Detecting fake news: a CNN and RNN approach

Orianna Akker [*1] **Mengqi Hu** [*1 2] **Nabeel Hussain** [2]

Abstract

This study investigates the use of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) techniques in classifying fake news. Four different models, namely 2-input CNN, 2-input RNN, 1-input CNN, and 1-input RNN, are proposed and evaluated on the binary text classification task. This study found that the 1-input CNN model achieved the highest accuracy of 96.7%, and that overall, the models using a singular, combined input achieved a much higher accuracy than those with two, separate inputs. The high accuracy achieved demonstrates the usefulness of neural networks in detecting fake news and provides insights for future research in this area.

1. Introduction

As the internet continues to be even more heavily relied on in everyday life and social media is ever present, the presence of fake news online is an ever-growing problem that 86% of the global online population have been exposed to(**?**). More concerningly, of these 86% of individuals who had been exposed, 86% admit to having fallen for the fake news at least once(Ipsos). It is clearly vital to find a way to confidently distinguish between real and fake news and this paper focuses on applying deep learning methods to a selection of online posts with the aim of classifying them as fake or real.

1.1. The Data Set

The data set used within this study consists of 44,898 internet posts, with 23,481 fake posts and 21,417 real posts. (Note that fake posts refer to real, published posts which contain false information that is posed as true information.) The data consists of the title and text of the posts/articles, their general subject of focus, and the date they were posted. The posts in this data set are from March 2015 - February 2018.

Upon investigation, it is found that the subject variable consists of different categories for the real and fake instances, and so this category is not fed into the neural network models. Additionally, the date variable does not provide any useful information with respect to real/fake classification, and thus this variable is also discarded.

1.2. Project Framework

In this report, CNN and RNN techniques as well as Naive Bayes methods, Decision Tree methods and Random Forest methods are applied to carry out fake news detection. A variety of structures and regularisation techniques were employed to achieved a high prediction accuracy and precision on test data.

The main results of our work found that the 1-input CNN model performed the best out of all four proposed models, with a test accuracy of 96.7%, while the 2-input CNN model achieved nearly only 78.5% test accuracy. Interestingly, it was also found that the 1-input RNN model massively outperformed it's 2-input counterpart, with over a 40% increase in test accuracy.

The results showed that simplifying and concatenating the information of the input data may lead to significantly higher accuracy in model fitting. The 1-input CNN model is clearly a suitable model for this problem as performs well using all metrics, has a low loss value, and compiles quickly. The 1-input RNN also achieves a high accuracy of 94.3% suggesting this is also a suitable solution to this task.

2. Related Work

Yang et al. proposed a TI-CNN model (Text and Image information based Convolutional Neural Network model) to solve the fake news identification problem, and achieved a 92.2% precision rate using this model. This model might not translate exactly to the data set used in this report since there are no images, but some convolutional techniques used could be incorporated(Yang et al., 2018).

Sastrawan et al. investigated the use of multiple models over different datasets and using varying pre-trained word embeddings. On one of the datasets, the CNN model achieved a 99.88% accuracy, the bidirectional LSTM achieved a 99.95% accuracy, and the ResNet model achieved 99.90% accuracy. The LSTM and ResNet models both used GLoVe word embedding models. However, the authors did not test their model on new datasets, and so the possibility of overfitting has not been eliminated in the bidirectional LSTM(Sastrawan et al., 2022). Similarly, Chauhan investigates the use of an LSTM neural network model with a GLoVe word embedding method in detecting fake news. Their hybrid RNN model achieved 99.88% accuracy on their chosen dataset, but again was not tests on any other datasets.(Chauhan & Palivela, 2021). GLoVe pre-trained word embeddings are clearly a popular and effective choice, and thus it chosen to incorporate these into our proposed models.

Barua et al. propose an ensemble technique that combines LSTM models and GRU models, and their model achieves 81% accuracy on their fake news dataset. Their model has significantly lower accuracy than the models that have already been discussed, but the model did show a maintained accuracy when tested on other datasets(Barua et al., 2019).

Nasir et al. use a hybrid CNN-RNN approach to make use of CNN's ability to extract local features and of LSTM's ability to learn long-term dependencies. Their hybrid model achieves 60% accuracy on one fake news dataset and 99% accuracy on the other. Although 60% accuracy is very low, it still performed much better than all other models tested, such as Random Forest, K-Nearest Neighbours, and CNN and RNN alone(Nasir et al., 2021).

Solovyeva and Abdullah propose a seperable CNN structure for general binary text classification tasks. The model consists of an embedding layer, separable convolutional layers, a convolutional layer and a global average pooling layer. Their model achieved 79.4% accuracy on the binary text classification task, which was higher than that of the LSTM, GRU and RNN models that it was compared to(Solovyeva & Abdullah, 2021).

