Predicting Stock Market Trend using Daily News Headlines

Smit Anish Kiri, Nitin Kumar Mittal
[MSDS Fall 2019]
December 10, 2019

I. Introduction

Trading in the stock market is difficult because of its volatile nature. A lot of factors contribute to the change in stock prices. Generally, News Headlines have a significant impact on the movement of a company's stock prices [1]. Unbiased and quick decisions based on real-time news can yield a long-term profit in trading. But human sentiments intervene and lead to delayed and incorrect decisions. To identify the impact of the news on stock price trend (here trend being categorized into Increase, Decrease, No Change), several machine learning techniques like Neural Network [2], Decision Trees [3], Random Forest [3], Recurrent Neural Network Model like Long Short Term Memory (LSTM) [4] can be employed to solve the above-stated classification problem.

In this project, we aim to build and compare different classification models to predict the Stock Market Trend for the next day after every news release. Daily Stock Market Trends for the current day will be classified into 3 categories:

1. Increase: If the next day's close increases by more than 0.5% compared to the current day's open.
2. Decrease: If the next day's close decreases by more than 0.5% compared to the current day's open.
3. No Change: If the next day's close is within 0.5% change in open of the current day's open.

II. Dataset

To build the classification models, we needed to incorporate 2 types of data, textual data (News Headlines) and numerical data (stock data). The datasets that we used are:

1. Reddit News Dataset (textual data) [5]: This dataset contains the top 25 news headlines from Reddit for each day from August 8th, 2008 to May 1st, 2016. Next day's stock trend labels were calculated for the same date range. Fig. 1. is the word cloud for Reddit News Dataset.

Fig. 1. Word Cloud for Reddit News Dataset

2. The UCI News Aggregator Dataset (textual data) [6]: This dataset contains headlines, URLs, and categories for 422,937 news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014. The features of the dataset include news id, title (headline), URL, publisher, source, category, hostname and timestamp. Fig. 2. is the word cloud for UCI News dataset.

Fig. 2. Word Cloud for UCI News Dataset

3. Daily Stock Market Data (numerical data): The stock data was scraped using the *pandas_datareader* [7] library in python for given date range and company. We scraped stock data for Google from August 8th, 2008 to May 1st, 2016 for Reddit News Dataset and from March 10th, 2014 and August 10th, 2014 for UCI News

Aggregator Dataset. The features of the stock data included stock open rate, close rate, daily high, daily low and volume. We also added engineered features like week of year, day of week and boolean features like is_month_end, is_month_start, is_quarter_end, is_quarter_start, is_year_end, is_year_start. To apply modeling techniques on above mentioned categorically features we used one-hot encoding.

We encoded output labels for the next day's stock trend, using respective values:

1. no change in stock trend - 0
2. increase in stock trend - 1
3. decrease in stock trend - 2

III. Data Processing and Modelling

A. Using Conventional Models with Stock Data

Our initial approach was to implement conventional models like Decision Trees, Random Forest and Deep Neural Networks from Keras to predict the stock trend by only using stock data with engineered features for the date range Reddit News Dataset was available. Stock data was shuffled and split into train and test datasets using 80% and 20% ratios respectively.

Table 1. shows accuracy, precision and recall achieved on test dataset from different models. Fig. 3. shows ROC [8] Curve for Decision Tree, Fig. 4. shows ROC Curve for Random Forest with 1000 estimators and Fig. 5. shows ROC Curve for Deep Neural Network [Appendix 2].

