NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4045

Natural Language Processing

Project

Group 10

BARABASEV DANIIL (N2202450G)

CHRISTOPHER ARIF SETIADHARMA (U2022400H)

KOH YI JOSHUA (U1922368F)

RYAN PHUAY QIAN HAO (U1922240H)

TRAN HIEN VAN (U1920891J)

Table of Contents

1. Abstract

This assignment aims to build a sentiment analysis application based on several Natural Language Processing (NLP) tasks. Our group decided to crawl financial news data on Reddit and make our sentiment analysis application by analysing the financial news headlines. This report will highlight our findings and the results we obtained.

Link for your video introduction: https://youtu.be/sl0M218O-go

2. Introduction

With the amount of data available to digest, it's sometimes hard to quickly assess the impact of real word problems. We decided to make this task more accessible in the field of finances. By gathering data from several financial sources and combining it with modern NLP approaches, we want to automate and speed up it. We will use various NLP libraries, such as NLTK and pre-trained language models.

By developing a simple-to-use UI, we were able to simplify the process in which users can gather the general financial sentiment from financial news sources. It is important to determine the sentiment of financial news for users to make sound decisions for their personal finances. Manually reading and assessing the sentiment of every news article to come out is not only unfeasible due to the sheer volume of financial news generated per day, but also due to user errors like fatigue, bias, or personal emotions. Hence, sentiment analysis is used to overcome the shortfalls in humans and generate the data required by users to make sound financial decisions.

2.1 Links

The file with our dataset and results:
https://entuedu-my.sharepoint.com/:u:/g/personal/n2202450g_e_ntu_edu_sg/ETgQj5ME0sZAlatVy8yoA_kB_7OlfR6K5waYtAOZt4eMug?e=YWradQ

Our source code and models:
https://entuedu-my.sharepoint.com/:u:/g/personal/n2202450g_e_ntu_edu_sg/EUOQonqHsF9Pl8gSjEqdUv8BAIdMHclbWm4UKjTpTv5wJg?e=hXp4TQ

3. Crawling

Our dataset initially contained 28,599 various records and 378,292 words, ranging from the 1st of January 2019 to the 14th of December 2019. The primary sources for our data were financial pages on Reddit. The subreddits we chose to crawl our data from are ‘FinanceNews’, ‘Economics’, ‘SecurityAnalysis’, ‘finance’, ‘business’, ‘econmonitor’.

We reasoned our decision to crawl Reddit was the convenience of it. Reddit pages are moderated and filtered for spam or bots (but we still had to perform filtering as we found out it is far from perfect). The second reason for Reddit was the ease of use of their API. With the Python library “psaw”, which uses the Reddit API for downloading data, we gathered a large amount of data from several news outlets (Bloomberg, The New York Times, The Reuters, and others).

3.1. Data filtering

Unfortunately, Reddit is rife with spam posts and bots. Despite Reddit performing its content moderation, we still had to clean the data to ensure that it was usable for our purposes. To achieve that, we implemented the following techniques:

1. *Removing invalid or duplicate entries:* the first step was to reduce the size by removing duplicate entries that lead to one source.

2. *Removing entries containing emojis*: from observing our dataset, we deduced that posts containing emojis were mainly spam or non-serious posts, so we decided to delete them.

3. Removing non-English titles: several articles linked weren’t in English by detecting them with the Python library called *langdetect* and deleting them. This library is easy to deploy and is quick enough for us to use.

After cleaning the data, we are left with a final dataset of 10,509 data points with the following attributes:

- id
- title
- score
- external_url
- author
- Submitted_time

3.2 Data statistics

Amongst the 10,509 data points, the most common words to appear overall are shown in the WordCloud below (figure x). As global superpowers, it is expected that the frequency in which China and the US will appear is very high, and this is a trend that we observed among the most common words in each sentiment. Sentiments among the data points mainly leaned toward the neutral and negative sides, with fewer positive titles.

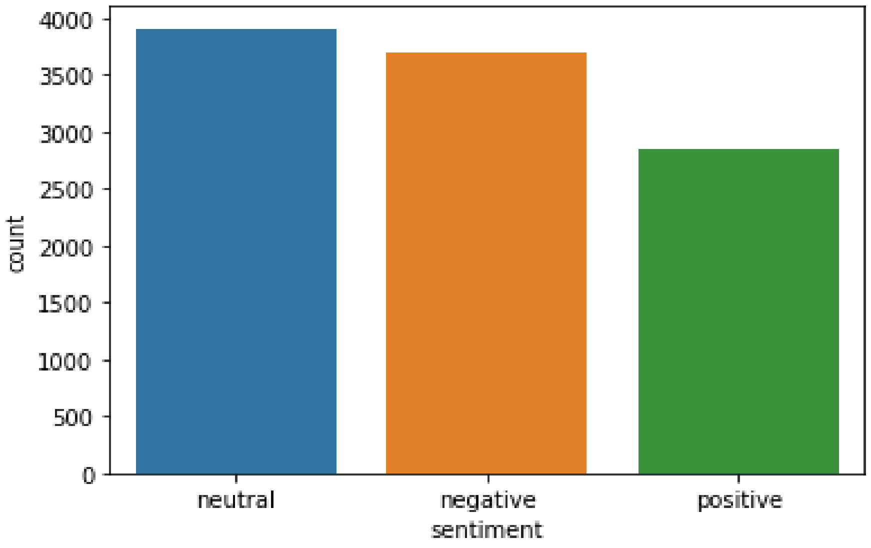


Figure 1 Sentiment distribution

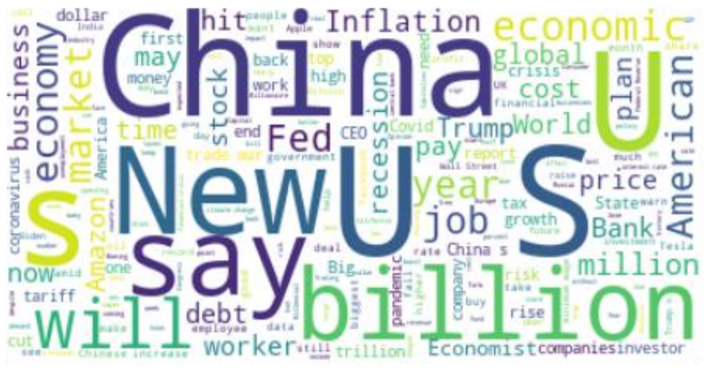


Figure 2 The most common words in our dataset

The most common words for the titles labelled “neutral” are very similar to the complete set of all data points.

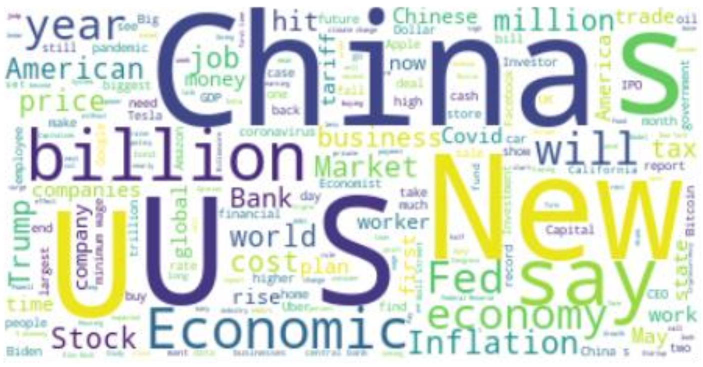


Figure 3 The most common words in "neutral" category

For titles that are labelled as "positive", positive keywords like "growth", "top", and "increase" are much more common than in titles that are labelled as "neutral".