Soni et al. propose their model, TextConvoNet - a novel CNN for binary (and multi-class) text classification. The model aims to identify n-gram features between words of the different sentences and the intra-sentence n-gram feature. The model achieved it's highest accuracy of 99.2% on the third dataset tested, and outperformed all other compared models on this data, including LSTM's, Random Forests, and VDCNN(Soni et al., 2022).

The models proposed in this report build upon the basics of models proposed in these related works. RNN and CNN methods are utilised in most of the related texts and so these techniques are incorporated into our models. Additionally, pretrained word embeddings appear to improve model accuracy, and so it is decided to utilise this within our model.

3. Model Architecture

This report compares the use of Convolutional Neural Newtork techniques and Recurrent Neural Networks, and compares the findings to results from a Naive Bayes model, a decision tree model, and a random forest model. Separate input of the text and title data (2-input) is compared to the combined input (1-input), where the text and title data are concatenated to create one variable.

3.1. 2-input CNN/RNN Models

As seen in the Relevant Work section, it is common to apply CNN and RNN techniques to text classification tasks and achieve a high accuracy.

The RNN components of the model process the output of the CNN layers by taking into account the sequence information and modeling the temporal dependencies between the inputs. This allows the model to capture long-term dependencies and dependencies between different parts of the input sequence(Liu et al., 2016).

3.1.1. CNN Model

The CNN-based model applies makes use of convolutional filters to learn local features in the input data, as well as pooling layers that reduce the dimensionality of the feature maps(Moriya & Shibata, 2018).

The parallel CNN model separately feeds the text and title inputs into an embedding layer that maps each word to a dense vector representation. This layer uses an embedding matrix that has been pre-trained using GLoVe word embedding. The resulting embedded sequences are then passed through three different convolutional layers with varying filter sizes and numbers of filters, which aim to detect differing local features. These 6 layers are then passed to Max Pooling layers that reduce the dimensionality of the output. Global Average Pooling layers are applied which aim to reduce the spatial dimensions of the feature maps and produce a compact representation that summarizes the most important information in the input data (Ghosh et al., 2018). The three title layers are concatenated together, and the three text layers are also concatenated, and both are reshaped. The merged layers are then individually passed twice through convolutional blocks, consisting of convolutional layers and max pooling layers. A Global Pooling layer is then applied. A dropout layer is applied to provide some regularisation before the two separate paths are merged into one. A further dense and dropout layer are applied before the output layer, which uses a sigmoid activation function to predict the 'fake' or 'real' news class. The detailed structure of this CNN model is shown in Figure 1. For this CNN model, there are 21,920,797 parameters in total, with 179,597 trainable parameters and 21,741,200 non-trainable parameters.

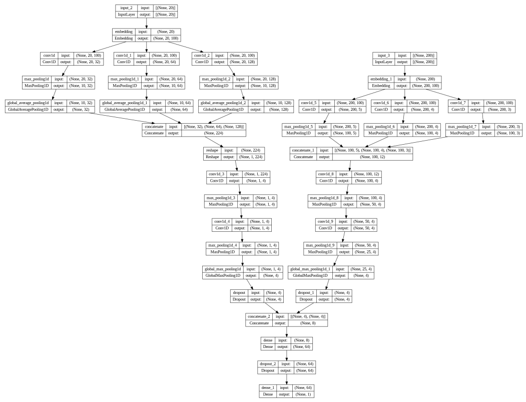


Figure 1. 2-input CNN structure

3.1.2. RNN Model

Similarly, the RNN model separately feeds the text and title inputs into an embedding layer that maps each word to a dense vector representation, again using GLoVe word embedding. Each branch of the model uses an embedding layer and one convolutional block (consisting of a convolutional layer and a max pooling layer), followed by two Long Short-Term Memory (LSTM) layers. A convolutional block is employed since convolutional layers in a deep neural network can help to increase the model's ability to learn complex features and patterns in the input data (Liu et al., 2015), and is particularly useful in capturing text and language patterns. The LSTM layers are applied in this model to try and capture long-term dependencies within the data, whilst avoiding any exploding or vanishing gradient problems. This is particularly applicable to text data which is inherently sequential. The two paths are then merged and passed through two dense layers. The parallel 2-input RNN model is shown in Figure 2.