Models	Accuracy	Precision	Recall
Decision Tree	36.8%	36.4%	36.8%
Random Forest (1000 Estimators)	35.0%	35.5%	35.0%
Deep Neural Network	34.7%	38.4%	34.7%

Table 1. Performance metrics for conventional models on Test dataset (stock data)

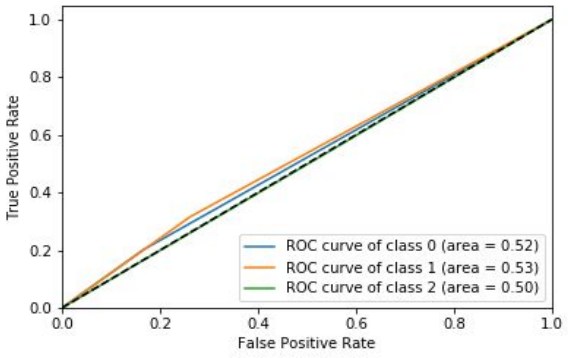

Fig. 3. ROC Curve for Decision Tree

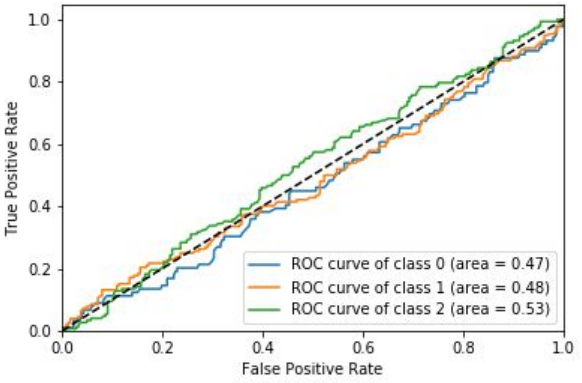

Fig. 4. ROC Curve for Random Forest

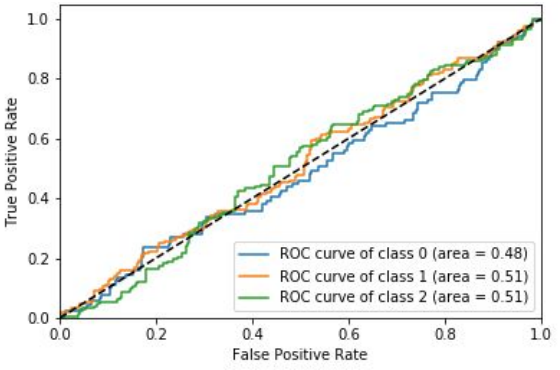

Fig. 5. ROC Curve for Deep Neural Network

From the above performance metrics table and ROC curves for conventional models on test data, we can say that these models were not performing well with stock data, just giving an absolute increase of nearly 3% on random guessing(33%) of 3 classes.

B. Using Conventional Models with Bag-of-Words Encodings for News Data

B.1 Using Textual Data (News) only with Bag-of-Words Encoding [9]

Our second approach was to use textual news data to predict the next day's stock trends. In the Reddit News Dataset, we combined multiple news headlines available for each day, creating a single sentence. We then cleaned the data by converting the text to lowercase and removing English stopwords [10] and specialized characters.

In order to use the textual data to train machine learning models, we used the Bag-of-Words encoding technique. Bag-of-Words is a representation of text that describes the occurrence of words within a document (single day news in our case). It involves building a vocabulary of known words and measuring the presence of words in documents (word frequencies in our case).

To apply Bag-of-Words encoding for Reddit News Dataset after processing and combining news, dataset was shuffled and split into train and test datasets using 80% and 20% ratios respectively. Vocabulary was built using train dataset with 8000 most frequent words in train dataset news corpus and sparse document term matrix were created for both train and test datasets using sklearn's CountVectorizer function [11].

We implemented models like Random Forest(with 1000 estimators), AdaBoost(with 50 decision stumps), Deep Neural Network [model summary- Appendix 3] with Reddit News Dataset encoded by Bag-of-Words.

Table 2. shows accuracy, precision and recall achieved on test dataset from different models. Fig. 6. shows ROC Curve for Random Forest with 1000 estimators, Fig. 7. shows ROC Curve for AdaBoost with 50 Decision Stumps and Fig. 8. shows ROC Curve for Deep Neural Network.

