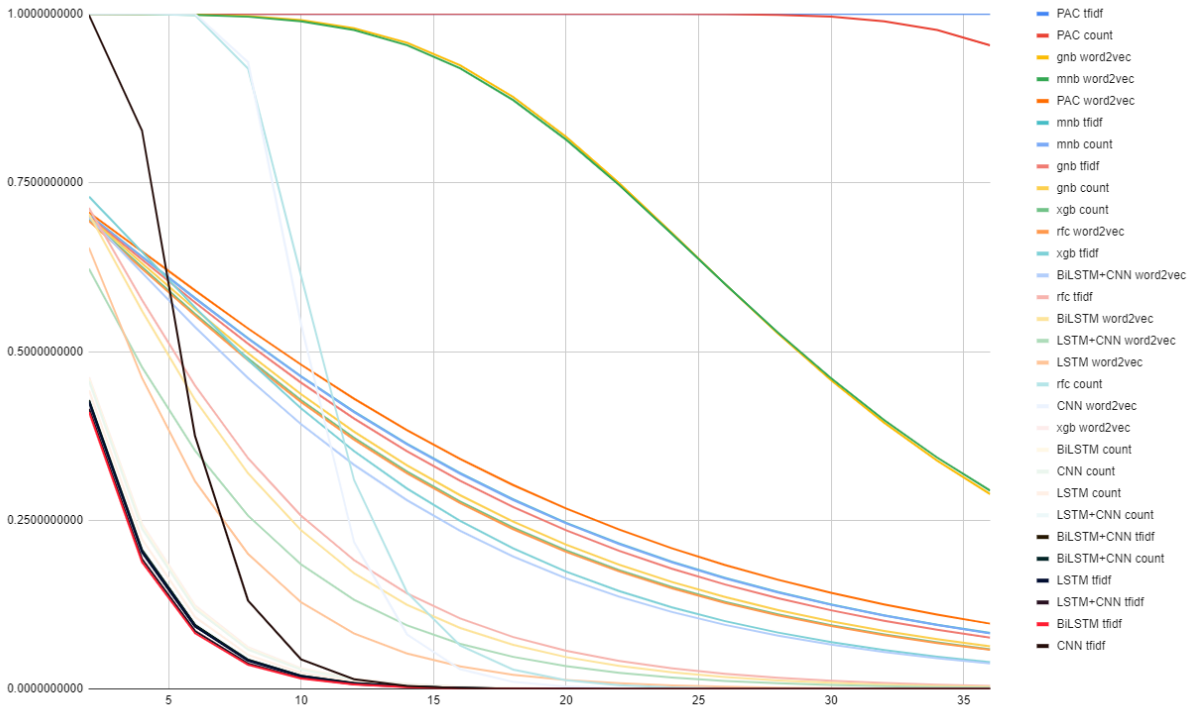

Figure 13: Graph of SPACE(n) performance across values of n.

Our group chose SPACE(24) for our experiments because we concur that it was large enough to give F1 score the needed importance in evaluation while still having speed play a decisive factor in the rankings. Having too low a value for *n* meant that models that were purely fast were ranked at the top, which is the case for SPACE(1) where Word2Vec Multinomial Naive Bayes ranked amongst the top despite having a very low F1 score.

From Figure 13, it is a fortuitous result that TF-IDF PAC ranked 1st across all values for what we considered to be reasonable values for *n*. This meant that while the rankings shifted underneath it, the TF-IDF PAC remained in an unassailed 1st place, which gives us further confidence that it is our model of choice for the problem we are considering. In practice, this means for a small to a reasonably large preference of classification performance over speed, TF-IDF PAC is the best model to deploy.

TF-IDF/CountVectorizer vs Word2Vec

In our results, we observe that traditional (i.e. non Deep Learning) methods like XGBoost, PAC, Naive Bayes etc, tend to fare worse when using Word2Vec vectors as inputs as compared to CountVectorizer or TF-IDF. A clear distinction in performance is drawn along the lines of semantic vs frequentist/tokenistic analysis. Word2Vec creates semantic representations of words while the latter two create variations of frequency metric for each word in each document in the corpus. This issue is split into two different issues: Why do the traditional methods do better on the frequentist inputs and why do the CNN/LSTM models do better on the semantic inputs?