3.2. 1-input CNN-RNN Models

3.2.1. CNN Model

The 1-input CNN model is largely similar to the 2-input but with a few modifications. The concatenated title-text data is singularly fed into this model, and so the parallel aspects of the 2-input model are not transferred. This model uses three convolutional blocks consisting of a convolutional layer, leaky ReLU activation, and a max pooling layer. Three dense layers are also used, and dropout layers are used throughout. This model is much simpler and more compact than the 2-input version. The 1-input CNN model is shown in Figure 3

3.2.2. RNN Model

The 1-input RNN model is almost the same as it's 2-input counterpart. The model uses an embedding layer, a convolu-

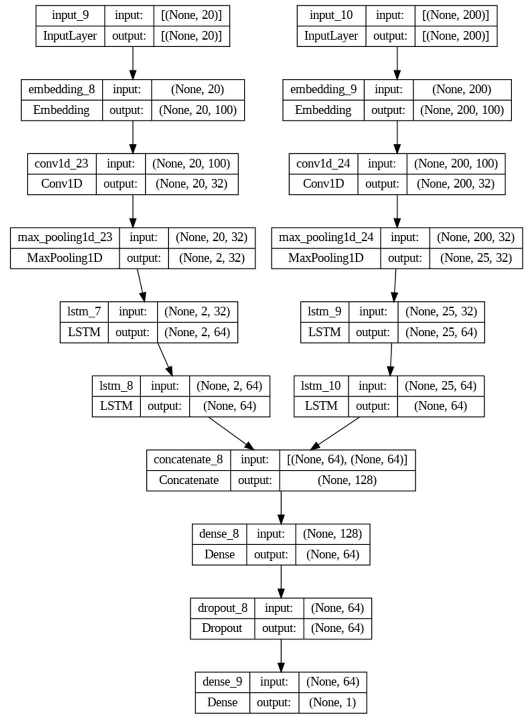


Figure 2. 2-input RNN structure

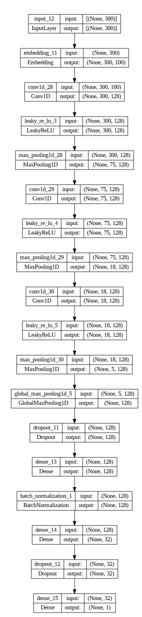


Figure 3. 1-Input CNN structure

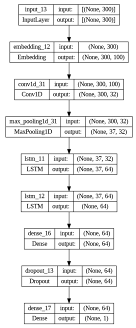


Figure 4. 1-Input RNN structure

tional block, two LSTM layers, and two dense layers. The 1-input RNN model is shown in Figure 4.

3.3. Other Models

The Naive Bayes method, Decision Tree method and Random Forest method are used to compare the predictive value of the proposed CNN and RNN models. These three models are popular data mining models that aim to predict categorical variables, and many are used to evaluate proposed models in related work.

Naive Bayes is a probabilistic algorithm based on Bayes' theorem that assumes that the features are conditionally independent of each other given the class label(Webb et al., 2010). It works by calculating the probability of each class given the input features and selecting the class with the highest probability. Naive Bayes is relatively simple, fast, and requires relatively little training data, making it a popular choice for text classification, spam filtering, and sentiment analysis(Gong & Yu, 2010; Dey Sarkar et al., 2014).

Decision Tree is a non-parametric algorithm that builds a tree-like model of decisions and their possible consequences(Myles et al., 2004). It works by recursively splitting the data into subsets based on the feature that provides the most information gain for the classification task. Each leaf node represents a class label, and each internal node represents a decision based on the value of a feature. Decision Tree is a powerful and interpretable algorithm that can handle both categorical and continuous data, but can be prone to overfitting and can be sensitive to small changes in the data(Li & Jain, 1998; Ikonomakis et al., 2005).

Random forest is a popular machine learning algorithm used for both classification and regression tasks(Rigatti, 2017). It is an ensemble learning method that combines multiple decision trees to create a forest of trees. Each decision tree in the forest is trained on a random subset of the features and data(Ali et al., 2012). The final output of the model is the average of the outputs of all the trees in the forest, which helps to reduce overfitting and improve the accuracy of the model(Bernard et al., 2009). Random forest has several advantages over other machine learning algorithms, including its ability to handle large datasets, high accuracy, and resistance to overfitting especially in the field of text classification(Xu et al., 2012; Kowsari et al., 2019).

All three methods are loaded directly from the 'sklearn' package in python and are fed the same inputs as the proposed CNN and RNN models.

4. Training Methods

The data is throroughly pre-processed before being fed into the neural network models, optimal parameters are set and various techniques are employed to prevent overfitting.