Models	Accuracy	Precision	Recall
Random Forest (1000 Estimators)	36.4%	36.4%	36.4%
AdaBoost (50 Decision Stumps)	36.4%	36.4%	36.4%
Deep Neural Network	35.2%	35.2%	35.2%

Table 2. Performance metrics on Test dataset (Reddit News with Bag-of-Words encoding)

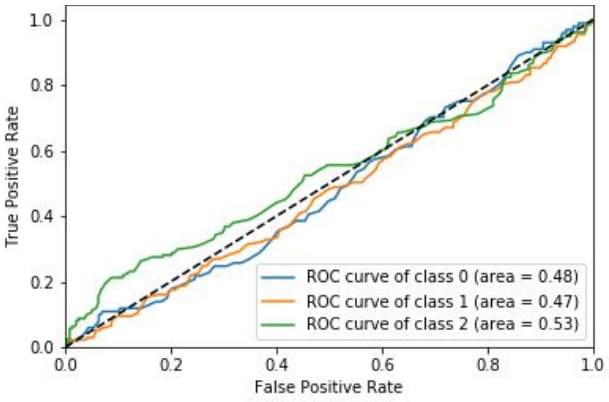

Fig. 6. ROC Curve for Random Forest

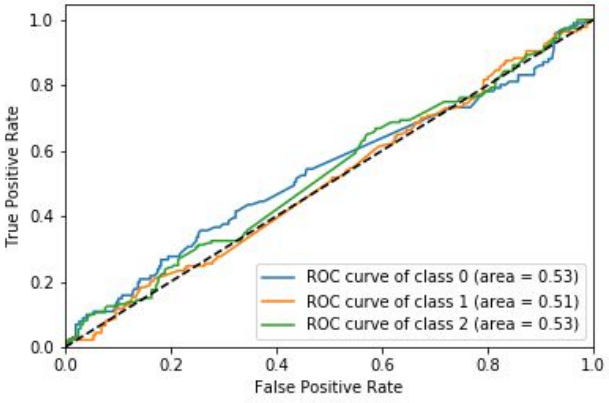

Fig. 7. ROC Curve for AdaBoost

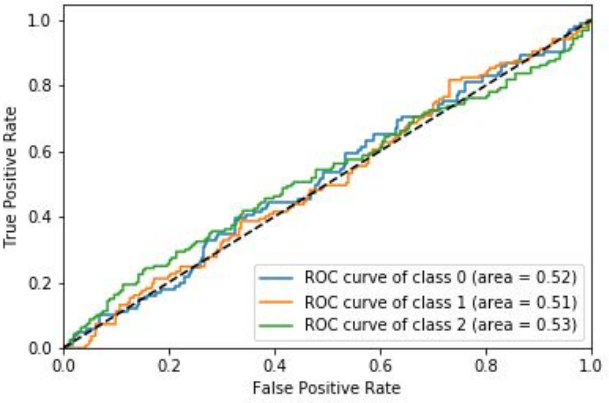

Fig. 8. ROC Curve for Deep Neural Network

Using Reddit News Data with Bag-of-Words encoding and implementing the models (mentioned in Table 2.) did not improve the performance metrics.

B.2 Using Both Textual Data (News) with Bag-of-Words Encoded and Stock Data.

We combined Reddit News Data encoded by Bag-of-Words and Stock Data to train conventional models hoping to improve the performance metrics. Encoded textual data and stock data with engineered

features were combined using the date as a common column. Prepared data was shuffled and split into train and test datasets using 80% and 20% ratios respectively.

We implemented models like Random Forest (with 1000 estimators), AdaBoost (with 50 decision stumps), Deep Neural Network [model summary- Appendix 3] with Reddit News Dataset encoded by Bag-of-Words.

Table 3. shows accuracy, precision and recall achieved on test dataset from different models. Fig. 9. shows ROC Curve for Random Forest with 1000 estimators, Fig. 10. shows ROC Curve for AdaBoost with 50 Decision Stumps and Fig. 11. shows ROC Curve for Deep Neural Network.