Traditional methods tend to focus more on specific tokens or features in the data to draw decision boundaries. Let us consider XGBoost. The Word2Vec input would consist of an array of word vectors, while a TF-IDF input would consist of a vector of TF-IDF scores. The XGBoost model would then try to split along the values of specific array elements. For TF-IDF, that element would be a scalar value, but for Word2Vec, that would be a word vector. TF-IDF XGBoost is essentially considering the frequency of specific words in an observation and drawing a decision boundary based on how frequent that specific word is. E.g. ‘vaccin’ , a word that appears very frequently in fake samples (see WordCloud in section 5). If the TF-IDF XGBoost sees that ‘vaccin’ has a very high score in the observation, it will then want to classify the observation as fake. With this intuition we can see how TF-IDF is useful for XGBoost and other traditional methods that essentially draw decision boundaries using specific features (e.g. PAC). Conversely, for Word2Vec XGBoost, the element to be considered is a word vector. Splitting a word vector has no intrinsic linguistic meaning and presents the model with additional considerations since each vector has many dimensions. Not only does the model need to find the best element/feature to split on, it will also need to find where in the word vector an appropriate split lies. This increase in dimensionality most likely means that information is spread out over more dimensions so each split brings incrementally lesser gains. Hence, the way traditional methods try to draw decision boundaries struggles to handle the Word2Vec output, since they are unable to properly deal with the meanings encoded in the word vectors.

Conversely, CNN/LSTMs do not do as well with frequentist inputs. The idea is simple. Since the input from TF-IDF/CountVectorizer is a scalar score, it does not make much sense to convolute or identify patterns in words solely off their frequencies. While there is some information to be gleaned from that (evidenced by the fact TF-IDF/CountVect CNN/LSTMs had >0.6 F1 score), there is only so much that can be learnt by considering patterns or features in frequencies alone. It is much more informative for these models to learn based on word meanings and aggregate the words into useful higher level features

(CNN) or patterns (LSTMs). Which is why Word2Vec versions of CNN/LSTM are the top performing models in terms of F1 scores. Being able to consider the subtle semantic differences in words or patterns gives these models another source of information that is unavailable to traditional methods.

In conclusion, traditional machine learning methods are more focused on drawing decision boundaries using simpler features or tokens, however, deep learning methods like CNN/LSTM take a more holistic view of the data and consider the meanings of words.. Since the two types of learning have different paradigms of learning, they differ in the types of inputs they prefer.

Unidirectional VS Bidirectional LSTM Models

From our results, we see that Bidirectional LSTMs performed better than their Unidirectional counterparts. These results are not a surprise. For NLP tasks, the contextual meanings of words are often determined by both words in front and behind of the target word. Especially considering the syntactical idiosyncrasies of English, words before and after the target word do play a part in giving the target word a context. For example, in the previous sentence, ‘syntactical’ gives further meaning to ‘idiosyncrasies’. Unidirectional LSTMs will not be able to capture that information while Bidirectional LSTMs account for this reality.

10. Limitations

The primary limitation for our study was the time and computational resources available for model training. This limitation resulted in the present study considering only 1 set of hyperparameters for each model. A common concern regarding using neural networks is the selection of hyperparameters, such as filters and kernel sizes, which can in turn affect model performance. In our paper, we held constant the hyperparameters for each type of neural network - 128 filters and kernel size of 5 for CNN, and dimension of 64 for LSTM (both unidirectional and bidirectional). We also utilised the same activation function throughout, utilising relu function in intermediate layers and sigmoid function in the output layer. This limits the generalisability of our model representation to each respective model class.

In addition, our chosen blended metric, SPACE(24), was based on subjective choice and general intuition. It is entirely possible that given a different set of real world constraints that n could be different. However, we have shown that our chosen model remains the same over what we feel is a reasonable range for n.

Furthermore, we concede that this evaluation metric is a novel approach in blending classification performance and model training time. Further experiments and verifications in different datasets and context are needed to extend the generalisability of this metric in evaluating the absolute model performance. However, we are confident in its ranking ability, which we are most concerned about in this specific study.

11. Further Extensions

Moving forward, we propose extensions to our present study. Firstly, we would want to tune the hyperparameters of our chosen models as adjusting our model hyperparameters could potentially impact model performance. This hence allows us to represent each class of models with greater generalisability.