4.1. Data Pre-processing and Tokenisation

The subject and date columns provide little value and are removed from the data set. The next step is to create two versions of the data: one with the text and title combined into one column, and the other that keeps them separate. The combined data is created by concatenating the title followed by the text, with a space in between, and is used in the 1-input CNN model(Figure 3) and 1-input RNN model. The separates data maintains the text and title data in 2 different columns and processes each column separately, and is used in the 2-input CNN and 2-input RNN models(Figure 1 and Figure 2) .

Then, stop words are eliminated from all of the text data. Stop words are commonly used words that are generally considered to be of little value in understanding the meaning of a text, such as 'a' and 'the'. Two stop words packages are downloaded and combined to provide a wider berth of words to compare the text data to. A function is defined that filters out all stop words and words with length less than three from the data, as these do not provide much prediction value.

GLoVe (Global Vectors for Word Representation) pre-trained word embeddings are utilised to create an embedding matrix that can be used in the embedding layer of a neural network. The GLoVe word embeddings are pre-trained vectors of words in a given corpus. The GLoVe embedding of dimension 100 is used, meaning each word in the vocabulary is represented by a vector of length 100. The GLoVe representations are stored in a dictionary and are used to create an embedding matrix which holds the GLoVe representation of relevant words in the vocabulary of the text data. This embedding matrix is then utilised in the embedding layers of all of the neural network models, instead of using randomly initialized vectors. This allows the neural network to leverage pre-existing semantic relationships among words in the pre-trained embeddings, and improve the model's performance on text classification.

The final step in pre-processing the data is to divide the dataset into the training data, validation data, and test data. An 80/20 split is used where 80% of the data is used to train the models, and 20% of the data is used to test the models. Within the 80% of training data, 20% is used as validation data whilst the model is being trained.

4.2. Parameter Settings

Once the models have been established, the optimizers, regularisation techniques, and their optimal parameters are decided on.

The optimizers package in Keras provides a collection of optimization algorithms that can be used to train neural networks. These algorithms are used to minimize the loss function during the backpropagation phase of training. In the sensitivity analysis part of this project, we applied SGD, Adam and different learning rates to test the stability of the best model. Subsequently, the Adam optimizer is chosen to compile the models with, and it is found that a learning rate of 0.0001 provides the highest test accuracy. Adam is an adaptive learning rate optimization algorithm that combines the advantages of AdaGrad, which performs well for sparse gradients, and RMSProp, which performs well for online and non-stationary problems. It's adaptive learning rate makes it particularly well-suited to handle large datasets and complex models, as well as being computationally efficient.

Dropout layers, L2 regularisation, Batch Normalisation, and Early Stopping are all regularisation techniques that have been utilised within the models to prevent overfitting of the data.

Dropout layers are employed within all of the proposed models, where a proportion of the nodes in the neural network at that stage of training are randomly dropped out of the network. The proportion is defined by the dropout rate. Thus, the model is forced to learn the more robust features present and no single node has the ability to become overly important within the model's classification.

L2 regularisation is utilised in every model, and works by adding a penalty term to the loss function during the training process. The penalty term is proportional to the squared magnitude of the model weights, and prevents overfitting by discouraging large weight values. The L2 parameter is set to 0.01 as this was found to produce the most accurate models.

Batch Normalisation is used in one of the final few layers of the 1-input CNN model. This layer helps to stabilize the distribution of it's inputs by normalizing them. This can prevent the activation function from saturating as well as reducing the dependence of the model's output on the specific values of the inputs, and thus can lead to a faster implementation of the model.

Early stopping terminates the training process once the model's performance on the validation set begins to worsen over the training epochs, and this technique is utilised for all the models. This technique helps to prevent overfitting of the data by stopping the training process before the model has the chance to memorize the training data.

Additionally, in the model fitting stage, the number of epochs is set to 100, as this was found to provide enough berth for the early stopping algorithm to terminate the training process. The batch size is set to 256 as this was found to strike a good balance between the speed of training and the stability of the optimization process.

5. Numerical Results

5.1. Evaluation Metrics

The following measures are utilised as evaluation indicators of the models applied to the test data. (Note TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative):

- **Accuracy -** $\frac{TP+TN}{TP+TN+FP+FN}$:

  This metric measures the percentage of correct predictions made by the model. It is defined as the ratio of the number of correct predictions to the total number of predictions made by the model. It is noted that accuracy can sometimes be misleading when the dataset is imbalanced, but the instances of fake and real news are balanced in this case, and so accuracy is a useful indicator of performance of our models.

- **Precision -** $\frac{TP}{TP+FP}$:

  This metric measures the proportion of true positives, i.e. real news, amongst all the predicted positives. Precision is useful when the goal is to minimize false positives, which is particularly relevant as it is clearly more damaging to misclassify fake news than real news.