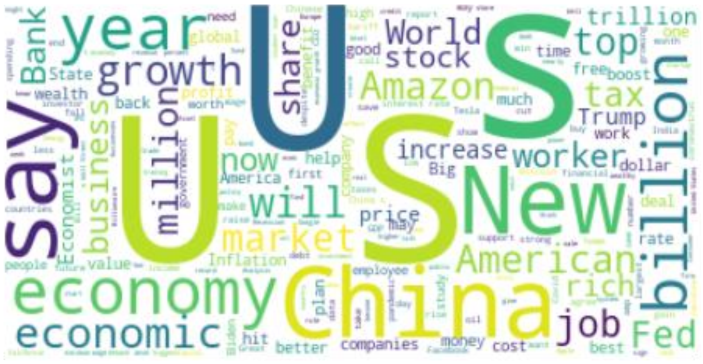


Figure 4 The most common words in "positive" category

For titles that are labelled as "negative", negative keywords like "inflation", "tariff", "debt", "Inflation" and "risk" are much more common than in titles that are labelled as "neutral".

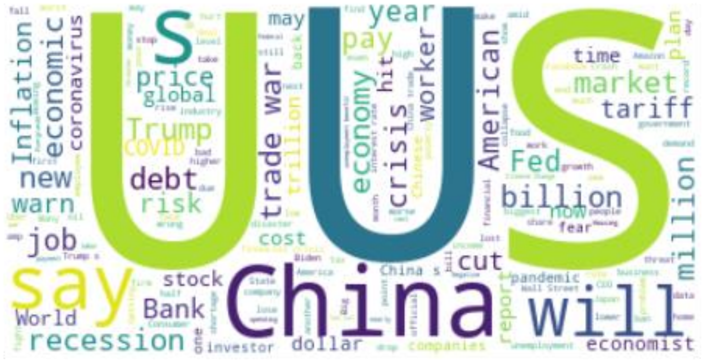


Figure 5 The most common words in the "negative" category

3.3 Labelling

For our project, we did a 20/80 split, where 20% of the data were labelled by hand for negative, neutral, or positive sentiment in the headline. The labelling was managed by two people consulting the rest of the group. Despite that, there were occasions when we were unable to agree on a label. We ran into several decisions we had to make to have consistent sentiment throughout all the headlines. One could argue about the correctness of our analysis as the opinions on the social policies could vary (social policies vs policies that help grow companies or support for global market vs trade wars with other superpowers). We labelled our dataset to the best of our knowledge, and we acknowledge that this could impact our results.

3.3.1 Auto labelling with VADER

For the 80% of data that we did not label by hand, we used a sentiment analysis model known as VADER Sentiment Analysis to label the data. VADER follows a set of rules to mathematically determine and quantify the emotions behind a statement, so there is no need for large sets of training data. VADER's resource-efficient approach helps us decode and quantify the emotions in streaming media such as text, audio, or video. We can use the valence score generated by VADER to determine if the headline has a negative, neutral or positive sentiment.

3.4 Other datasets

As part of the training described further in the report, we used a pre-labelled dataset from Financial Phrase Bank that contains the sentiments for financial news headlines from a retail investor's perspective. We used this dataset mainly for benchmarking models to compare it later with our dataset or enhance the accuracy of models. The use of this dataset is always mentioned.

3.5 Data notations

Used data in all the experiments:

- Financial Phrase Bank dataset: 4837 samples (a.k.a Kaggle dataset)
- Self-labelled crawled Reddit dataset: 2047 samples (a.k.a Reddit dataset)
- Remaining crawled Reddit dataset: 8409 (a.k.a serving dataset)
- Combination of Financial Phrase Bank dataset and Reddit dataset (a.k.a combined dataset)

4. Classification

For all models in this section, we considered four training scenarios as follows:

- Training only on the Kaggle dataset (model 1)
- Training only on the Reddit dataset (model 2)
- Transfer learning from Kaggle to Reddit dataset (model 3)
  Transfer Learning is a machine learning technique where we use a pre-trained model as the starting point for a model on a new domain task. A model pre-trained on one task is transferred to a new but related task as an optimisation that enables significant progress when modelling the domain task. If a model is trained on a large and general enough corpus of data, this model will allow us to take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. In this project, we conducted transfer learning from the Kaggle dataset, which consists of benchmark financial headlines dataset, to our crawled Reddit dataset.
- Training on Combined dataset (model 4)
  We combined Reddit and Kaggle datasets, shuffled the data, and then used it to train models. We adopt this method to increase the data amount and enable the model to jointly learn the features from these two datasets, which may lead to better generalisation capability.

We split the dataset into train/validation/test datasets with a ratio of 80/10/10 with train and validation data to train and tune model hyperparameters while using the test dataset to pick the optimal model. As the test dataset is the data the model didn't see before, the test set's metric will reflect how well the model generalises to predict unseen data. It is important to note that serving data should not contain sentiment labels. However, we want to compare the difference between our model and another existing model, Vader, which we use to get sentiments for this dataset. We don't use the serving dataset (with Vader's labels) as the test data for choosing the model because the distributions of Reddit/Kaggle and Serving datasets are very different due to the labelling method. It is fair to compare Reddit data to get the best model.

The only candidates for the optimal model are models 2, 3, and 4, as they are trained with the domain task for Reddit data. The first model is only for transfer learning of model 3 and the reference. We keep this reference to monitor our labelling method for Reddit data because our labels may be affected by annotators' financial knowledge. In contrast, Kaggle data has a well-established labelling method from research groups. Although the test data between models (2,3) and model 4 are slightly different in the amount of data, it is still reasonable to use these test metrics to compare these model performances. The label distributions are similar, and the Kaggle dataset is a benchmark dataset with a valid labelling method.

4.1. CNN

4.1.1. Overview

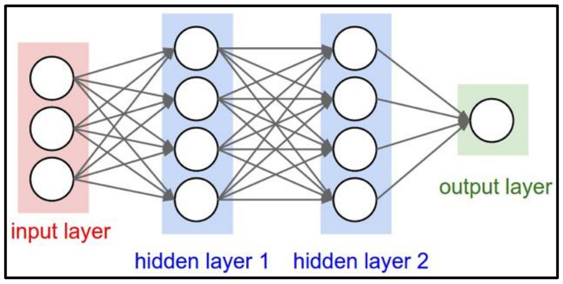


Figure 6 A regular 3-layer Neutral Network [4]

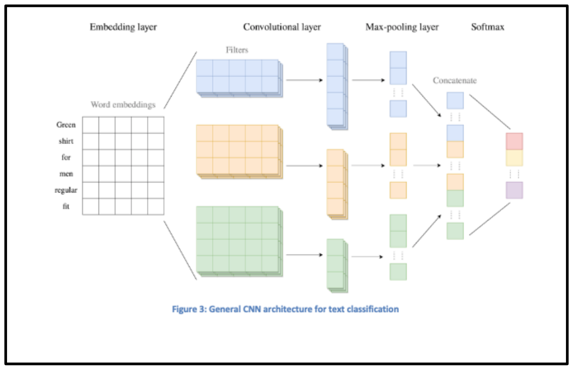


Figure 7 A ConvNet arranges its neurons in 1 dimension (unigram, bigram, etc.) [5]

