H6751 Text and Web Mining

Group Project Report

Title: Predicting the Trend of DJIA Stock Index by Categorizing the Reddit News Headlines

Word Count: 2612 words
(excluding the Cover and the References)

Submitted by:

Name	*Matric No.*	*Email*
PENG YANBIN	G2002419K	PENG0107@e.ntu.edu.sg
LI NANXI	G2001409J	LINA0016@e.ntu.edu.sg

Date: April 1, 2021

Introduction

1.1 Background

Dow Jones Industrial Average (DJIA) is one of the most important indexes of the US stock market. It measures the daily stock market trend by integrating the stock prices of 30 major companies in USA listed on New York Stock Exchange (NYSE) and NASDAQ. And these 30 public-owned large companies are considered as leaders in the economy of United States. And therefore, DJIA can be seen as a holistic indicator of the US stock market and is strongly related to the investors' sentiment. In our research, we use the DJIA as our target variable.

In the field of economics, news can be viewed as an indicator of the changes and sentiment of the society and the finance market. When the news reflects the sentiment of the world, the news can also change people's attitudes and opinions on their investment behaviors and afterwards the market trend.

To sum up our ideas, before modelling, we can presume that the textual content of the news can be effective in predicting the market (index) trend.

1.2 Literature Review

From the perspectives of institutional traders, when we evaluate the stock market and trend of stock prices, news is a critical component that we cannot forget about. We can extract fundamentals and qualitative information from news reports which affect the expectations of the market participants. (Hagenau et al., 2013).

Similarly, from the perspective of the academics, large number of literatures and studies have proven that news has significant impact on the change of the stock market. One existent research has indicated that the headlines of news is a significant factor for predicting the stock prices. And some researchers have successfully predicted the future profits of an company based on sentiment analysis of news (Schumaker & Chen, 2009). Schumaker et al. (2012) discovered that subjective news articles are easier to predict the direction of stock prices by sentiment analysis of financial news articles. Gorth and Muntermann (2011) conducted a research on intraday market risk management by analysis textual data that disclose risk. However, in all the research we browsed through, the accuracies of predicting stock prices are not significantly high.

1.3 Structure and Objective

The objective of our study is to predict the trend of stock market (DJIA index) by analyzing the intraday news headlines on Reddit.

Our study carries out sentiment analysis on the intraday news headlines of Reddit World News channel and established multiple neural network models to predict 3 types of trends in DJIA index (rise, not changed, decrease). The 4 neural network models we built are CNN, RNN, Fast-text Model and BERT.

With the four models, we tuned the hyperparameters based on our understandings of the rationale of the models. After tuning, we also compared and interpreted the results from the various models, and luckily, we have successfully drawn some constructive results and conclusions from the research.

Data Preprocessing

Based on the raw dataset, we first carried out text preprocessing to clean the data, remove the stop words and split the dataset into train data, test data and validation data. Then, we tokenized the news text into word tokens and form the vocabulary. We used one hot encoding to represent the news text and carried out word embedding. Finally, we can input the vectors to our deep learning models for training.

News headline data from Reddit World News, the trend of DJIA index is discretized into 3 categories:

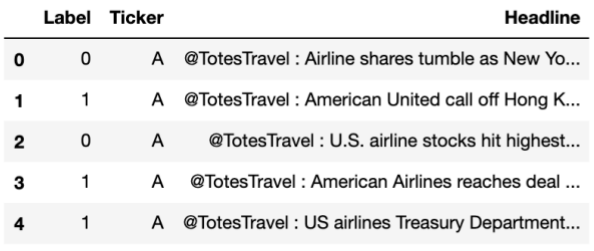
0 - stock went down when the news article was published.
1 - stock went up when the news article was published.
2 - stock stayed the same when the news article was published.

The source of the above data comes from Kaggle. (https://www.kaggle.com/search)

2.1 Data profile:

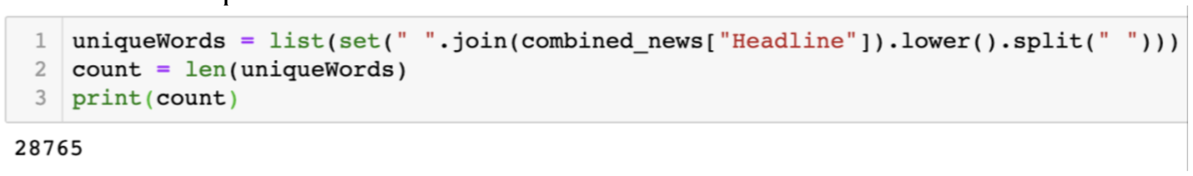
There are 13181 rows of news headlines in the raw dataset, and 3 columns which are Label, Ticker and Headline respectively. We will only use the 'Label' and 'Headline' column in our study.

The top five rows of the dataset are shown as below:



	Label	Ticker	Headline
0	0	A	@TotesTravel : Airline shares tumble as New Yo...
1	1	A	@TotesTravel : American United call off Hong K...
2	0	A	@TotesTravel : U.S. airline stocks hit highest...
3	1	A	@TotesTravel : American Airlines reaches deal ...
4	1	A	@TotesTravel : US airlines Treasury Department...

