Prediction of Article Reliability

Text Mining & Language Analytic

March 2023

1 Introduction & Dataset

In this enquiry, A number of machine learning models were created to predict the reliability of news articles. The dataset was stored in a comma-separated file (csv) with 20,800 news articles from various sources. Each article is annotated as either reliable (0) or unreliable (1). The dataset file was organised into 5 columns: id, title, author, text and label.

The dataset contained some issues that needed to be handled in our data preparation phase. Among the 20,800 observations, there were 558 observations with missing title, 1957 observations with missing author, and 39 observations with missing text. Some of these data had interference for training, and the problem of news title and news text separation also needed to be solved.

In this dataset, there were 10,387 reliable news items (49.94%) and 10,413 unreliable news items (50.06%), which means that this was a very balanced data, ideal for machine learning binary classification model training. However, as we mentioned earlier, there were some problems with this data. The balance of the data still needed to be discussed again after the data had been pre-processed.

2 Data Preparation

Because the dataset contained 5 columns, as part of the news, the title needed to be combined with text into one column to make it a complete news article, while id, as irrelevant to the news, will be removed, and author, as persons' name, were too common in daily life and will also be removed to prevent adverse effects on the accuracy of the model. Then we obtained the dataframe containing the text and the label, which would be converted into lists.

Some news articles less than 50 characters long would be removed because of the low word count to avoid over-fitting. For the Naïve Bayes classifier and k Nearest Neighbours (KNN) classifier , we kept all the data satisfying the conditions, but because of insufficient computing power, for the Convolutional Neural Network (CNN) model and the Long Short-Term Memory (LSTM) model, all parts exceeding 300 words were removed. If under the condition of sufficient computing power, all the data was needed for training.

After cleaning the data, we obtained a data of 20,142 observations, of which 10,387 were reliable news, accounting for 51.57%, and 9,755 were unreliable news, accounting for 48.43%. In general, the number of reliable and unreliable news items was basically the same and very balanced, so it was still very suitable for the training of the binary classification model.

In the Naïve Bayes model and the kNN model, the Scikit-Learn `train_test_split` was used to split the dataset according to 70% as the training dataset and 30% as the test dataset. Although there were methods in pytorch to split the dataset, Neural Network also used both

datasets as training and test datasets in order to keep the data consistent. As we could see from Table 1, all the datasets were kept very balanced, so that no large errors were caused by imbalance in the data when training the model.

During data preparation, word_tokenize is used to tokenise words and then remove stop words and punctuation. For some empty elements after pre-processing, it was also necessary to remove these data. Then the text would be reformatted as a bag of words with all the words in text being elements in a list. For the neural networks, the labels and texts were likewise read and converted to lists.

	Total sample		**Training sample**		**Test sample**	
Label	**Frequency**	**Percentage**	**Frequency**	**Percentage**	**Frequency**	**Percentage**
Reliebale	10387	51.57%	7286	51.68%	3101	51.32%
Unreliebale	9755	48.43%	6813	48.32%	2942	48.68%

Table 1: Distribution of labels in the datasets

3 Text Representation

For the naive Bayesian and kNN models, the text representation was processed using the TfidfVectorizer function in Scikit-learn, which took the input text data and pre-processed it by tokenising words and removing stop words, and then calculated a feature vector for each word based on the product of the Term Frequency and Inverse Document Frequency. The product of Term Frequency and Inverse Document Frequency is used to calculate the feature vector of the word, which was used to indicate the importance of each word.

For Neural Network, the words and labels are indexed using the `TEXT.build_vocab` and `LABEL.build_vocab` functions in PyTorch, and the unknown words and padding words are indexed using the `TEXT.vocab.stoi` function. And when representing text, the `nn.Embedding` function vectorises the text as required. The advantages of word embedding allow not only to vectorise text without generating too much data as in One-Hot, but also to calculate the distance between different words.

4 Machine Learning Architectures

4.1 Naïve Bayes

Bayes' theorem is a formula for calculating conditional probabilities that describes how to update our probability estimates of events with new evidence, given that certain prior probabilities are known. The plain Bayesian model will calculate two posterior probabilities P(true — X) and P(false — X) and classify X into classes with higher posterior probabilities.