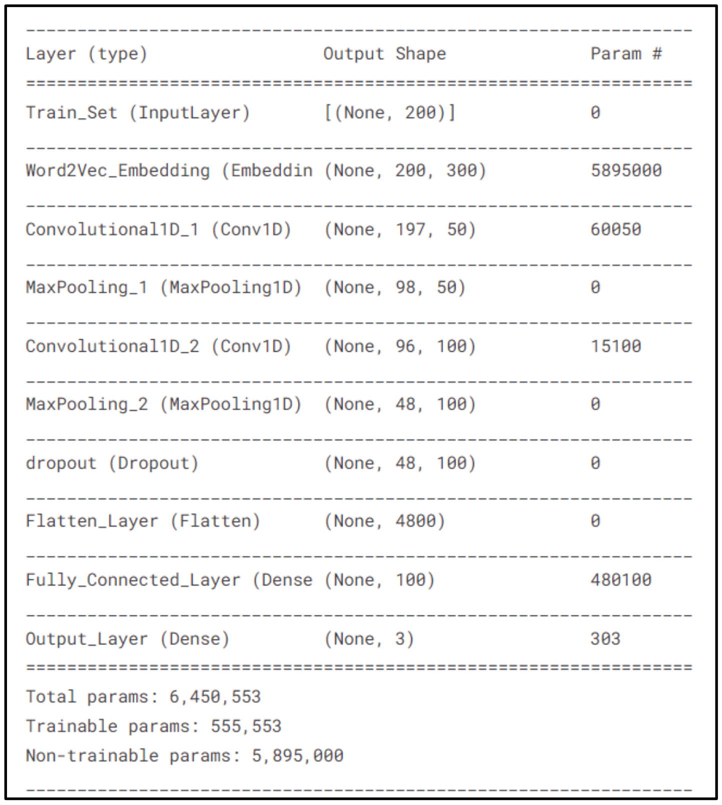
Convolutional neural networks are widely used deep learning architectures in visual imagery recognition. CNN is often used in cases like image classification, facial recognition, or object detection. It was just a question of time when the CNN would be applied to other tasks such as sentence classification, sentiment analysis, text summarisation, and others.

The main idea of sliding or convolving a predetermined data window is the central idea of CNN. The layers that consist of neurons that scan their input for patterns are called convolutional layers. It produces local connections, where input regions are connected to different neurons. The second type of layer is the so-called downsampling layer or pooling layer, which reduces the dimension of the input. As images can be represented by their pixel values, we can encode the text as an array of vectors. Each vector corresponds to one token, typically a word, but it could be a character. The idea is the same as in image processing, but instead, we work with sequential data, which is just a one-dimensional convolution.

4.1.2 Experiment

For the CNN model, we incorporate GoogleNews-vectors-negative300 word embedding into our model architecture. Word embedding is a learned representation that allows words with similar meanings to have a similar representation. The sequence model makes use of word embedding to convert text data into a densely distributed vector for each word. Three main techniques to learn a word representation are Embedding Layer, Word2Vec, and Glove. In our experiment, we first obtained a 300-dimensional pre-trained word2vec from Google and used it as our initial Embedding layer weight to train jointly with other layers on our domain tasks. As GoogleNews-vectors-negative300 is pre-trained on the part of the Google News dataset (3 million words and phrases) in the News domain (similar to our Financial Headlines domain), the embedding can capture the relationship of words frequently used in our context.

The model architecture is summarised below. We designed two layers of a 1D convolutional net followed by a max pooling layer to downsample the feature map. We also add a dropout layer with the probability of 0.2 to avoid overfitting.

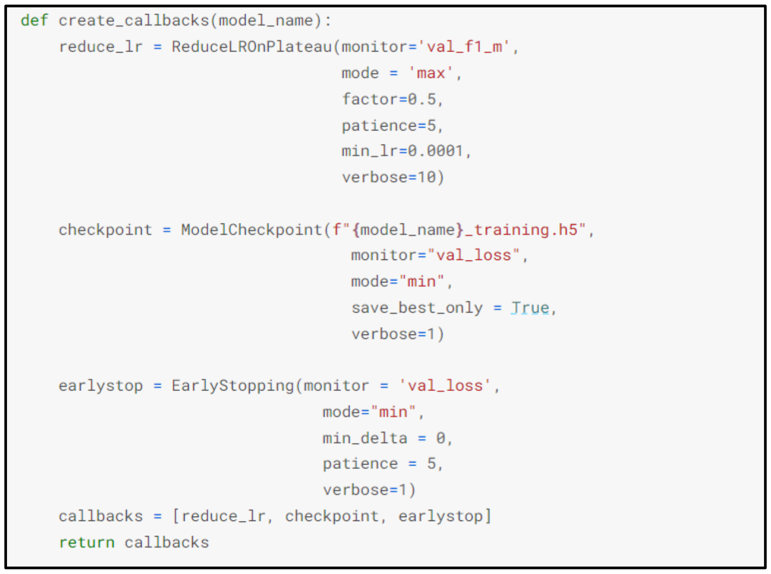


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Train_Set (InputLayer)       [(None, 200)]             0
_________________________________________________________________
Word2Vec_Embedding (Embeddin (None, 200, 300)          5895000
_________________________________________________________________
Convolutional1D_1 (Conv1D)   (None, 197, 50)           60050
_________________________________________________________________
MaxPooling_1 (MaxPooling1D)  (None, 98, 50)            0
_________________________________________________________________
Convolutional1D_2 (Conv1D)   (None, 96, 100)           15100
_________________________________________________________________
MaxPooling_2 (MaxPooling1D)  (None, 48, 100)           0
_________________________________________________________________
dropout (Dropout)            (None, 48, 100)           0
_________________________________________________________________
Flatten_Layer (Flatten)      (None, 4800)              0
_________________________________________________________________
Fully_Connected_Layer (Dense (None, 100)               480100
_________________________________________________________________
Output_Layer (Dense)         (None, 3)                 303
=================================================================
Total params: 6,450,553
Trainable params: 555,553
Non-trainable params: 5,895,000
_________________________________________________________________
```

Figure 8 The CNN model architecture summarised

We didn’t do further preprocessing for all four training scenarios except tokenising word to sequence in this experiment. We have conducted a dry run of preprocessed data (with stemming and lemmatising data), but the model performance is significantly lower compared to the current version. It should be due to the short length of text sentences (headlines).

During training, there are three Keras callbacks to monitor the hyperparameter tuning process and save the checkpoint: EarlyStopping and ReduceLROnPlateau, and ModelCheckpoint. EarlyStopping callback will stop the training when the validation loss is not increased as it has coverage and the model started overfitting. ReduceLROnPlateau helps to reduce the learning rate to maximise the validation f1 score. It is proved that a larger learning rate could cause the loss to oscillate around its minimum. Such callbacks prevent learning from stagnation without overfitting, which helps to achieve higher accuracy.