- **Recall -** $\frac{TP}{TP+FN}$:

  This metric measures the proportion of true positives among all the actual positives. Recall is useful when the goal is to minimize false negatives, such as in disease diagnosis, and is perhaps less important than precision in the context of this report.

- **F1 score -** $\frac{TP}{TP+\frac{1}{2}(FP+FN)}$:

  This metric is a harmonic mean of precision and recall. It is a balanced metric that takes both precision and recall into account. F1 scores range from 0 to 1, where 1 is indicative of the best performance. F1 score is commonly used when the data set is imbalanced, and both precision and recall need to be considered, and again

Model	Accuracy	Precision	Recall	F1 Score	Loss
2-input CNN	78.5%	78.5%	78.5%	78.5%	0.533
2-input RNN	51.3%	38.6%	50.0%	34.3%	2.17
1-Input CNN	96.7%	96.7%	96.7%	96.7%	0.179
1-Input RNN	94.3%	94.2%	94.3%	94.3%	0.271
Naive Bayes	57.0%	57.8%	57.5%	56.7%	-
Decision Tree	93.3%	93.3%	93.3%	93.3%	-
Random Forest	89.7%	89.7%	89.7%	89.7%	-

Table 1. Evaluation of Models

Optimizer	Learning rate	Regularizer	Accuracy	Precision	Recall	F1 Score
Adam	0.0001	L1	0.74	0.75	0.74	0.74
Adam	0.001	L1	0.54	0.68	0.53	0.42
Adam	0.01	L1	0.50	0.29	0.48	0.33
Adam	0.0001	L2	0.79	0.81	0.79	0.78
SGD	0.0001	L1	0.53	0.54	0.53	0.53

Table 2. The sensitivity analysis on parallel CNN

is perhaps not as pertinent as precision and accuracy within this context.

The loss function used when compiling and training the models is binary cross-entropy loss. The function calculates loss as the negative log-likelihood of the true class. This function is chosen as it is suitable for a binary classification task and is well-fitted to data sets with an approximately equal number of samples in the two classes.

5.2. Numerical Results

The primary goal of this evaluation was the assess the performance of our proposed models for binary text classification. As mentioned, this study was performed on the Kaggle dataset: Fake and Real News.

5.2.1. Dataset

Fake and Real News The data set used consists of 44,898 internet posts, with 23,481 fake posts and 21,417 real posts. The data consists of the title and text of the posts. 20% of data was used to to test the model performance, and a 80:20 train:validation split on the remaining data was used to train the model.

5.2.2. Results

The performance of our proposed models are compared against the three baseline methods: Naive Bayes, Decision Trees, and Random Forests. The baseline methods are tests on the 1-input version of the data. The models are evaluated using the metrics stated and Table 1. shows the performance of each model when applied to the test data.

As can be seen from Table 1., the 2-input models are both outperformed by the Decision Tree and Random Forest method. However, the Naive Bayes method does not perform very well on this data set with only 57% accuracy and all the proposed models except the 2-input RNN outperform this model. It is noted that the baseline method results are based on 1-input data.

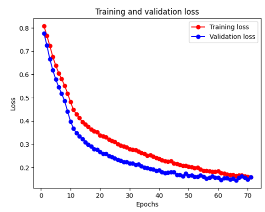
It is seen that the 1-input models achieved the highest accuracies of 96.7% and 94.3%. They also had much a lower loss value than the 2-input models. The 2-input CNN model achieved an accuracy of 78.5%, which is lower than the 1-input models but still outperformed the 2-input RNN model, which achieved a low accuracy of only 51.3%.

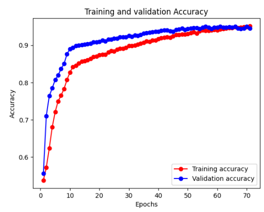
In terms of precision, recall, and F1 score, the 1-input CNN and 1-input RNN models also outperformed the other models, achieving highly in all three metrics. Interestingly, all of the models perform quite consistently across each of their evaluation metrics, except for the poorly performing 1-input RNN model, which has a very low F1 score of 34.3%.

Figure 5 shows the trends of training loss, validation loss, training accuracy, and validation accuracy over the training epochs for each model. In Figure 5a and Figure 5b, the loss function of the 2-input CNN converges at around 65 epochs, and the training and validation accuracy converge at around 60 epochs also. In Figure 5c and Figure 5d, the training and validation loss do not appear to be converging, and neither do the training and validation accuracies. They would perhaps converge if patience of the Early Stopping regulariser was extended, but this is found to reduce the test accuracy. In Figure 5e and Figure 5f, the training and validation loss of the 1-input CNN model converge quickly at around 10 epochs, and the accuracies are seen to converge at around 7 epochs. In Figure 5g and Figure 5h, the training and validation loss of the 1-input RNN model converge almost immediately, and the same can be said for the accuracies. It appears that a higher accuracy and lower loss is achieved over much fewer epochs in the 1-input models than in the 2-input ones.