Also, research can be done into creating a better blended metric of classification performance and training speed such that it is dataset and hardware agnostic, such that models can be compared across different problem sets or tasks, since our current SPACE(24) metric requires manual adjustment.

Next, our paper can be extended to include more sources of information such as Facebook posts, news articles and blog posts. Furthermore, a greater diversity of data, in terms of geography and time period, can be included to ensure that the models that we have trained are more adaptable and robust in detecting fake news.

Furthermore, we could expand our model to train not just on text data, but also the metadata associated with the data. For instance, future models could train on the time and sequence of articles, the author profile and platform choice. Further studies could also consider and place more weight on article length, since it was established that the length of fake news is significantly shorter than that of real news. This will improve the robustness of our model and its generalisability in multiple contexts.

12. Conclusion

While COVID-19 has crippled economies around the world, the prevalence of fake news related to COVID-19 has threatened the stability of societies in today's digitally connected world. With the pervasiveness of the Internet, fake news has thus become a 'silent killer' in many countries around the world - the spread of fake news has caused anxiety amongst people and fostered mistrust in modern medicine. Hence, fake news detection has become more important than ever. Without a reliable, fast and effective method to identify fake news, it will be an arduous task to regulate information and protect people from blindly believing in misinformation.

Since social media has become a popular propagation channel for fake news, we have looked into different types of classification models and their speed and performance in detecting fake news on social media. To this end, we suggest using a Passive-Aggressive Classifier with TF-IDF vectorisation for its potent blend of training speed and classification performance.

Appendix 1: Full Results Tabulation (Sorted by SPACE(24) score)

Model	Vectoriser	Loss	Accuracy	Precision	Recall	AUC	F1 Score	Time (sec)	SPACE(24)
PAC	TFIDF	2.4210	0.9299	0.9226	0.9416	0.9297	0.9320	0.0133	1.000000
PAC	COUNT	2.7841	0.9194	0.9268	0.9141	0.9195	0.9204	0.0271	0.999918
MNB	TFIDF	0.2393	0.9141	0.8862	0.9542	0.9788	0.9189	0.1616	0.675313
MNB	COUNT	0.2969	0.9188	0.9180	0.9233	0.9766	0.9206	0.1696	0.673736
BILSTM+CNN	WORD2VEC	0.2218	0.9357	0.9320	0.9427	0.9769	0.9374	74.7573	0.208596
BILSTM	WORD2VEC	0.1916	0.9328	0.9448	0.9221	0.9808	0.9333	67.0973	0.188647
LSTM+CNN	WORD2VEC	0.2677	0.9311	0.9227	0.9439	0.9724	0.9332	44.0183	0.187933
LSTM	WORD2VEC	0.2031	0.9305	0.9415	0.9210	0.9768	0.9311	37.3143	0.178312
RFC	TFIDF	0.2431	0.9246	0.9256	0.9267	0.9771	0.9262	2.7267	0.158668
RFC	COUNT	0.2406	0.9235	0.9314	0.9175	0.9762	0.9244	3.0639	0.151109
CNN	WORD2VEC	0.2431	0.9217	0.9138	0.9347	0.9715	0.9241	26.4311	0.149364
XGB	COUNT	0.2153	0.9106	0.9176	0.9061	0.9729	0.9118	1.4559	0.121036
XGB	TFIDF	0.2199	0.9106	0.9128	0.9118	0.9709	0.9123	1.9851	0.114297
GNB	TFIDF	5.4070	0.8435	0.7969	0.9301	0.8424	0.8584	1.1759	0.030957
GNB	COUNT	5.8709	0.8300	0.7704	0.9496	0.8273	0.8507	1.1801	0.024928
XGB	WORD2VEC	0.3904	0.8364	0.8233	0.8648	0.9106	0.8436	2.2011	0.017281
RFC	WORD2VEC	0.4418	0.7891	0.7777	0.8213	0.8808	0.7989	1.1470	0.005592
GNB	WORD2VEC	17.0936	0.5047	0.5073	0.9897	0.4943	0.6708	0.0259	0.002659
MNB	WORD2VEC	11.6501	0.6606	0.7433	0.5109	0.7049	0.6056	0.0110	0.000539
BILSTM	COUNT	0.5936	0.7056	0.7183	0.6953	0.7588	0.7066	2058.7930	0.000240
CNN	COUNT	0.5933	0.7068	0.7238	0.6873	0.7388	0.7051	208.7419	0.000228
LSTM	COUNT	0.5997	0.7050	0.7244	0.6804	0.7316	0.7017	956.0869	0.000203