```
def create_callbacks(model_name):
    reduce_lr = ReduceLROnPlateau(monitor='val_f1_m',
                                  mode = 'max',
                                  factor=0.5,
                                  patience=5,
                                  min_lr=0.0001,
                                  verbose=10)

    checkpoint = ModelCheckpoint(f"{model_name}_training.h5",
                                 monitor="val_loss",
                                 mode="min",
                                 save_best_only = True,
                                 verbose=1)

    earlystop = EarlyStopping(monitor = 'val_loss',
                              mode="min",
                              min_delta = 0,
                              patience = 5,
                              verbose=1)
    callbacks = [reduce_lr, checkpoint, earlystop]
    return callbacks
```

Figure 9 The CNN model callback method

4.2 Long Short-Term Memory (LSTM)

4.2.1 Overview

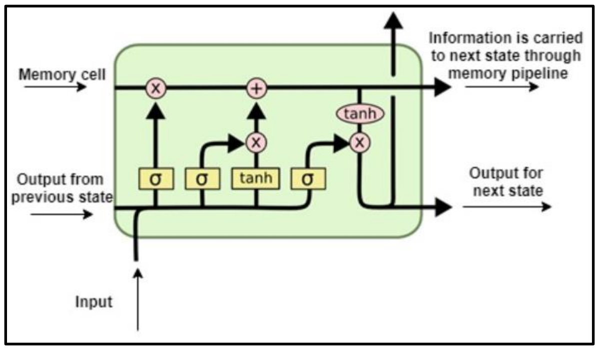

Memory cell
Information is carried to next state through memory pipeline
tanh
Output from previous state
σ
σ
tanh
σ
Output for next state
Input

Figure 10 The LSTM architecture [6]

Long Short-Term Memory networks, which are usually just called LSTM, are a type of recurrent neural network but are more than traditional recurrent networks in terms of memory. These networks are built upon RNN to learn long-term dependencies. The core idea behind LSTMs is the cell state, acting as a conveyor belt for information flow from the current state to another state. The LSTM use gates to remove or add information to the cell state.

RNNs are vulnerable to the vanishing gradients problem, which makes learning long data sequences difficult. The gradients contain information utilised in the RNN parameter update during training. However, as we go deeper into the network, the gradient becomes smaller and may vanish. LSTM can solve this vanishing gradient by using forgot gate to determine which information in the cell state should be forgotten when processing new data.

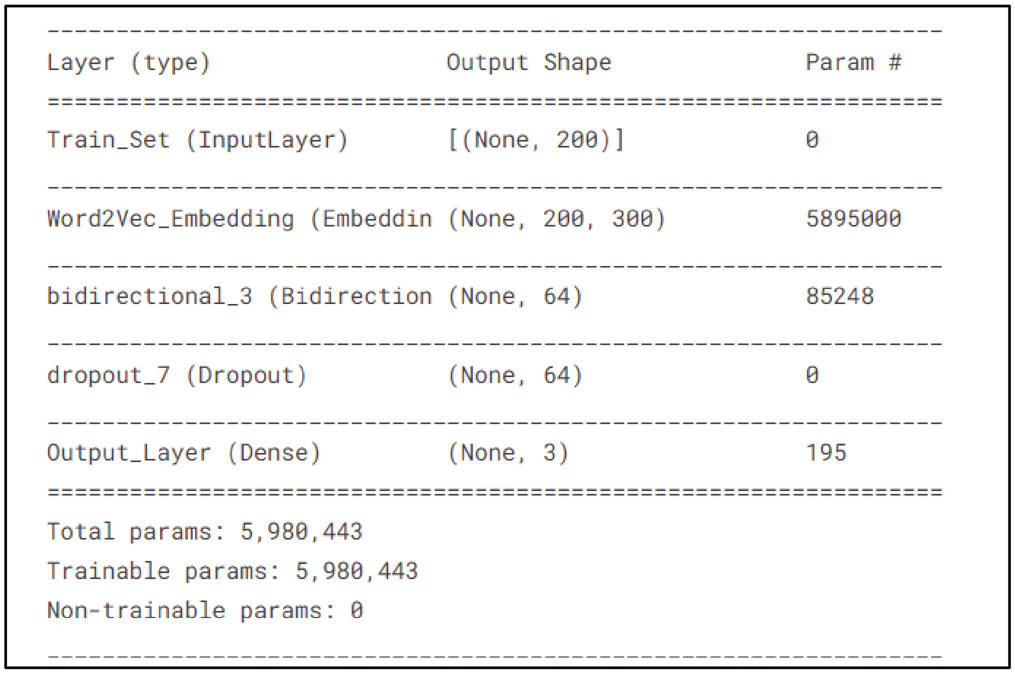
The advantage of using LSTM in natural processing tasks is that we can use a sequence and understand its meaning of it instead of just predicting the sentence class based on statistics. That means that compared to a classical approach of non-neural network classifiers, where each word is classified into one of the categories, here we assign the classification to a meaning. With appropriate layers for embedding and encoding, it has the potential to be very accurate with class prediction.

4.2.2 Experiment

Similarly to CNN, we also add an Embedding layer to learn word representation from our domain. In this experiment, we didn’t opt for the use of GoogleNews vectors to examine the effect of embedding layers in modelling sentiment analysis tasks.

The model makes use of a Bi-LSTM layer to obtain context information. The Bi-LSTM consists of two LSTMs: one taking the input in a forward direction and the other in a backward direction so it is capable of learning long-term bidirectional dependencies between time steps of sequence data. As usual, we also adopt Dropout with 0.2 probability to avoid overfitting.

The model architecture is as follows:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Train_Set (InputLayer)       [(None, 200)]             0

_________________________________________________________________
Word2Vec_Embedding (Embeddin (None, 200, 300)          5895000

_________________________________________________________________
bidirectional_3 (Bidirection (None, 64)                85248

_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0

_________________________________________________________________
Output_Layer (Dense)         (None, 3)                 195
=================================================================
Total params: 5,980,443
Trainable params: 5,980,443
Non-trainable params: 0
_________________________________________________________________
```

Figure 11 The LSTM model summarised

Similar to the previous experiment, we didn't preprocess the data for the same reason. In addition, we employed the three callbacks mentioned above to monitor the model's hyperparameter tuning and training process.

4.3. FinBERT

4.3.1 Overview

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language model developed by Google in 2019, and instead of pre-learning words, this model has pre-learned context. That allows us to finetune this model with just one additional output layer. BERT is used in a variety of tasks, such as question-answering and language inference.

The main limitations of standard language models are that they are unidirectional and limit the architecture, where tokens can only refer to previous tokens in the self-attention layers. Instead, BERT accomplishes fusing the left and right context, and that allows it to pre-train the bidirectional Transformer. With this, BERT can overcome the problem of defining prediction goals. Traditionally models predict the next word in a sequence, a directional approach that limits learning. BERT implements two main training strategies:

- *Masked LM*: 15 % of the words in the sequence are replaced with [MASK] tokens before training. After the model tries to predict the original words of the masked tokens, the loss function accounts for masked values and ignores the prediction of the non-masked values. However, this results in slower convergence.
- *Next Sentence prediction*: in this method, BERT pairs sentences in a way that 50 % of sentences are subsequent sentences from the corpus and the other 50 % are not. Then in the training process, the model has to distinguish between the two sentences in training.

When training the Bert model, Masked LM and Next Sentence prediction are trained together, and the goal is to minimise the combined loss function of the two strategies.