Models	Accuracy	Precision	Recall
Random Forest (1000 Estimators)	36.7%	36.7%	36.7%
AdaBoost (50 Decision Stumps)	37.4%	37.4%	37.4%
Deep Neural Network	34.4%	34.4%	34.4%

Table 3. Performance metrics on Test dataset
(Combined Reddit News with Bag-of-Words encoding and Stock data)

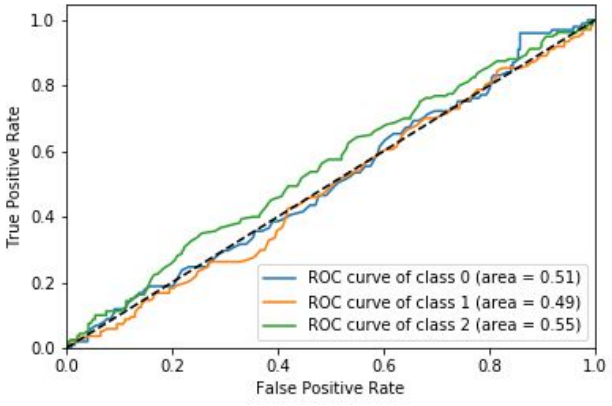

Fig. 9. ROC Curve for Random Forest

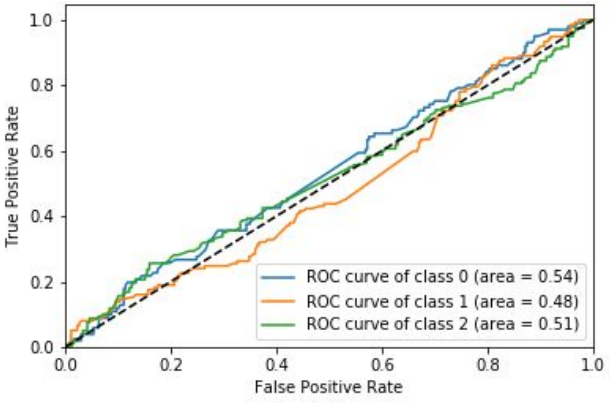

Fig. 10. ROC Curve for AdaBoost

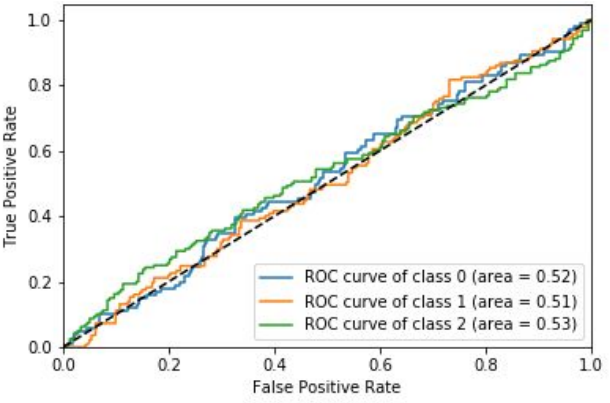

Fig. 11. ROC Curve for Deep Neural Network

Using combined Reddit News Data with Bag-of-Words encoding and stock data and implementing the models (mentioned in Table 3.) gave an insignificant improvement in performance metrics for test dataset.

C. Using Word Embeddings for Encoding News Data and Stock Data Sequences with Long-Short Term Memory Recurrent Neural Network Model

Since implementing conventional models mentioned under Section B.1 and B.2 with Bag-of-Words encodings for textual data did not significantly improve test performance metrics, we started using Recurrent Neural Networks Modeling Techniques and Word Embeddings to encode textual Data.

C.1 Word Embeddings

Word embeddings [12] is a text encoding technique which is capable of capturing the context of a word in a sentence. This encoding technique maps individual words to a real-valued vector in n-dimension space. It allows words with similar meaning to have similar vector representation and thus help machine learning models to generalize between words in sentences. Pre-trained word

vector embeddings like GloVe [13] can also be used that help model to converge quickly and give better performance metrics.