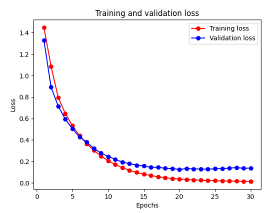
5.3. Sensitivity Analysis: 2-input CNN

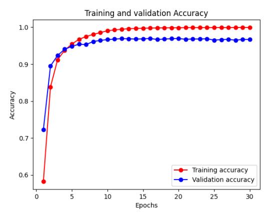
Sensitivity analysis is conducted, through the application of different optimizers, learning rates, and regularisation techniques, on the 2-input CNN model to more deeply explore it's optimality. Using the Adam optimizer, three different learning rates are explored for training the model. It is found that larger learning rates lead to a smaller accuracy, precision, recall and F1 scores. Fixing the optimizer and learning rate, the applications of L1 and L2 regularizers are compared, and it is found that the accuracy and precision of the model with the L2 regularizer is better than that with the L1 regularizer. It is also found that the test accuracy and precision decrease when the SGD optimizer is used in place of the Adam optmizer. Overall, it is found that applying the L2 regularizer improves the overall model. The results of the sensitivity analysis are summarised below in Table 2.



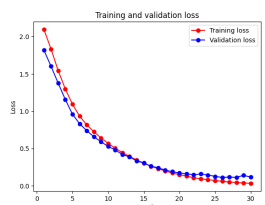


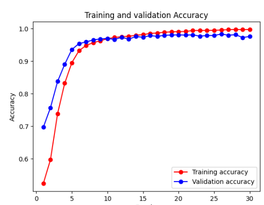
(a) Loss Plot of 2-input CNN
(b) Accuracy Plot of 2-input CNN



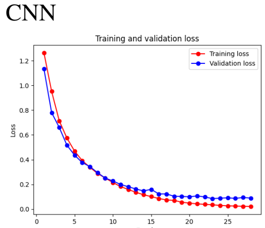


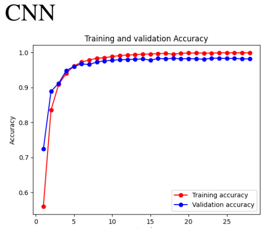
(c) Loss Plot of 2-input RNN
(d) Accuracy Plot of 2-input RNN





(e) Loss Plot of 1-Input CNN
(f) Accuracy Plot of 1-Input CNN





(g) Loss Plot of 1-Input RNN
(h) Accuracy Plot of 1-Input RNN

Figure 5. Accuracy and Loss Plot

6. Conclusion

In this study, four different proposed models were evaluated on their performance in a binary text classification task involving classifying real and fake news posts. The results indicate that the 1-input models largely outperform the 2-input models across all of the evaluation metrics used. This indicates that there is real predictive value added by concatenating the related text inputs, i.e. text that all comes from the same article/post. It also suggests the neural network is not very successful at linking together the two separate inputs when run in parallel.

The Decision Tree, and Random Forest models also performed reasonably well but were outperformed by the deep learning models. The Naive Bayes method, however, did not perform well, although it did perform better than the 2-input RNN model. The 2-input RNN model achieves an accuracy of only 51.3%, which is only minutely better than the expected accuracy of 50% that would be achieved by randomly guessing the classification. In fact, it is interesting that there is such a massive improvement in accuracy for the 1-input RNN model, that achieves 94.3% accuracy, which is almost entirely the same as the 2-input version (but modified to take in a singular input). This again confirms the conclusion that concatenating the text and title data into one input adds much predictive value.

The highest performing 1-input CNN model achieved a higher accuracy than the majority of models covered in related works. For example, our 1-input CNN model achieved a higher accuracy that Yang et al.'s TI-CNN-1000 model which had an accuracy of 92.2%(Yang et al., 2018). However, our model still underperformed against Chauhan's hybrid RNN model which achieved an accurcy of 99.88% on their chosen dataset(Chauhan & Palivela, 2021), suggesting there is still room for improvement in our 1-input CNN model.

Overall, the 1-input CNN and 1-input RNN models seem to be the most promising models for this fake news binary text classification task based on their high accuracy and good performance in precision, recall, and F1 score. However, further analysis and experimentation may be needed to confirm these findings.