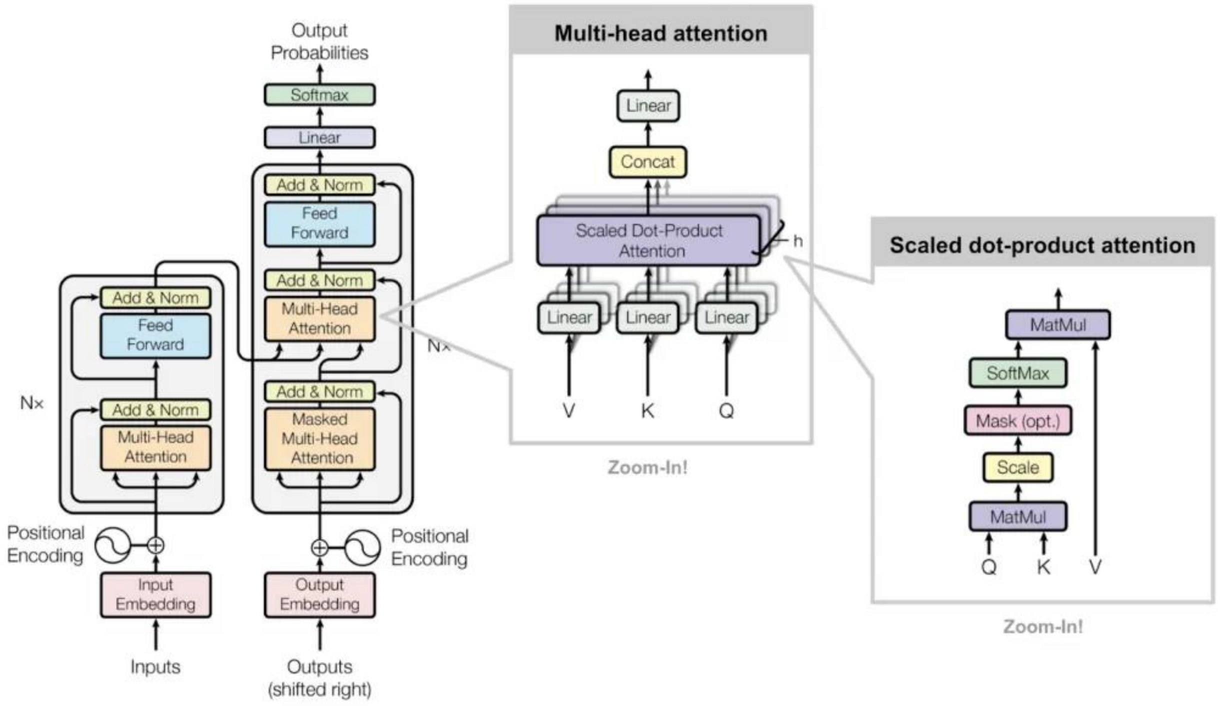


Figure 12 BERT architecture

FinBERT is fine-tuned on two datasets. The language is trained on a subset of the Reuters TRC2 dataset, and for the sentiment analysis is used Financial PhraseBank dataset.

BERT comes with pre-trained tokenisation and preprocessing. The most notable cases for our case would be:

- *Stemming or lemmatisation*: BERT uses byte pair encoding that encodes words like **standing** like **stand** + **##ind** because this information is needed for some NLP tasks. Therefore there is no need for us to do it by ourselves.
- *Decapitalisation*: in our case, we need to know the difference between **US** and **us**, so decapitalisation should not be performed. Fortunately, BERT provides cased and uncased models.
- *Removing stop words and high-frequency words*: BERT uses the Transformers model, and it does not affect the result.

In our testing, we use EarlyStopping callback to stop model training if the validation loss didn't decrease to avoid overfitting.

4.4 Results & Evaluation

4.4.1 Results

We evaluate the model performance by using the f1 score. As our data distribution is not balanced between neutral and (positive and negative), it will not be very objective if we use accuracy as the primary metric to evaluate. Below is the metric for the test dataset (10% of the Reddit dataset):

Training dataset	**Kaggle (model 1)**		**Reddit self-labeled (model 2)**		**Kaggle → Reddit self-labeled (Transfer Learning) (model 3)**		**Kaggle + Reddit self-labeled (Combined) (model 4)**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc
CNN	0.77	0.75	0.55	0.57	0.62	0.62	0.78	0.78
LSTM	0.64	0.68	0.39	0.52	0.52	0.56	0.71	0.72
FinBERT	0.89	0.89	0.69	0.72	0.75	0.79	0.85	0.87

Table 1 FinBERT results on datasets

The results of best models (model 4) on the serving dataset (8000 data labelled by Vader):

Model pre-trained on Combined data	Serving dataset	
	F1	Accuracy
CNN	0.36	0.36
LSTM	0.36	0.36
FinBERT	0.44	0.48

Table 2 Overall results of tested models

4.4.2 Evaluations

For all CNN, LSTM, and FinBERT, model 3 has better performance than model 2 on the same dataset, and model 4 has outperformed model 3 with a similar distribution dataset. We can conclude that:

- Transfer learning from a pre-trained model on a larger dataset (Kaggle) can increase performance.
- More data helps the model to learn more information and increase the metric.

Compared between CNN and LSTM, surprisingly, CNN performs slightly better than the latter. This result may be due to our simple LSTM architecture, and the embedding layer we used for LSTM does not contain GoogleNews vector weights. We design such an LSTM architecture to examine the advantage of CNN and pre-trained word2vec in the context of sequence data like text. In the next section, we will improve our LSTM model to improve performance further.

As expected, FinBERT gives the best performance among the three models. As BERT utilises the attention mechanism from Transformers architecture, the model can learn features from our data and generalise well in our test set. In addition, we also make use of a pre-trained model which is pre-trained on the finance domain, therefore, the performance outperforms other models.

Overall, our models' performances are affected by the data imbalance. Our data distribution has excessive Neutral sentiments compared to Negative or Positive. Therefore, our model predictions usually fall in Neutral sentiment. When we apply the model to the serving dataset with Vader's labels, our metrics are around 40%, as indicated in table 2. We manually random check the serving dataset to see why the results are low.

Examples of serving dataset samples

In the next three tables, we will highlight the results of FinBERT labelling. The following table shows examples of correct labelling. The headlines that contain strong sentiment words such as “rising inflation” or “unemployment” were usually labelled correctly.

Headline	Assigned label	Prediction
There Is Shadow Inflation Taking Place All Around Us	Negative	Negative
When Environmental Regulations Backfire	Negative	Negative
September jobs report: Economy added just 194,000 jobs, unemployment rate falls to 4.8 percent	Positive	Positive
NatWest faces fine over money laundering failings	Positive	Positive
How Stock Price Trends Are Driven Primarily By Institutional Operators Who Manipulate Stock Prices	Negative	Negative

Table 3 FinBERT examples of correct labelling

Next are examples of headlines that were classified wrongly.

Headline	Assigned label	Prediction
EU and Mercosur group agree draft free-trade deal	Neutral	Negative
Job opening rate at an all time high, but hiring rate has been flat below 2005-6 levels over the past two years	Negative	Positive
A Boston startup developing a nuclear fusion reactor just got a roughly $50 million boost	Neutral	Negative
Full-time minimum wage workers cannot afford a 2-bedroom rental anywhere in the US	Negative	Positive
A Wealth Tax on the Rich Won’t End Deficits	Neutral	Positive

Table 4 FinBERT examples of wrong labelling

The last example is a table of headlines that the Vader does not correctly label but were correctly labelled by the model.

Headline	Assigned label	Prediction
Trump suggests U.S. should start manipulating the dollar — contradicting U.S. policy	Positive	Negative
Model that predicted 2008 financial crisis suggests another recession is coming: expert	Positive	Negative
Trump has reportedly tasked aides to find a way to weaken the US dollar	Positive	Negative
White House projects $1 trillion deficit for 2019	Positive	Negative
Israeli Tech Companies Raised $2.32 Billion in Q2 2019, Report Says	Negative	Neutral
Microsoft outperforms analysts’ revenue predictions with a ‘blow out’ Q4	Negative	Positive
On the Other Hand by Howard Marks	Neutral	Negative

Table 5 FinBERT examples of the wrong label but a correct prediction

As expected, our model usually mislabeled Positive and Negative sentiments due to the shortage of data for such sentiments. However, as the serving dataset is semi-supervised labelled, the annotations are just for reference.

To conclude, for serving data, we will consider FinBERT model 4 as our optimal model to predict. We still report all models' predictions as references.

5 Innovations

5.1 Stacking CNN-LSTM architecture

We explored adding a convolution layer followed by max pooling before the LSTM layer. We will refer to this model as CNN-LSTM.