There are 28765 unique words in the whole dataset:



```
uniqueWords = list(set(" ".join(combined_news["Headline"]).lower().split(" ")))
count = len(uniqueWords)
print(count)
```

```
28765
```

Categorial news headlines counts in our dataset:

Label	Counts
0	8565
1	4300
2	316

2.2 Data preprocessing:

Step 1. Transfer all words into lower case, and filter the non-alphabetic characters

Step 2. Remove the stop-words and conduct lemmatization.

Words that appear too frequently like “me, I, this” in the raw text have little practical meaning for the subsequent text mining task. These words are normally auxiliary words, adverbs, prepositions, conjunctions, etc. And therefore we removed the stop words using the internal package in Python.

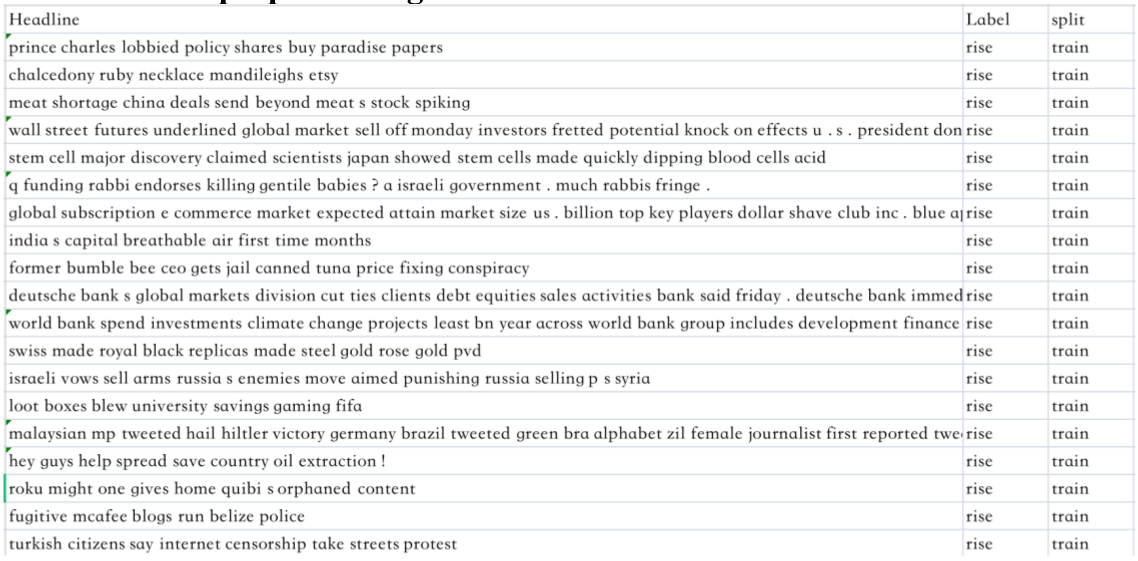
Also, to unify the forms of words and to reduce confusion for the models, we utilized lemmatization for preprocessing the textual content, i.e., the news headlines in our dataset.

Step 3. Split the dataset into train data, validation data and test data randomly.
The proportion of splitting is 0.7 for train data, 0.15 for validation data, 0.15 for test data. To have a more balanced split result, we specified a reasonable splitting proportion, and carried out the splitting proportion based on ‘Label’ value of the dataset to avoid bias.

Splitting data	**Counts**
Train data	9266
Test data	1976
Validation data	1976

Since there is not null value in the whole dataset, we don’t need to clean and process missing value.

2.3 Dataset after preprocessing:



Headline	Label	split
prince charles lobbied policy shares buy paradise papers	rise	train
chalcedony ruby necklace mandileighs etsy	rise	train
meat shortage china deals send beyond meat s stock spiking	rise	train
wall street futures underlined global market sell off monday investors fretted potential knock on effects u . s . president don	rise	train
stem cell major discovery claimed scientists japan showed stem cells made quickly dipping blood cells acid	rise	train
q funding rabbi endorses killing gentile babies ? a israeli government . much rabbis fringe .	rise	train
global subscription e commerce market expected attain market size us . billion top key players dollar shave club inc . blue a	rise	train
india s capital breathable air first time months	rise	train
former bumble bee ceo gets jail canned tuna price fixing conspiracy	rise	train
deutsche bank s global markets division cut ties clients debt equities sales activities bank said friday . deutsche bank immed	rise	train
world bank spend investments climate change projects least bn year across world bank group includes development finance	rise	train
swiss made royal black replicas made steel gold rose gold pvd	rise	train
israeli vows sell arms russia s enemies move aimed punishing russia selling p s syria	rise	train
loot boxes blew university savings gaming fifa	rise	train
malaysian mp tweeted hail hiltler victory germany brazil tweeted green bra alphabet zil female journalist first reported twe	rise	train
hey guys help spread save country oil extraction !	rise	train
roku might one gives home quibi s orphaned content	rise	train
fugitive mcafee blogs run belize police	rise	train
turkish citizens say internet censorship take streets protest	rise	train