The Naive Bayes Model will calculate the likelihood from the frequency of occurrence of each word under each category in the training dataset and determine the prior probability by learning from the training dataset. Then, for new text data, the model will calculate posterior probabilities based on the prior probabilities and likelihoods and classify them into categories with higher posterior probabilities.

4.2 k-Nearest Neighbours

The basic principle of kNN is to find the k nearest training samples and predict the labels of the test samples based on their labels. In text classification, KNN typically uses a bag-of-words

model or TF-IDF features to represent texts, using Euclidean distance or cosine similarity to calculate the distance between two texts. The advantages are simplicity, ease of understanding and implementation, and the disadvantages are high computational complexity and large storage space.

4.3 Convolutional Neural Network (CNN)

The basic structure of Convolutional Neural Networks (CNNs) for natural language processing (NLP) includes convolutional layers, pooling layers, and fully connected layers. In text classification tasks, input text is usually transformed into word embedding vector representations. These vectors are then processed by the convolutional layers to extract relationships and importance among them. Next, the feature size is further reduced by pooling layers, making it suitable for classification using the fully connected layers.

4.4 Long Short-Term Memory (LSTM)

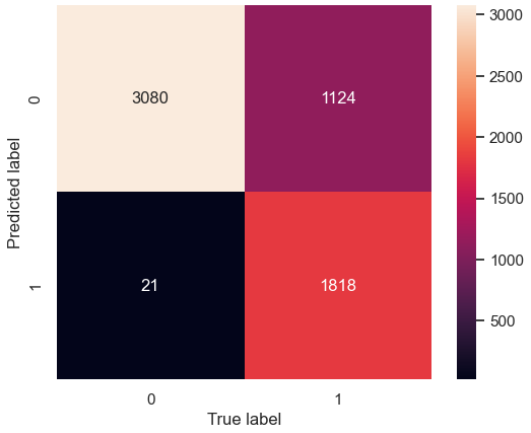
Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Network (RNN) used to address the issues of gradient vanishing and exploding in traditional RNN. LSTM consists of a "forget gate", "input gate", and "output gate" which control the flow and retention of information within LSTM. LSTM can handle variable-length sequence data and has adaptive feature extraction capabilities and does not require manual feature engineering. Meanwhile, It requires a long training time and a large amount of labeled data and computational resources.

5 Experimental Results

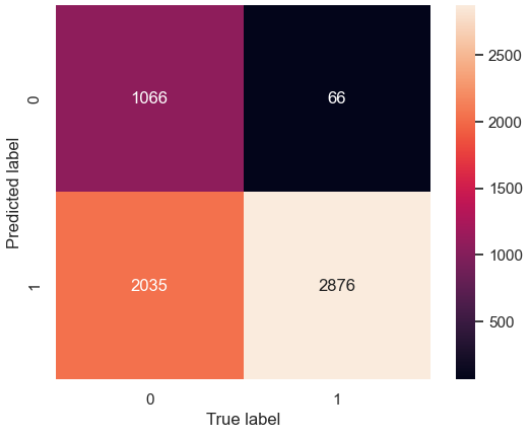
All five models were tested against the test dataset and the test results were presented in confusion matrices, while the model performance statistics were calculated.The confusion matrices were presented in pictures, while model performance statistics were presented in tables.

The Naïve Bayes seemed to perform better than the kNN models, with the accuracy of 0.81 compared to 0.65 and 0.76. Meanwhile theNaïve Bayes model scored higher than the kNN models in all three other categories. It should also be noted that theNaïve Bayes model also took less time to complete training than the kNN models.

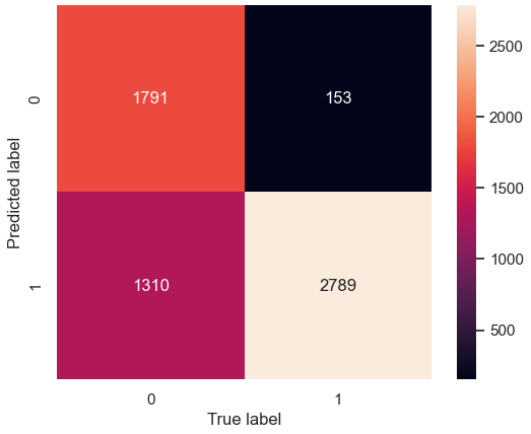
In comparison, the CNN model had almost all scores of 0.97, slightly higher than the LSTM model of 0.95. Neither of these models had complete data, so the CNN and LSTM models probably performed better than the plain Bayesian and kNN models. However, the LSTM model consumed far more computational resources than several other models.