Fig. 12. gives an example where words of two sentences that have similar contexts are mapped close to each other.

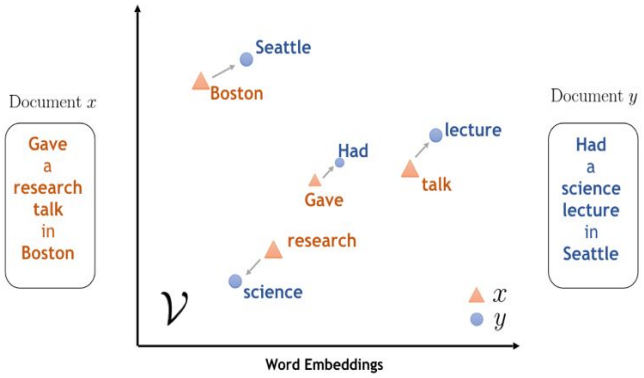

Fig. 12. Vector Representation by Word Embeddings [14]

C.2 Implementing Word Embeddings

In this approach, we used the Reddit News Dataset and for each day we combined multiple news headlines available into a single sentence thus creating a single sentence for every day. We got news sentences for nearly 2000 days.

Following steps were performed to encode textual data (Reddit News Dataset) using Word Embeddings:

1. News Data was cleaned by removing stop words and special characters.

2. Vocabulary of size 35,184 unique words was created using cleaned news data.

3. Every word in vocabulary was assigned an index as per its sequence in vocab. Ex. if "man" is present at index 10 in vocabulary then 10 is assigned as it's index.

4. Used pre-trained GloVe vector embeddings with 400,000 different words, each mapped to 300 dimensional vectors. Filtered our vocabulary to include only those words which appeared more than 10 times in the Reddit News Dataset or were present in GloVe's vocabulary.

5. Added <UNK> (Unknown) and <PAD> (Padding) to our vocabulary and indexed them.

6. Created 300 dimensional vector embeddings for every word in vocabulary by extracting embeddings from GloVe and appending normalized vector embeddings for 43 words that were not present in GloVe.

7. Final vocabulary had 31,674 words and vector embeddings had 300 dimensional embeddings for 31,674 words.

8. Converted textual sentences into sequences of word indexes using indexes from vocabulary. Visualization for news sequences is given in Appendix 4.

9. Fixed the length of each sequence to 308 (majority length sequences).

C.3 Preparing Sequences of Stock Data

To classify the stock trend corresponding to a day, we also took stock data with engineered features for the past 30 days. The shape of our data became (1959, 30, 81). Since we used 30 days of data, our labels started from the 31st day thus reducing the overall data size. Visualization for stock sequences is given in Appendix 5

C.4 Creating Train and Test Datasets

It was important to use the same day's news and stock sequences to classify stock trends. To create train and test datasets we took news and stock sequences for common dates and sorted both sequences by dates in ascending order. Using the same shuffling for news and stock sequences we split both sequences into train and test datasets with 80% and 20% split ratios respectively.

C.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a special kind of Recurrent Neural Network that is capable of learning long term dependencies. It has a memory unit which enables it to remember historical values while training. Can be visualized from Fig. 13.