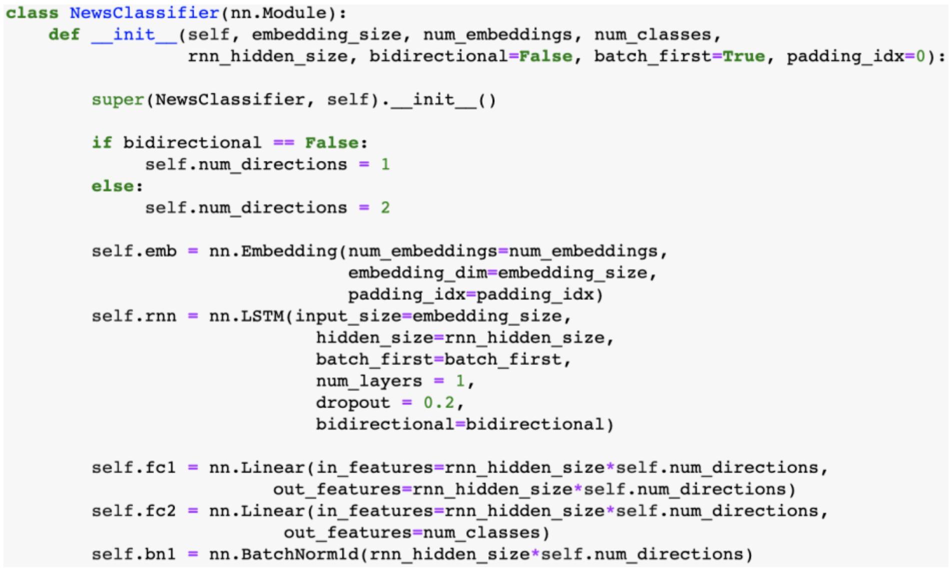
Models

4.1 RNN

As indicated in the data profile above, our project is to identify the 3 stock index trends by categorizing the textual contents (i.e., news headlines). Therefore, RNN provides models suitable for analyzing textual contents while taking the words’ sequence into consideration.

4.1.1 LSTM

Also, since the length of a news headline can reach above 100, when building the prediction model, we need to avoid the problem of vanishing gradients and to retain the long-text information holistically. In this case, for the first trials, we used LSTM as our neural network models. Our model is listed as below:

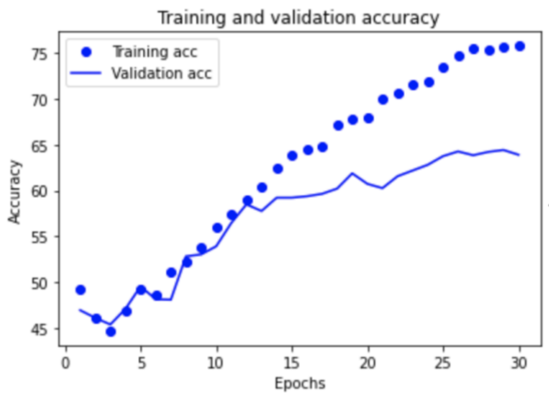


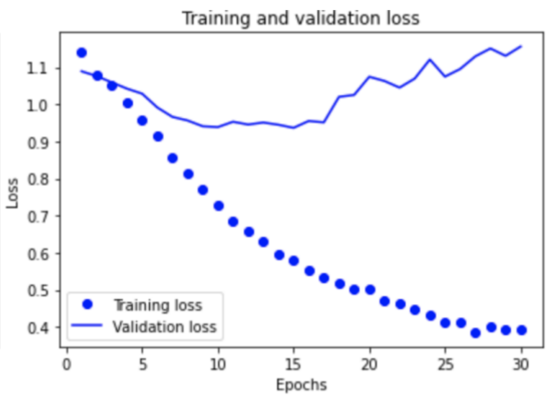
```
class NewsClassifier(nn.Module):
    def __init__(self, embedding_size, num_embeddings, num_classes,
                 rnn_hidden_size, bidirectional=False, batch_first=True, padding_idx=0):

        super(NewsClassifier, self).__init__()

        if bidirectional == False:
            self.num_directions = 1
        else:
            self.num_directions = 2

        self.emb = nn.Embedding(num_embeddings=num_embeddings,
                                embedding_dim=embedding_size,
                                padding_idx=padding_idx)
        self.rnn = nn.LSTM(input_size=embedding_size,
                           hidden_size=rnn_hidden_size,
                           batch_first=batch_first,
                           num_layers = 1,
                           dropout = 0.2,
                           bidirectional=bidirectional)

        self.fc1 = nn.Linear(in_features=rnn_hidden_size*self.num_directions,
                             out_features=rnn_hidden_size*self.num_directions)
        self.fc2 = nn.Linear(in_features=rnn_hidden_size*self.num_directions,
                              out_features=num_classes)
        self.bn1 = nn.BatchNorm1d(rnn_hidden_size*self.num_directions)
```

Tuning Process:

trial	accuracy	loss	model	embedding size	rnn hidden size	bidirectional	epoch	learning rate	drop out	batch size
1	52.656	0.933	lstm	150	64	FALSE	20	0.0001	0.2	64
2	57.031	0.951	lstm	150	64	TRUE	20	0.0001	0.2	64
3	57.187	0.986	lstm	150	64	TRUE	30	0.0001	0.2	64
4	59.895	0.911	lstm	150	100	TRUE	30	0.0001	0.2	64
5	58.958	0.914	lstm	150	150	TRUE	30	0.0001	0.2	64
6	55.624	0.937	lstm	150	125	TRUE	30	0.0001	0.2	64
7	57.343	0.975	lstm	200	100	TRUE	30	0.0001	0.2	64

After tuning our LSTM models, we found out that the best hyperparameter combination is shown in the highlighted row in the capture above (embedding size: 150, rnn hidden layer size: 100, learning rate: 0.0001 and epoch:30). The best LSTM model has an accuracy of 59.895% with 0.911 in loss. The graphs of accuracy and loss change are shown as below.