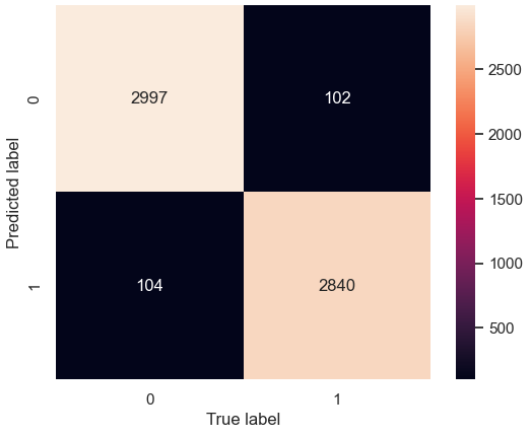
(a) Naïve Bayes model



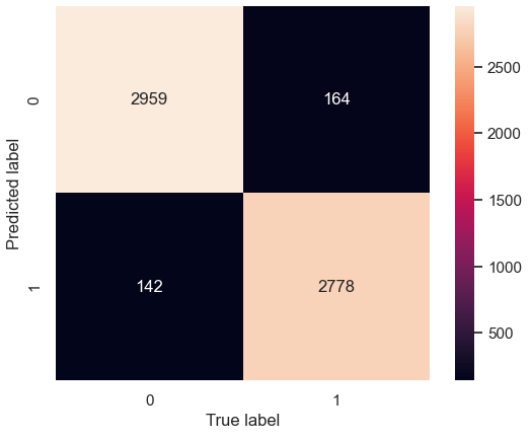
(b) 3NN model



(c) 7NN model



(d) CNN model



(e) LSTM model

Figure 1: Confusion Matrices

Label	Accuracy	F1-score	Precision	Recall
Total	0.81	0.80	0.86	0.81
Reliable		0.73	0.99	0.84
Unreliable		0.99	0.62	0.76

Table 2: Bayes model Confusion Matrix

Label	Accuracy	F1-score	Precision	Recall
Total	0.65	0.62	0.76	0.66
Reliable		0.94	0.34	0.50
Unreliable		0.59	0.98	0.73

Table 3: 3NN model Confusion Matrix

Label	Accuracy	F1-score	Precision	Recall
Total	0.76	0.75	0.80	0.76
Reliable		0.92	0.58	0.71
Unreliable		0.68	0.95	0.79

Table 4: 7NN model Confusion Matrix

Label	Accuracy	F1-score	Precision	Recall
Total	0.97	0.97	0.97	0.97
Reliable		0.97	0.97	0.97
Unreliable		0.96	0.97	0.97

Table 5: CNN model Confusion Matrix

Label	Accuracy	F1-score	Precision	Recall
Total	0.95	0.95	0.95	0.95
Reliable		0.95	0.95	0.95
Unreliable		0.95	0.94	0.95

Table 6: LSTM model Confusion Matrix

6 Discussion

Given the time and computational power, it was not possible to incorporate all the data into the model training, nor was it possible to fine-tune the hyperparameters to obtain a better model. Nevertheless, we can see from the results that CNN and LSTM perform better. At the same

time, we also needed to be aware that neural network models required a large amount of labelled data and we were only able to provide a small amount of data, therefore, both models were likely to suffer from overfitting problems. This is subject to future enhancement.

In the process of training the models, kNN and LSTM took significantly more time than the other two models. Considering the time factor, the Bayes and CNN models can be given priority, which needs to be combined with the actual situation of the data to make the final decision.

For the final question about the function, a more logical approach would be to use the input function to ask the user to enter the news text and the model name. However, to make it easier to present the results of the function, we will provide the news text and the model name directly for the time being, while the actual process can be implemented using the input function.