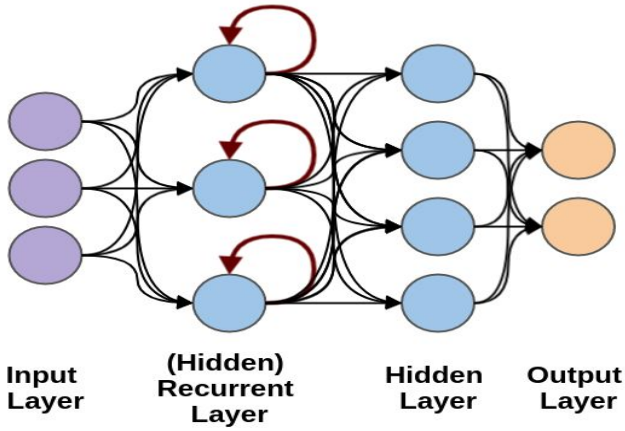

Fig.13. Recurrent Neural Network Architecture [15]

Since stock trends are dependent on historical values, we incorporated LSTM for training. Also, LSTM performs better with textual data as it can remember the dependencies of words in a sentence to get the essence of a sentence [16].

Basic LSTM architecture (Fig. 14.) is composed of LSTM units where each LSTM unit (Fig. 15.) has a memory cell and three gates: forget gate, input gate and output gate. These gates control the flow of information inside the LSTM unit.

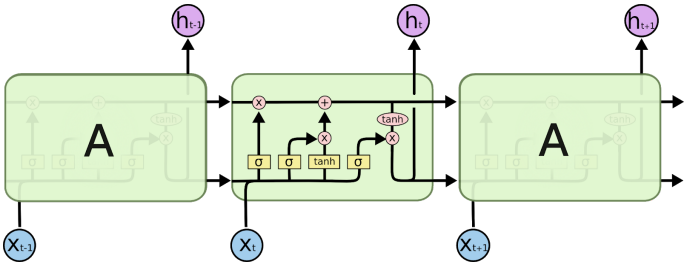

Fig. 14. Basic LSTM Architecture [17]

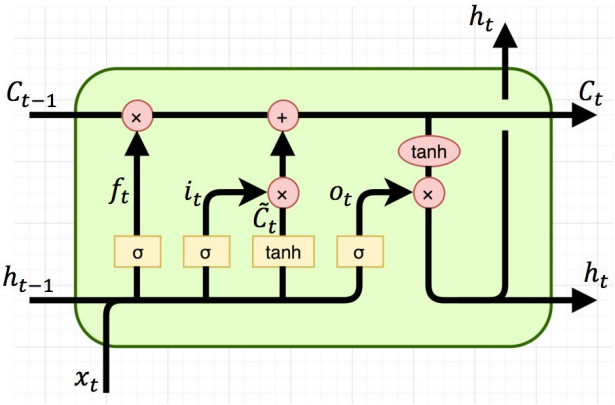

Fig. 15. LSTM Cell [18]

The forget gate (f_t) decides what information to discard from the cell state. The input gate (i_t) decides what information to store in the cell state. The output gate (o_t) decides what goes to output.

C.5 Building LSTM Model using Word Embeddings for Sequential News and Sequential Stock Data

Here 2 Recurrent Neural Network (RNN) models were required. First RNN Model to take news sequences along with their vector word embeddings as inputs and second RNN Model to take stock sequences as input. We used Keras Functional API [19] to combine both RNN models and identify the impact of both news and stock sequences on next day's stock trends. Keras Functional API provides a concatenate layer to combine different models. It also provides an advantage to get impacts of individual models as auxiliary outputs along with combined impact as the main output. Here impact means the classification of next day's stock trends. Fig. 16. is the architecture for the model used.