LSTM+CNN	COUNT	0.5791	0.7120	0.7661	0.6266	0.7690	0.6894	1120.5744	0.000133
BILSTM+CNN	TFIDF	0.6296	0.6548	0.6466	0.7125	0.7015	0.6779	2225.3081	0.000089
BILSTM+CNN	COUNT	0.5784	0.7103	0.7860	0.5934	0.7705	0.6762	2272.5826	0.000084
LSTM	TFIDF	0.6975	0.5099	0.5099	1.0000	0.6174	0.6754	1045.5528	0.000081
LSTM+CNN	TFIDF	0.6291	0.6525	0.6466	0.7022	0.7024	0.6733	1122.7999	0.000075
BILSTM	TFIDF	0.6531	0.6209	0.6055	0.7365	0.6634	0.6646	1997.6188	0.000055
CNN	TFIDF	0.6576	0.6139	0.5987	0.7365	0.6559	0.6605	195.9644	0.000048
PAC	WORD2VEC	16.1398	0.5327	0.5356	0.6289	0.5308	0.5785	0.0953	0.000021

Legend:

COUNT: Count Vectorisation

TFIDF: TF-IDF Vectorisation

WORD2VEC: Word2Vec Vectorisation

CNN: TextCNN Model

BiLSTM: Bidirectional LSTM Model

XGB: XGBoost Classifier

RFC: RandomForest Classifier

GNB Gaussian Naive Bayes Classifier

MNB: Multinomial Naive Bayes Classifier

PAC: Passive Aggressive Classifier

Appendix 2: Group Contribution

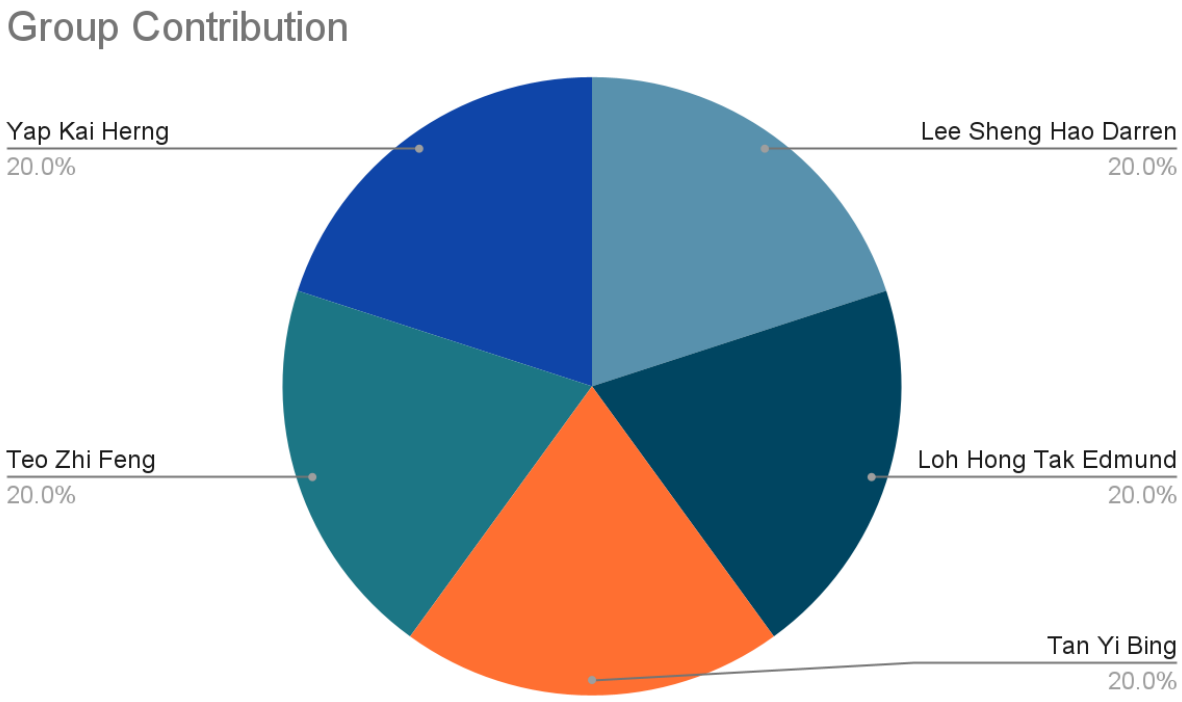

Figure 14: Everyone contributed equally to everything! :)