For the first variation of this CNN-LSTM, we use trainable embedding. This proves to have slightly better results than the typical architecture in Section 4.2.2, as seen in the table below.

We also explored a second variation, which uses the GoogleNews-vectors-negative300 word embedding like our CNN model in Section 4.1.2, and we call this CNN-LSTMv2. This model had better performance than CNN-LSTMv1, however, it is still unable to perform better than the CNN model.

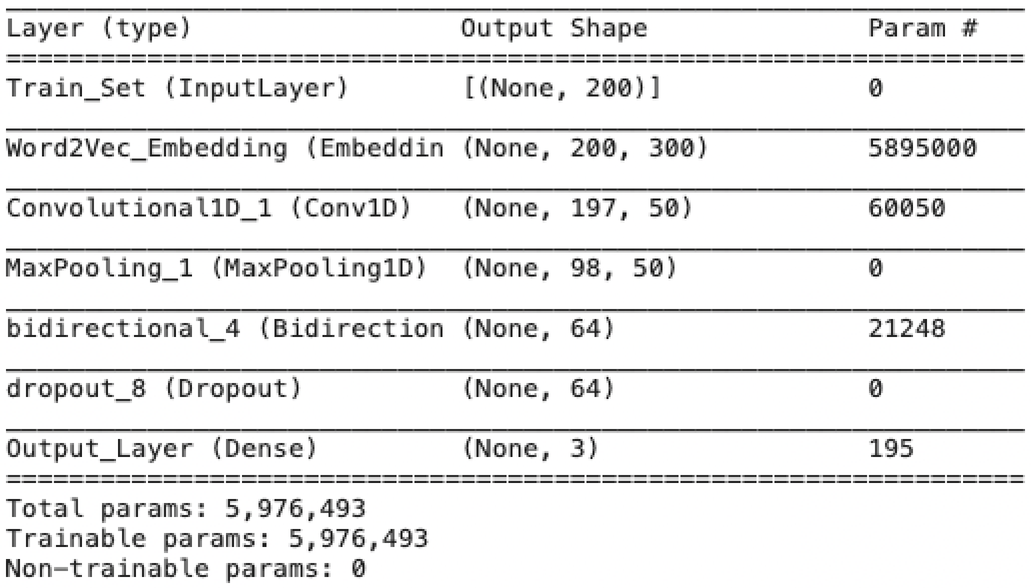

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Train_Set (InputLayer)       [(None, 200)]             0
_________________________________________________________________
Word2Vec_Embedding (Embeddin (None, 200, 300)          5895000
_________________________________________________________________
Convolutional1D_1 (Conv1D)   (None, 197, 50)           60050
_________________________________________________________________
MaxPooling_1 (MaxPooling1D)  (None, 98, 50)            0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 64)                21248
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0
_________________________________________________________________
Output_Layer (Dense)         (None, 3)                 195
=================================================================
Total params: 5,976,493
Trainable params: 5,976,493
Non-trainable params: 0
```

Figure 13 The summary of CNN-LSTM model

5.2 Ensemble learning method

5.2.1 Averaging Voting Classifier

A Voting Classifier is a machine learning model that combines similar or conceptually different machine learning models and predicts based on the highest majority of voting. The strategy is to create one model that trains all of these models and predicts output by using their combined majority of votes for each output category instead of creating separate dedicated models and finding their accuracy.

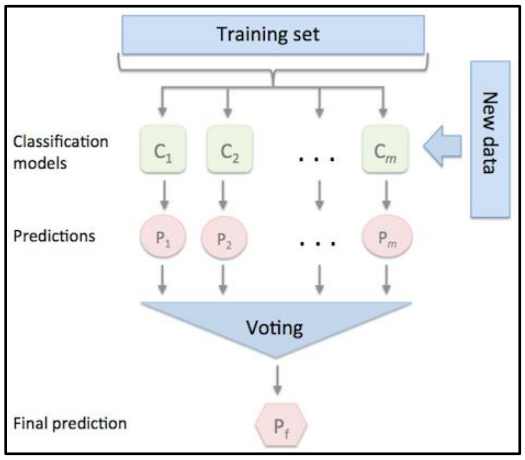


Figure 14 Model classifier [7]

In our experiments, the ensemble model consists of 3 models: RandomForestClassifier, SVC and LogisticRegression, with some hyper-parameters tuning to enhance the accuracy:

- *Random Forest* is a classification model that builds decision trees on different samples and takes their majority vote for classification.
- *C-Support Vector Classifier* is a supervised learning classification model using vectorised space to classify.
- *Logistic model* is a statistical model that monitors the probability of an event taking place. This can be used to classify with a certain degree of precision.

We employ soft voting, which means the final label is predicted by averaging the class probabilities. Here, we also adopted the four mentioned training methods and split the dataset into train/test datasets with a ratio of 9:1. The results for the test set are presented in Table 6.

5.2.2 Weighted Ensemble by using Deep Neural Network (DNN)

We make use of our models in the Classification task, which are: CNN, LSTM, and FinBERT. Instead of jointly training as 5.2.1, we will separately train each model (Classification task) and then use the trained model to output the predictions. After that,

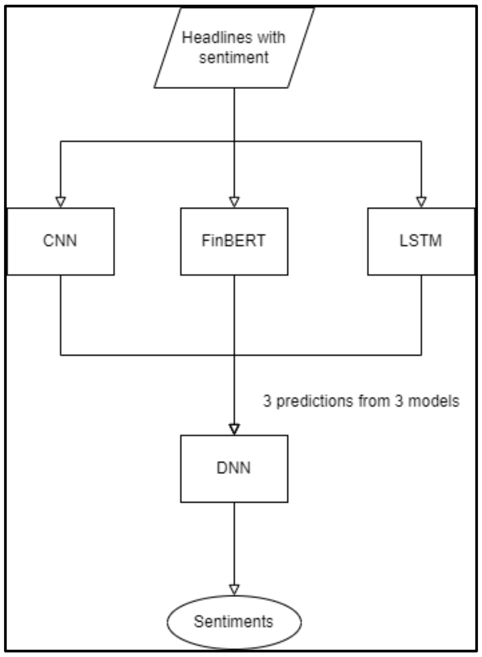
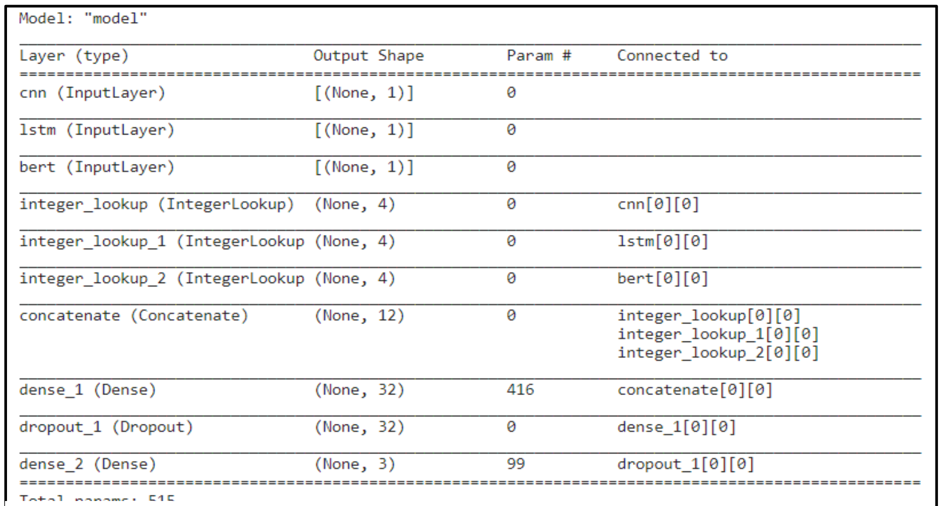


Figure 15 The workflow of the Ensemble algorithm

we will use these model predictions as input for a DNN, and the output will be the final labels of the data. Below is the workflow of the algorithm:

The DNN model we design is a shallow network with only 1 Dense layer followed by a Dropout layer, not to mention the input and output layers. It is trained to find the suitable weight for each model prediction when we combine them to generate the correct label. It

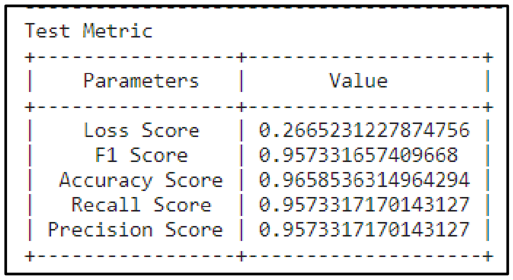