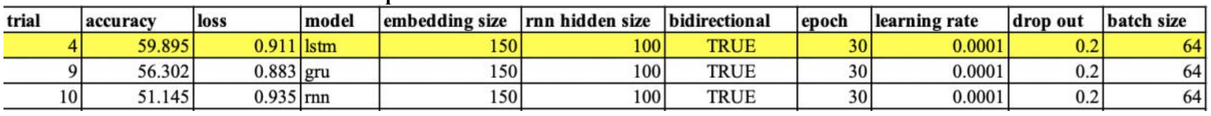


Also, we found out that the bidirectional model is way better than the non-bidirectional ones because it can incorporate more information from words both preceding and following.

4.1.2 RNN and GRU

Also, we tried out the traditional RNN and GRU models, but neither of them performs better than LSTM in this project. As just mentioned, when categorizing the long textual data, LSTM can contain more information of the texts and provide more accurate prediction results.

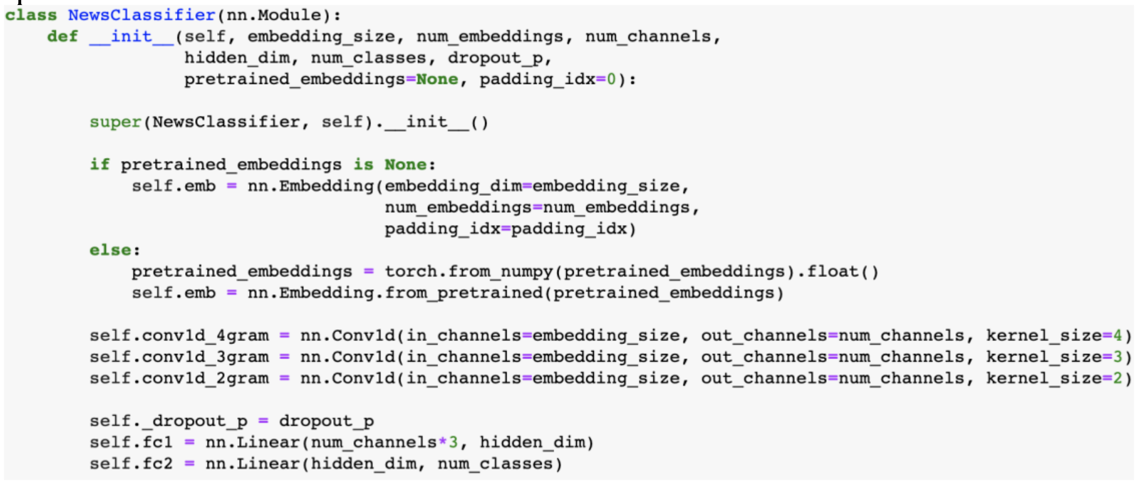
The RNN and GRU model performances are shown as below:



trial	accuracy	loss	model	embedding size	rnn hidden size	bidirectional	epoch	learning rate	drop out	batch size
4	59.895	0.911	lstm	150	100	TRUE	30	0.0001	0.2	64
9	56.302	0.883	gru	150	100	TRUE	30	0.0001	0.2	64
10	51.145	0.935	rnn	150	100	TRUE	30	0.0001	0.2	64

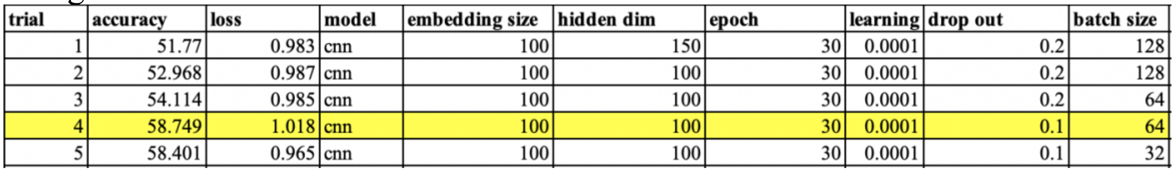
4.2 CNN

However, since this is a textual classification problem, we also implement CNN models for prediction. Our model is as below:



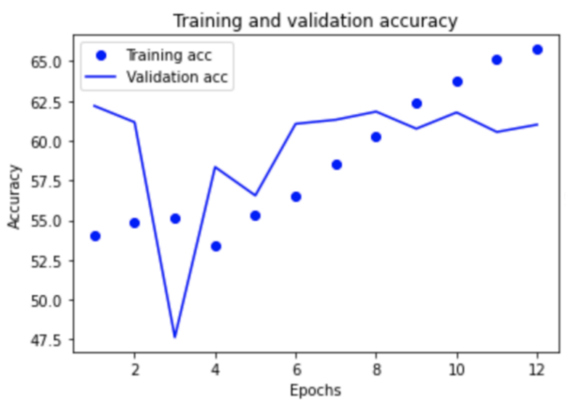
```
class NewsClassifier(nn.Module):
    def __init__(self, embedding_size, num_embeddings, num_channels,
                 hidden_dim, num_classes, dropout_p,
                 pretrained_embeddings=None, padding_idx=0):

        super(NewsClassifier, self).__init__()

        if pretrained_embeddings is None:
            self.emb = nn.Embedding(embedding_dim=embedding_size,
                                    num_embeddings=num_embeddings,
                                    padding_idx=padding_idx)
        else:
            pretrained_embeddings = torch.from_numpy(pretrained_embeddings).float()
            self.emb = nn.Embedding.from_pretrained(pretrained_embeddings)

        self.conv1d_4gram = nn.Conv1d(in_channels=embedding_size, out_channels=num_channels, kernel_size=4)
        self.conv1d_3gram = nn.Conv1d(in_channels=embedding_size, out_channels=num_channels, kernel_size=3)
        self.conv1d_2gram = nn.Conv1d(in_channels=embedding_size, out_channels=num_channels, kernel_size=2)

        self._dropout_p = dropout_p
        self.fc1 = nn.Linear(num_channels*3, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, num_classes)
```

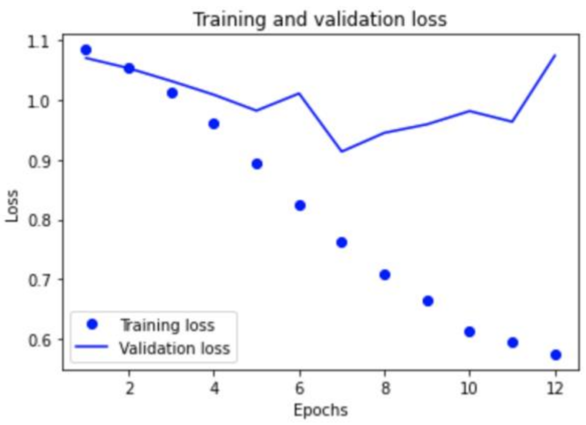
Tuning Process:



trial	accuracy	loss	model	embedding size	hidden dim	epoch	learning	drop out	batch size
1	51.77	0.983	cnn	100	150	30	0.0001	0.2	128
2	52.968	0.987	cnn	100	100	30	0.0001	0.2	128
3	54.114	0.985	cnn	100	100	30	0.0001	0.2	64
4	58.749	1.018	cnn	100	100	30	0.0001	0.1	64
5	58.401	0.965	cnn	100	100	30	0.0001	0.1	32

Our model tuning process is listed as above. And the hyperparameter parameter combination with the best performance is, as highlighted in the yellow row, 100 in embedding size, 100 in hidden layer dimensions, 0.1 in dropout rate and 64 in batch size. Also, the accuracy and loss of the model are 58.749% and 1.018 respectively. Our graphs of accuracy and loss are shown as below:





4.3 Fast-Text Model

Fast-Text is a word vector computation and text classifier open-sourced by Facebook AI Research in 2016.The most prominent advantage for Fat-text is its training speed. It is able to process over 1 billion words in 10 minutes with standard multi-core CPUs. (https://fasttext.cc/docs/en/supervised-tutorial.html)

The structure of Fast-Text model is composed of three layers: input layer, hidden layer and output layer. The structure of Fast-Text is very similar to the one of CBOW. Fast-Text predicts the sentence label based on the input context.

Input Layer: To take the sequence of words into account, FastText uses the N-gram feature. As shown by the figure below, the input x is n-gram vectors. The features are embedded and averaged to form the hidden variable. (Joulin et al., 2016)

Hidden Layer: In Fast-Text, the hidden layer is obtained by summing and averaging the input layers, multiplied by the weight matrix A.

Output layer: Multiply the hidden layer with the weight matrix B.