References

BBC News. (2020, April 24). Coronavirus: Trump's disinfectant and sunlight claims fact-checked. https://www.bbc.com/news/world-us-canada-52399464

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Chew, H. M. (2020, May 21). 6 in 10 people in Singapore have received fake COVID-19 news, likely on social media: Survey. CNA. https://www.channelnewsasia.com/singapore/fake-covid-19-news-study-ncid-messaging-platforms-whatsapp-673131

Depoux, A., Martin, S., Karafillakis, E., Preet, R., Wilder-Smith, A., & Larson, H. (2020). The pandemic of social media panic travels faster than the COVID-19 outbreak. Journal of Travel Medicine, 27(3). https://doi.org/10.1093/jtm/taaa031

Freeman, D., Waite, F., Rosebrock, L., Petit, A., Causier, C., East, A., Jenner, L., Teale, A. L., Carr, L., Mulhall, S., Bold, E., & Lambe, S. (2020). Coronavirus conspiracy beliefs, mistrust, and compliance with government guidelines in England. Psychological Medicine, 1–13. https://doi.org/10.1017/s0033291720001890

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1), 1-309.

KPMG. (2020, July). The Impact of Covid-19 - A Global to Local Overview. https://assets.kpmg/content/dam/kpmg/sg/pdf/2020/08/The-Impact-of-COVID-19.pdf

Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J., & Zittrain, J. L. (2018). The science of fake news. Science, 359(6380), 1094–1096. https://doi.org/10.1126/science.aao2998

Lim, Y.L. (2021, October 25). Pofma correction direction issued to Truth Warriors website over Covid-19 false claims. *The Straits Times*. https://www.straitstimes.com/singapore/correction-direction-issued-to-truth-warriors-website

Nasir, J. A., Khan, O. S., & Varlamis, I. (2021). Fake news detection: A hybrid CNN-RNN based deep learning approach. *International Journal of Information Management Data Insights*, 1(1), 100007.

Nyilasy, G. N. (2021, November 1). Fake news in the age of COVID-19. Pursuit. https://pursuit.unimelb.edu.au/articles/fake-news-in-the-age-of-covid-19

Preston, S., Anderson, A., Robertson, D. J., & Huhe, N. (2021, November). *Detecting fake news on Facebook: The role of emotional intelligence*. PLOS ONE. https://doi.org/10.1371/journal.pone.0246757

Revez, J., & Corujo, L. (2020, December). *Librarians against fake news: A systematic literature review of library practices (Jan. 2018–Sept. 2020)*. The Journal of Academic Librarianship. https://doi.org/10.1016/j.acalib.2020.102304

Stewart, E. (2021). Detecting Fake News: Two Problems for Content Moderation. Philosophy & Technology. Published. https://doi.org/10.1007/s13347-021-00442-x

Surowiecki, J. (2021, November 12). Covid misinformation spreads because so many Americans are awful at math. *The Washington Post*. https://www.washingtonpost.com/outlook/math-covid-vaccinations-jeremy-mcanulty/2021/11/12/bfe89018-417f-11ec-a3aa-0255edc02eb7_story.html

TensorFlow. (n.d.). *Recurrent neural networks (RNN) with Keras.* https://www.tensorflow.org/guide/keras/rnn

Uscinski, J. E., Enders, A. M., Klofstad, C., Seelig, M., Funchion, J., Everett, C., Wuchty, S., Premaratne, K., & Murthi, M. (2020). Why do people believe COVID-19 conspiracy theories? Harvard Kennedy School Misinformation Review. Published. https://doi.org/10.37016/mr-2020-015

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380), 1146-1151.

Yoon, K. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630*.