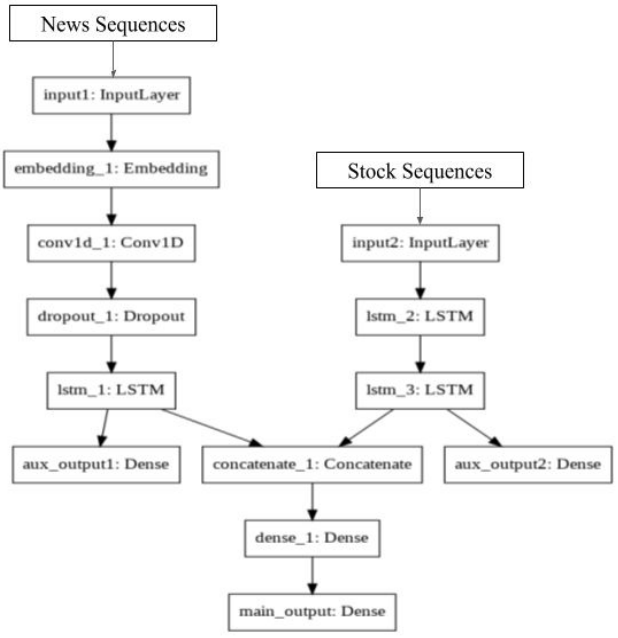

Fig. 16. RNN Model Architecture to predict next day's stock trend using news and stock sequences

Embedding layer takes pre-trained GloVe vector embeddings and news sequences (from input layer) as inputs. Convolutional 1 dimensional layer is used for feature extraction and helps to identify local patterns in news sequences. LSTM layer is used to learn from sequential data. Concatenate layer is used to combine multiple models. Aux_output1 gives classification of next day's stock trend from news sequences and aux_output2

gives classification of next day's stock trend from stock sequences and main_output gives classification based on both sequences. Detailed model summary is given in Appendix 6.

Table 4. shows accuracy, precision and recall achieved on test dataset from different models. Fig. 17. shows ROC Curve for RNN with news sequences as input, Fig. 18. shows ROC Curve for RNN with stock sequences as input and Fig. 19. shows ROC Curve for Deep Neural Network.

Models	Accuracy	Precision	Recall
Model 1 using News Sequences	36.7%	36.7%	36.7%
Model 2 using Stock Sequences	32.4%	32.4%	32.4%
Model 1 and Model 2 Combined	40.0%	40.0%	40.0%

Table 4 Performance metrics on Test dataset
(News Sequences from Reddit News Dataset)