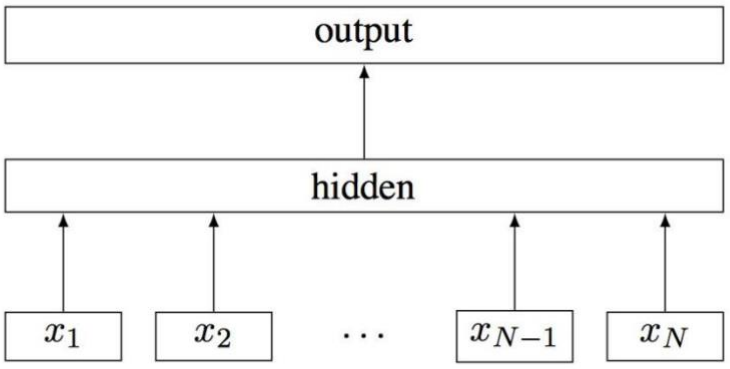
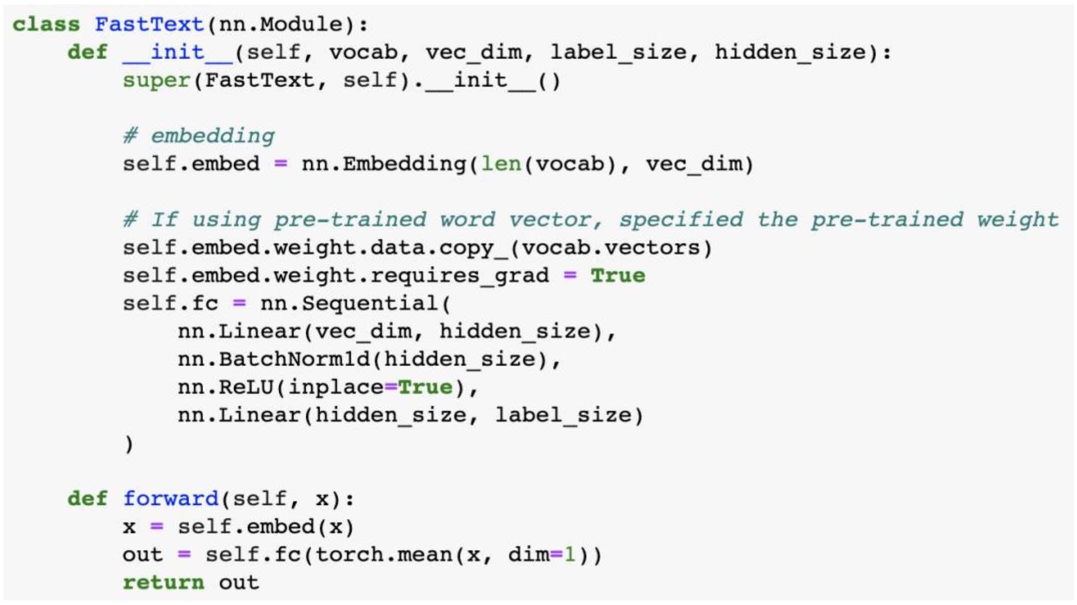


Figure x. Fasttext model structure.

The formula of loss function of Fast-text is shown as below: y_n is the true label and x_n is the normalized N-gram feature. A and B are weight matrices.

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n))$$

Modelling:
When defining the Fast-text model class, we firstly used word embedding and then introduced the full-connected neural network. After that, we conducted batch normalization after the hidden layer in order to avoid the problems of overfitting, gradient explosion and gradient vanishing. Lastly, we used Re-Lu as the activation function. The code of modelling is shown as below:



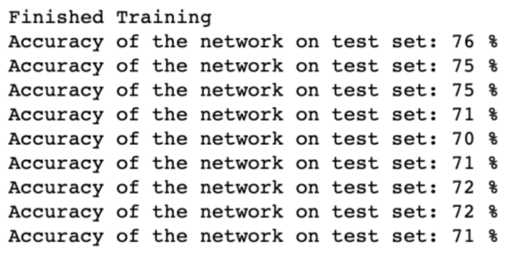
```python
class FastText(nn.Module):
    def __init__(self, vocab, vec_dim, label_size, hidden_size):
        super(FastText, self).__init__()

        # embedding
        self.embed = nn.Embedding(len(vocab), vec_dim)

        # If using pre-trained word vector, specified the pre-trained weight
        self.embed.weight.data.copy_(vocab.vectors)
        self.embed.weight.requires_grad = True
        self.fc = nn.Sequential(
            nn.Linear(vec_dim, hidden_size),
            nn.BatchNorm1d(hidden_size),
            nn.ReLU(inplace=True),
            nn.Linear(hidden_size, label_size)
        )

    def forward(self, x):
        x = self.embed(x)
        out = self.fc(torch.mean(x, dim=1))
        return out
```

In the training process, we used a pre-trained word vector in Glove - *glove.6B.300d.txt.* The advantage of using a pre-trained word vector is that it can speed up the Fast-Text training process and solve the problem of insufficient training data. Our training dataset size is less than 10k which can be considered as insufficient. And therefore, using a pre-trained model can help to improve the model performance.

Hyperparameter Set and Testing Result:

Splitting data	**Counts**
batch_size	64
epoch	10
embedding_dim	300
Learning rate	0.001
Hidden_size	200

After testing, it's shown that the average prediction accuracy is around 70%.



```
Finished Training
Accuracy of the network on test set: 76 %
Accuracy of the network on test set: 75 %
Accuracy of the network on test set: 75 %
Accuracy of the network on test set: 71 %
Accuracy of the network on test set: 70 %
Accuracy of the network on test set: 71 %
Accuracy of the network on test set: 72 %
Accuracy of the network on test set: 72 %
Accuracy of the network on test set: 71 %
```

Limitation:
Our model is a simplified form of the Fast-Text implementation. It doesn't use bigram or any hierarchical soft-max function.

4.4 BERT
As we all know, BERT is a newly introduced powerful text analyzing tool. In our research, we fine-tuned our BERT model for news headlines classification.