```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
cnn (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
lstm (InputLayer)               [(None, 1)]          0
__________________________________________________________________________________________________
bert (InputLayer)               [(None, 1)]          0
__________________________________________________________________________________________________
integer_lookup (IntegerLookup)  (None, 4)            0           cnn[0][0]
__________________________________________________________________________________________________
integer_lookup_1 (IntegerLookup (None, 4)            0           lstm[0][0]
__________________________________________________________________________________________________
integer_lookup_2 (IntegerLookup (None, 4)            0           bert[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 12)           0           integer_lookup[0][0]
                                                                 integer_lookup_1[0][0]
                                                                 integer_lookup_2[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 32)           416         concatenate[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 32)           0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 3)            99          dropout_1[0][0]
==================================================================================================
Total params: 515
```

Figure 16 The DNN model overview

can be considered a weighted ensemble method that assigns weights for each model output based on their importance or ability.

Here, we utilised three models, LSTM, CNN, and FinBERT, trained on Combined data to generate the prediction. For this experiment, the dataset we use to examine is only the Reddit dataset which consists of around 2000 headlines. We split this training dataset into train/test with a ratio of 9:1, as usual. We report two metrics, one for DNN and one for the whole system, to have a complete picture of the algorithm.



```
Test Metric
+-----------------+--------------------+
|    Parameters   |        Value       |
+-----------------+--------------------+
|    Loss Score   | 0.2665231227874756 |
|     F1 Score    | 0.957331657409668  |
|  Accuracy Score | 0.9658536314964294 |
|   Recall Score  | 0.9573317170143127 |
| Precision Score | 0.9573317170143127 |
+-----------------+--------------------+
```

Figure 17 DNN's metric on the test set

As we can see, the test metric for DNN is very high, achieving around 95% both accuracy and f1-score. However, we suspected that the DNN could only learn to find the optimal weights from 3 model predictions as inputs as plain numbers without actually learning the relationship of text or the three models themselves. We run the whole pipeline from end-to-end on the test set and report the test metric results in Table 17. Undoubtedly, the whole pipeline does not perform well as expected.

We propose another solution by randomly picking a combo of weights for 3 model predictions. The output is based on the new "probability" calculated from the sum of these 3 model predictions with corresponding weights. However, the final output is not desirable, and the whole pipeline predicts almost wrong for all the test samples.

5.3 Sarcasm Detection

We explored the use of sarcasm detection to improve our sentiment analysis performance. Sarcasm detection can be considered a subset of sentiment analysis [1]. We took a naive approach to this. After performing sentiment analysis and sarcasm detection separately, if the predicted sentiment is positive and sarcasm is also detected, the predicted sentiment would then be negative.

The sarcasm detection model is an LSTM trained on the NewsHeadline dataset, which contains half sarcasm from TheOnion and half non-sarcasm from HuffPost [2].

It is found in Table 6 that this naive approach did not improve the performance of the sentiment analysis. This is probably caused by our dataset containing very few sarcasm headlines compared to where the sarcasm detection model is trained. Therefore, the sentiment true positives become sentiment false negatives if sarcasm is detected.

Since our task is regarding news headlines, there is little to no sarcasm involved as they are from credible sources. This would not be the case for other kinds of datasets, such as Tweets where the authors would be more sarcastic. Therefore, sarcasm detection would be more powerful and helpful in specific cases of sentiment analysis.

Another possible approach for sarcasm detection is training a multitask classifier, where a single neural network is used to predict sentiment and sarcasm, resulting in improved results for both tasks [3]. The intuition is that both tasks are related. We often use sarcasm to express negative sentiments. However, we did not include this approach in our work because we did not label sarcasm in our dataset. Auto-labelling with a sarcasm detection model and using it for the multitask classifier would lead to the same problem as above, where the model for auto-labelling would be overly sensitive to sarcasm, and it would detect too many sarcasm false positives.

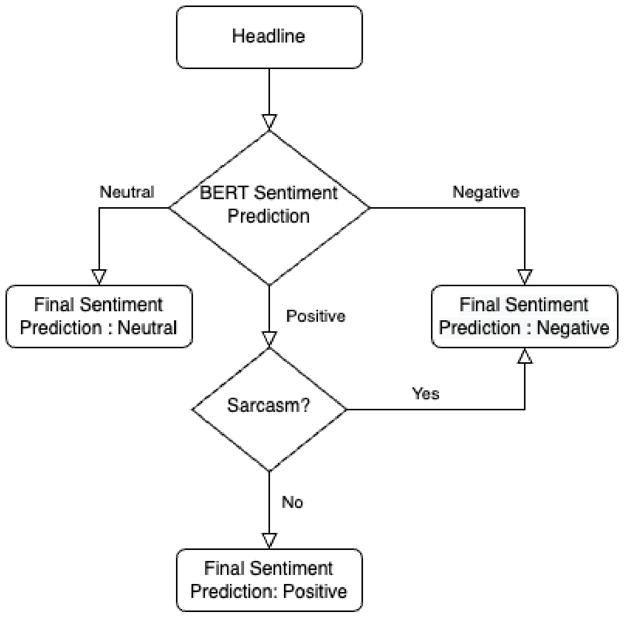

Headline
BERT Sentiment Prediction
Neutral
Negative
Final Sentiment Prediction : Neutral
Positive
Final Sentiment Prediction : Negative
Sarcasm?
Yes
No
Final Sentiment Prediction: Positive

Figure 18 Decision-making flowchart for our sarcasm detection

5.4 Results and Evaluation

Training dataset	**Kaggle (model 1)**		**Reddit self-labeled (model 2)**		**Kaggle → Reddit self-labeled (Transfer Learning) (model 3)**		**Kaggle + Reddit self-labeled (Combined) (model 4)**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc

CNN	0.77	0.75	0.55	0.57	0.62	0.62	0.78	0.78
LSTM	0.64	0.68	0.39	0.52	0.52	0.56	0.71	0.72
FinBERT	0.89	0.89	0.69	0.72	0.75	0.79	0.85	0.87
CNN-LSTM	0.68	0.70	0.48	0.60	0.46	0.51	0.70	0.70
CNN-LSTMv2	0.79	0.79	0.52	0.50	0.62	0.62	0.69	0.70
Averaging Voting Classifier	0.8	0.8	0.6	0.63	0.59	0.63	0.74	0.76
Weighted Ensemble using DNN	NA	NA	NA	NA	NA	NA	0.35	0.37
FinBERT + Sarcasm Detection	0.86	0.86	0.64	0.62	0.69	0.72	0.82	0.83

Table 6 Results of our innovations

Our model performance shows that the optimal model is still FinBERT after we propose several innovations. The ensemble methods fail to capture the underlying relationship between words, hence, the performances are lower than FinBERT, although they use several classifiers. However, the FinBERT and sarcasm Detection also fail to improve the performance. Our dataset is crawled mostly from high-reputation and reliable sources such as Reuters and WSJ, therefore, they may not contain much sarcasm as the authors would want to keep their papers as objective as possible.

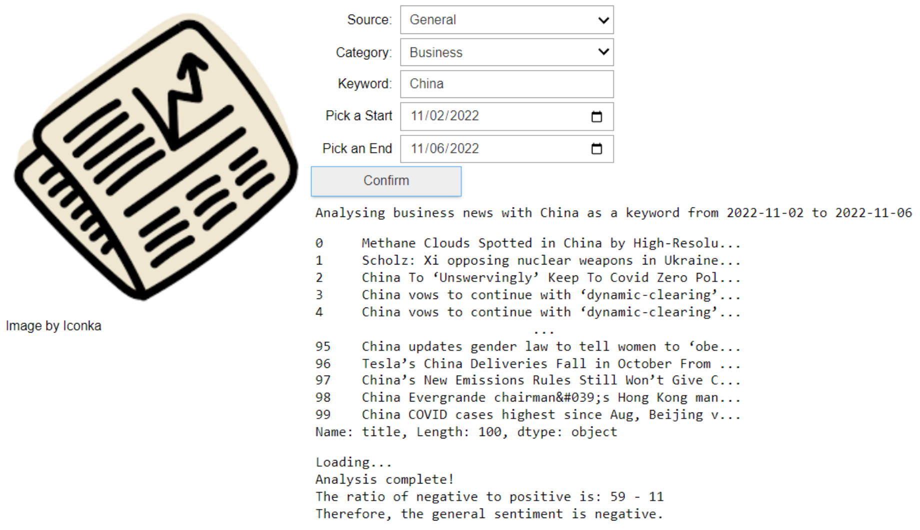
The problems of our dataset are imbalanced data due to the subjective tagging of news headlines, influenced by personal experiences and thoughts and the shortage of training data in the finance domain. For news sentiment analysis, the underlying factors that can cause the model mislabels are usually due to the headlines' tone, sarcasm and negation. As the headlines aim to catch attraction from audiences, they should be written shockingly and or using negations, idioms, and sarcasm. Such writing styles can affect our model's capability to identify the sentiment correctly.

4.4. News Retrieval

For us to retrieve the latest news and filter them based on the criteria that the user would like to view them in, we used the mediastack news API, which lets us gain access to worldwide news, headlines and blog articles in real-time so that our users can filter through the most recent news with over 7,500+ news sources for them to select from. Not only that but thanks to the functionalities of the API, we can also filter out items such as categories, dates, specific keywords and news sources to allow the user to gather the general sentiment of either more specific conditions if they want more advanced information or to leave the settings at their default selections for beginners if they would just like a general forecast across all sources and categories.

4.5 UI

For our UI, to have our information and the inputs of the user to be in the same system, we decided to go with ipywidgets which allows input features in the form of widgets such as dropdowns for certain categories and specific news outlets that we deemed as popular such as CNN or BBC, text fills which allows for certain keywords to be searched, for example, if the user was curious on how Elon Musk buying over twitter would affect the economic forecast, they could put his name in the keywords section and lastly, a date range specifier for the user so they can select older news sources if they desire or if they are curious if economic predictions were correct, etc. However, if the user is uncertain about all of these fields, we also have general as a choice which allows the user to vaguely see how the forecast is across all news sources, categories, etc for the day. We then used Voila which allows it to be displayed in the form of a web app and lastly, to make it easily accessible and usable to all, anytime and anywhere, used Binder which is a 3rd party site as a platform to host our application so anyone can use it on the world wide web.



Source: General
Category: Business
Keyword: China
Pick a Start 11/02/2022
Pick an End 11/06/2022