7. Discussion

Although high accuracy was achieved within the proposed models, there are still a host of limitations within this study. The data set used is limited only to news posts from 2016-2017, and is taken from a very small set of sources, and thus has a very limited scope. In fact, most of the news articles seem to involve *Trump*, suggesting there is a very narrow focus of all the news posts (See Appendix 8.1). Future work could involve evaluating the proposed models on a wider diversity of datasets to test their generalisability. Additionally, hyperparameter tuning could be explored in further depth to potentially obtain a more optimal model performance.

The time period of the data does not cover any of the Covid-19 pandemic, when the spread of fake news was rife, and this could provide an interesting further scope for a similar study. Deep learning methods are constantly being developed and improved upon, and perhaps new techniques could be applied to this task in the future. As video-based social media sites like Tiktok are growing increasingly popular, it would be clearly be beneficial to develop neural network models that can distinguish between fake and real news posted in such videos, and this provides further scope for future research.

The models proposed in this study have vital applications in the real world. They have the potential to be utilised by internet users to help them distinguish between fake and real news and news sources, thus allowing people to begin trusting what they read online again. However, as mentioned before, the models would need to be tested on a wider variety of datasets before potentially being used in such an application.

In summary, this study demonstrates the effectiveness of CNN and RNN models for binary text classification tasks and highlights the importance of feature engineering and data pre-processing for achieving high model accuracy. However, there is still much room for improvement and scope left to explore in future work.

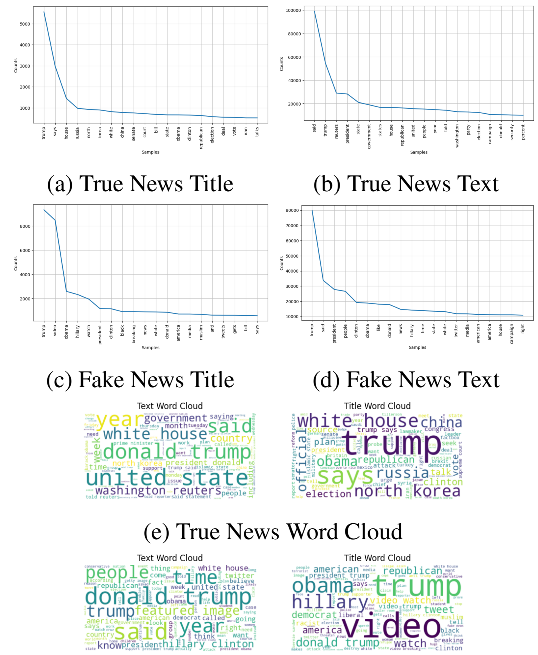
8. Appendix

8.1. High Frequency Words

The Figure 6 shows the top 20 most frequent words in the true news titles(Figure 6a), the true news texts(Figure 6b), the fake news titles(Figure 6c) and the fake news texts(Figure 6d). The top 5 most frequent words in the true news titles are *trump*, *says*, *house*, *russia* and *north*. The top 5 most frequent words in the true news text are *said*, *trump*, *reuters*, *president* and *state*. The top 5 most frequent words in the fake news titles are *trump*, *video*, *obama*, *hillary* and *watch*. The top 5 mosst frequent words in the fake news texts are *trump*, *said*, *president*, *people* and *clinton*. Figure 6e and 6f represent word cloud depictions for the true news and fake news articles, where the size of the words are proportionate to their relative frequency.

8.2. Types of News

Figure 7 depicts the distribution of subjects of the news articles. It is seen that true news articles are categorised into politics and world news, whilst fake news articles are categorised into ordinary news, politics news, government news, left news, US news and middle east news. The proportion of politics news and world news in the true news data is approximately equal(Figure 7a). The category of ordinary news accounts for the majority of fake news articles(Figure 7b). It is noted that the fake news articles are categorised differently to the true news articles (or at least they cover a broader range of subjects), and thus news subject does not appear to be a very useful category in terms of classification.