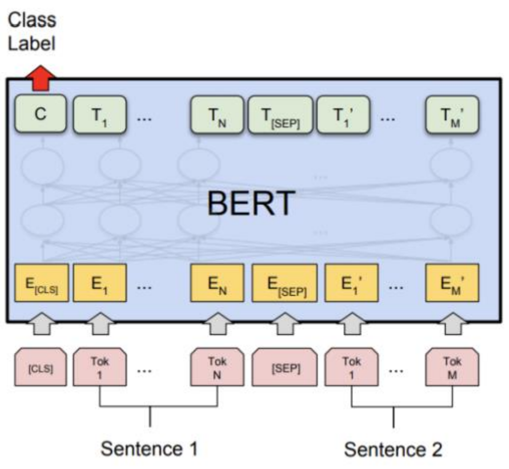


Figure x. Bert for classification. (Devlin et al., 2018)

The main reason why we choose BERT in modelling is that the structure of it makes it powerful in capturing the text data and therefore efficient in classifying the data. It not only used multi-head attention to take the importance of each word into consideration, but also use multiple encoders to capture the different features of a news headline. And thus, we use BERT in our research.

Modelling:
After transforming the news texts into lower case, we selected 'BERT-base-uncased' pretrained model for our task.

Devlin et al. (2018) suggested that the number of epochs from 2 to 4 is enough for the BERT model to work well. And in our task result, it's shown that after epoch 2, the loss on validation data started to increase and the model started to overfit.

After modifying the batch size, learning rate and drop-out rate, we obtained the parameter set that brings the model the best performance.

Hyperparameter Set and Result:

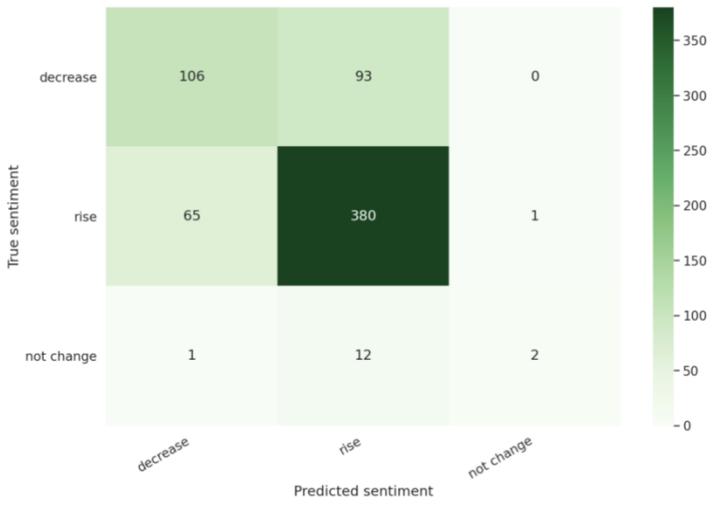
Params	**Value**
batch_size	16
Max embedding length	130
Learning rate	2e-5
Number of epochs	4
Num_workers	0
Drop out	0.3

In our final model, the accuracy of it reaches around 74%.

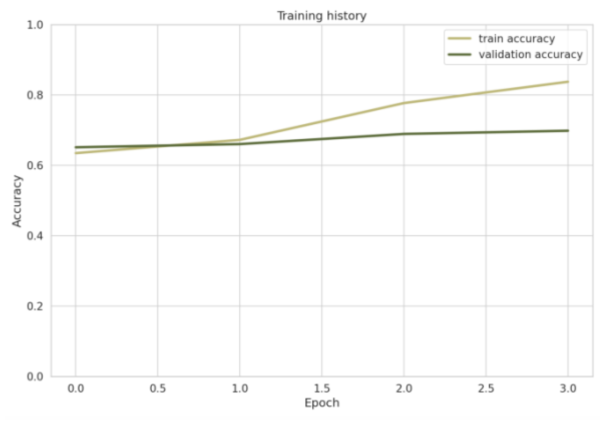
From the confusion matrix, we can notice that the precision for the labels of "decrease" and "not change" is much lower than that for the label of "rise". This indicates that it is harder for the model to correctly classify the news headlines implying deceasing or unchanged market trends. Unbalanced dataset could also be the reason for the score difference in the confusion matrix.

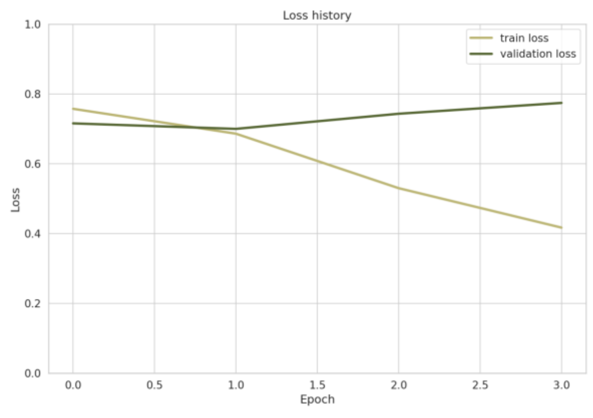
```
              precision    recall  f1-score   support

    decrease       0.62      0.53      0.57       199
        rise       0.78      0.85      0.82       446
  not change       0.67      0.13      0.22        15

    accuracy                           0.74       660
   macro avg       0.69      0.51      0.54       660
weighted avg       0.73      0.74      0.73       660
```

The confusion matrix of BERT model:



The accuracy and loss graphs of the BERT model:





Modelling Results

1. RNN(LSTM) outperforms CNN

In RNN, the structure of taking the word sequence into account is semantically interpretable and reasonable. The modified version of RNN, LSTM, helps the model to overcome the problem of vanishing gradients, which further makes the model a powerful one especially when our text content is in significant length.