```
Analysing business news with China as a keyword from 2022-11-02 to 2022-11-06

0     Methane Clouds Spotted in China by High-Resolu...
1     Scholz: Xi opposing nuclear weapons in Ukraine...
2     China To ‘Unswervingly’ Keep To Covid Zero Pol...
3     China vows to continue with ‘dynamic-clearing’...
4     China vows to continue with ‘dynamic-clearing’...
                              ...
95    China updates gender law to tell women to ‘obe...
96    Tesla’s China Deliveries Fall in October From ...
97    China’s New Emissions Rules Still Won’t Give C...
98    China Evergrande chairman&#039;s Hong Kong man...
99    China COVID cases highest since Aug, Beijing v...
Name: title, Length: 100, dtype: object

Loading...
Analysis complete!
The ratio of negative to positive is: 59 - 11
Therefore, the general sentiment is negative.
```

Figure 19 UI Interface

5. Conclusion

In conclusion, despite several innovation methods we attempted to implement, model performance shows that FinBERT is still the optimal model we have decided to use. The other methods we chose to adopt were unable to capture the deeper nuances between the relationships in words.

As we are students without in-depth knowledge in the financial domain, our models could significantly improve if our data were to be more accurately hand-labelled by those with greater expertise in this domain.

We achieved our objective of creating a simple-to-use platform for users to access financial news sentiment in the form of a web application with a simple user interface, eliminating the need for users to learn and understand the deeper underlying coding principles.

6. Contributions

- BARABASEV DANIIL (N2202450G): Data Crawling, BERT/FinBERT, report completion
- CHRISTOPHER ARIF SETIADHARMA (U2022400H): Innovation: (CNN-LSTM & Sarcasm Detection)
- KOH YI JOSHUA (U1922368F): Data labelling, statistics, UI and news retrieval
- RYAN PHUAY QIAN HAO (U1922240H): Data labelling, statistics and visualisation, NLTK VADER
- TRAN HIEN VAN (U1920891J): Data Crawling, Classification: CNN, LSTM, all models code standardization, Innovation: Ensemble learning method

7. Table of Figures

8. References

[1] E. Cambria, S. Poria, A. Gelbukh and M. Thelwall, "Sentiment Analysis Is a Big Suitcase," in *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 74-80, November/December 2017, doi: 10.1109/MIS.2017.4531228.
[2] R. Misra and P. Arora, "Sarcasm Detection using Hybrid Neural Network," arXiv, 2019, doi: 10.48550/ARXIV.1908.07414.
[3] N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, and A. Gelbukh, "Sentiment and Sarcasm Classification with Multitask Learning," arXiv, 2019, doi: 10.48550/ARXIV.1901.08014.
[4] CS231n Convolutional Neural Networks for Visual Recognition. (2022). CS231n: Convolutional Neural Networks for Visual Recognition. https://cs231n.github.io/neural-networks-1/
[5] https://machine-learning-company.nl/en/technical/convolutional-neural-network-text-classification-with-risk-assessment-eng/
[6] https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[7] http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/

8.1 Our data

The file containing the results:
https://entuedu-my.sharepoint.com/:u:/g/personal/n2202450g_e_ntu_edu_sg/ETgQj5ME0sZAlatVy8yoA_kB_7OlfR6K5waYtAOZt4eMug?e=YWradQ
The file containing the source code for our project:
https://entuedu-my.sharepoint.com/:u:/g/personal/n2202450g_e_ntu_edu_sg/EUOQonqHsF9Pl8gSjEqdUv8BAIdMHclbWm4UKjTpTv5wJg?e=hXp4TQ