(a) True News Title (b) True News Text

(c) Fake News Title (d) Fake News Text

(e) True News Word Cloud

(f) Fake News Word Cloud

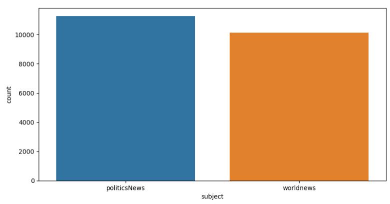
Figure 6. Frequency of Words in Different Contents

8.3. Word Counts of Articles

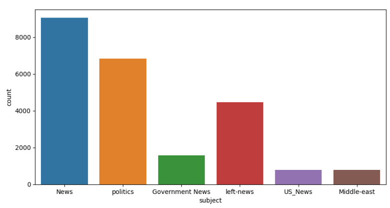
Figure 8 shows the number of words in each title and main text. More than 4,000 of the true news titles are 10 or 11 words, while the maximum number of words in the titles of true news articles is 24(Figure 8a). More than 3,500 true news main text chunks are about 500 words, while the maximum number of words in the text of true news articles is 5979 (Figure 8b). The total number of words in fake news titles are around 16-21, whilst the maximum number of words in fake news titles is 54(Figure 8c). The word count in the fake news text is quite similar to the true news text, whilst the maximum number of words number (9,956) is much larger than that of the true news text(Figure 8d).

9. Statement of Contribution

MengQi formulated and wrote the data pre-processing code, explored the data, and wrote the code for the baseline methods. Orianna created the 4 neural network models. Orianna and Mengqi co-wrote the report. Nabeel attempted to code a transformer but did not contribute anything else.

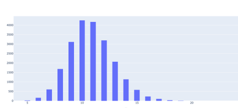


(a) True News Title

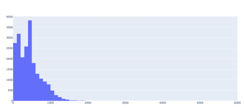


(b) True News Text

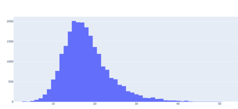
Figure 7. Types of True and Fake News



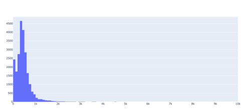
(a) True News Title



(b) True News Text



(c) Fake News Title



(d) Fake News Text

Figure 8. Word Counts in Different Contexts

References

Ali, J., Khan, R., Ahmad, N., and Maqsood, I. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

Barua, R., Maity, R., Minj, D., Barua, T., and Layek, A. K. F-nad: an application for fake news article detection using machine learning techniques. In *2019 IEEE Bombay section signature conference (IBSSC)*, pp. 1–6. IEEE, 2019.

Bernard, S., Heutte, L., and Adam, S. On the selection of decision trees in random forests. In *2009 International Joint Conference on Neural Networks*, pp. 302–307. IEEE, 2009.

Chauhan, T. and Palivela, H. Optimization and improvement of fake news detection using deep learning approaches for societal benefit. *International Journal of Information Management Data Insights*, 1(2):100051, 2021.

Dey Sarkar, S., Goswami, S., Agarwal, A., and Aktar, J. A novel feature selection technique for text classification using naive bayes. *International scholarly research notices*, 2014, 2014.

Ghosh, A., Bhattacharya, B., and Chowdhury, S. B. R. Adgap: Advanced global average pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Gong, Z. and Yu, T. Chinese web text classification system model based on naive bayes. In *2010 International Conference on E-Product E-Service and E-Entertainment*, pp. 1–4. IEEE, 2010.

Ikonomakis, M., Kotsiantis, S., and Tampakas, V. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

Ipsos. Fake news: A global epidemic vast majority (86%) of online global citizens have been exposed to it. URL `https://www.ipsos.com/en-us/news-polls/cigi-fake-news-global-epidemic`.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

Li, Y. H. and Jain, A. K. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

Liu, P., Qiu, X., and Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

Liu, T., Fang, S., Zhao, Y., Wang, P., and Zhang, J. Implementation of training convolutional neural networks. *arXiv preprint arXiv:1506.01195*, 2015.

Moriya, S. and Shibata, C. Transfer learning method for very deep cnn for text classification and methods for its evaluation. In *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*, volume 2, pp. 153–158. IEEE, 2018.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., and Brown, S. D. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004.

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

Rigatti, S. J. Random forest. *Journal of Insurance Medicine*, 47(1):31–39, 2017.

Sastrawan, I. K., Bayupati, I., and Arsa, D. M. S. Detection of fake news using deep learning cnn–rnn based methods. *ICT Express*, 8(3):396–408, 2022.

Solovyeva, E. and Abdullah, A. Binary and multiclass text classification by means of separable convolutional neural network. *Inventions 2021*, 2021.

Soni, S., Chouhan, S. S., and Rathore, S. S. Textconvonet: a convolutional neural network based architecture for text classification. *Applied Intelligence*, 2022.

Webb, G. I., Keogh, E., and Miikkulainen, R. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

Xu, B., Guo, X., Ye, Y., and Cheng, J. An improved random forest classifier for text categorization. *J. Comput.*, 7(12): 2913–2920, 2012.

Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., and Yu, P. S. Ti-cnn: Convolutional neural networks for fake news detection. *arXiv preprint arXiv:1806.00749*, 2018.