However, in CNN, despite of its consideration for the n-grams within the textual content, the sequence of the news headlines is not considered. Therefore, we assume that the misplacing of the information may cause confusion and disturbance to the prediction model.

2. 2 new models (Fast-Text and BERT) outperform the traditional ones (RNN and CNN)

Both the new models achieved accuracies of over 70%, and BERT even outperforms the other 3 traditional neural network models with the accuracy of 74%. From this result, we can infer that both the new models are able to capture more information from the textual content due to their more sophisticated model structure.

Remaining Issues

1. Remaining Unpredictable Causation

Since the stock market trend is very hard to predict due to the various factors, the news headlines are only one aspect of the reflection of the market sentiment. Other reasons bringing volatility to the stock market, such as significant stock trading within one day, may not be easily noticed, quantified or predicted within the news headlines. Therefore, with the prediction accuracy of nearly 60%, we believe that our research has found out part of the causation for the market trend.

2. Limited Modelling Methods

There are many state-of-art neural network models with stronger power of extracting and capturing the information within the textual content in the area of deep learning. However, due to the limit of time, we cannot implement all of them and we only tried out two of them, Fast-Text and BERT.

3. Unbalanced Dataset

In our project, the portion of 3 news headline categories, ‘rise’, ‘not change’ and ‘decrease’, is 20:1:10. Due to the limited source of the news headlines, the dataset in our project isn’t balanced enough, and therefore the model itself may perform not as well in categorizing the news of certain categories.

4. Biased News Content
Since our news headline data source is only from Reddit and only the top 25 user-voted news headlines are selected, the dataset may not fully reflect the sentiment of the market. Instead, it is more of a representation of the hottest topics discussed worldwide and therefore it may be biased and not as effective in predicting the market trend. To overcome this problem, in the future, we may try to collect the finance and economic news headlines for stock market trend prediction instead.

Conclusion

1. News headline textual contents are effective in predicting the stock market trend:
Just as we all know, the stock market indexes all year long fluctuate with various factors both perceptible and imperceptible. However, the investors’ sentiment is still one of the most important indicators for the stock market trend. Therefore, we chose the dataset of news headlines for predicting the market trend since the investors’ sentiment can be reflected on and be changed by the news content.

After training and testing the models, we found out that all of the 4 neural network models have accuracies of over 50%. On top of that, both the new models (Fast-text and BEERT) even have accuracies of over 70%, which indicates that the well-trained models can be relatively effective in predicting the 3 types of market index trends rice, not change & decrease). In this case, we can come to the conclusion that the textual content of news headlines is effective in forecasting the market index change.

2. BERT is so far the most powerful tool for building the prediction model in our research:
As indicated in the modelling results above, after tuning, the RNN model (LSTM) outperforms the CNN one. Also, the BERT model is so far the best model in our research because it can capture and learn the information comprehensively and efficiently.

BERT utilizes multi-head attentions to quantify the importance and the sequence of the words in the corpus. On top of that, BERT model also introduces multiple encoder layers for capturing multi-dimensional information within the text (i.e., grammar, sentiment, etc.). In this case, the BERT model becomes the best model for news content classification in our research.

References

- Khan, W., Ghazanfar, M., Azam, M., Karami, A., Alyoubi, K., & Alfakeeh, A. (2020). Stock market prediction using machine learning classifiers and social media, news. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-020-01839-w
- Schumaker, R., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. ACM Transactions on Information Systems, 27(2), 1–19. https://doi.org/10.1145/1462198.1462204
- Hagenau, M., Liebmann, M., & Neumann, D. (2013). Automated news reading: Stock price prediction based on financial news using context-capturing features. Decision Support Systems, 55(3), 685–697. https://doi.org/10.1016/j.dss.2013.02.006
- Li, X., Huang, X., Deng, X., & Zhu, S. (2014). Enhancing quantitative intra-day stock return prediction by integrating both market news and stock prices information. Neurocomputing (Amsterdam), 142, 228–238. https://doi.org/10.1016/j.neucom.2014.04.043
- Li, X., Wu, P., & Wang, W. (2020). Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. Information Processing & Management, 57(5), 102212–. https://doi.org/10.1016/j.ipm.2020.102212
- Schumaker, R., Zhang, Y., Huang, C., & Chen, H. (2012). Evaluating sentiment in financial news articles. Decision Support Systems, 53(3), 458–464. https://doi.org/10.1016/j.dss.2012.03.001
- Joulin, E. Grave, P. Bojanowski, and T. Mikolov. (2016). Bag of Tricks for Efficient Text Classification. Facebook AI Research. cite arxiv:1607.01759.