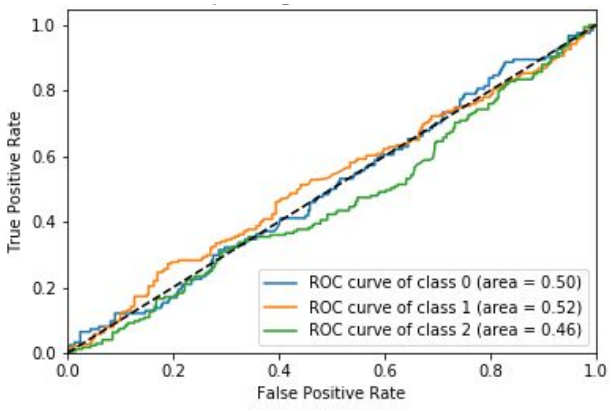

Fig. 17. ROC Curve for RNN Model using News Sequences

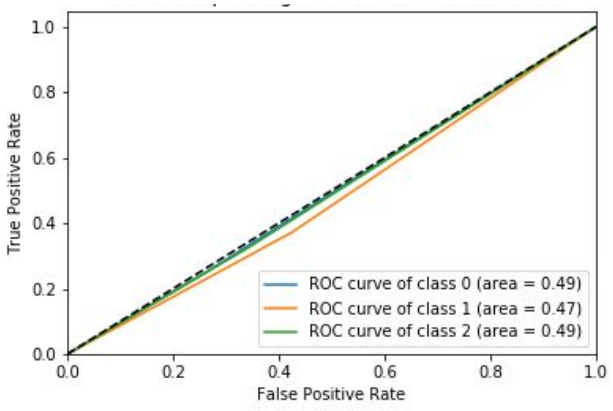

Fig. 18. ROC Curve for RNN Model using Stock Sequences

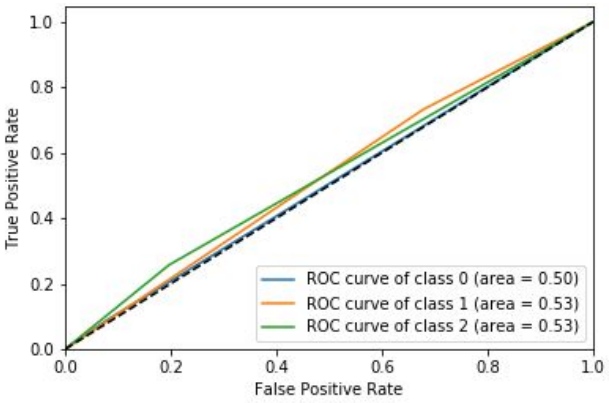

Fig. 19. ROC Curve for Combined RNN Model

We expected to get significant improvements in performance metrics using RNN models and Vector Word Embeddings but we were not able to achieve it. After scrutinizing the possible reasons for the poor performance of the RNN model, we identified that Reddit Dataset mainly consisted of political news and didn't have a noticeable impact on Google Stock Prices. Though using the RNN model with Reddit Dataset encoded by GloVe word embeddings gave the best classification of next day's stock trends till now.

D. Using UCI News Aggregator Dataset with Word Embeddings and LSTM

In order to achieve better performance metrics, we used the UCI News Aggregator Dataset and filtered the dataset to get the news headlines belonging to the technology category to understand the impact on Google stock trends.

UCI News Dataset consisted of multiple news (nearly 89000) under technology category for only 79 days. Combining multiple news for a single day would have limited total data for training and testing. Therefore, instead of combining all news for a day into a single sentence we calculated the impact of an individual news headline on next day's stock trend. We generated word vector embeddings and news sequences using similar steps mentioned under section C.2. We limited the length of single news sequences (with word indexes) to 16.
Only news sequences and pre-trained word embeddings from GloVe were used to train the RNN model with embedding and LSTM layers. The summary of the model used is given in Fig. 20.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 15, 300)	5935200
conv1d_2 (Conv1D)	(None, 15, 32)	28832
max_pooling1d_2 (MaxPooling1	(None, 7, 32)	0
lstm_2 (LSTM)	(None, 100)	53200
dense_2 (Dense)	(None, 3)	303

Total params: 6,017,535
Trainable params: 6,017,535
Non-trainable params: 0

Fig. 20. Model Summary of RNN with UCI dataset

Performance metrics using the above RNN Model is given in Table 5. Fig. 21. shows ROC Curve on test data.

Models	Accuracy	Precision	Recall
Model using News Sequences	73.5%	73.4%	73.5%

Table 5. Performance metrics on Test dataset (UCI Technology News)

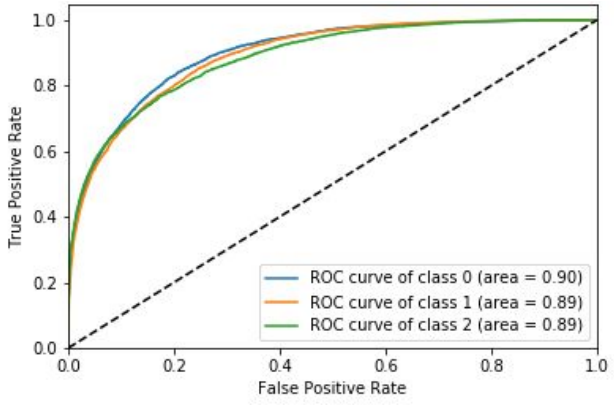

Fig. 21. ROC curve for RNN Model on UCI News Test Dataset

From Table 5 and Fig. 21, we can identify that using vector word embeddings as word encoding technique for textual data (news headlines) and LSTM layer to train on news sequences in RNN model can classify stock trends for next day with better accuracy, given an appropriate source of news for particular stock type (tech, finance, etc.)

IV. Conclusions

Performance metrics from different models trained and tested with Reddit News Dataset helped to identify that word vector embeddings is a better encoding technique to understand the essence of a sentence for classification problem. Also, on comparing performance metrics for all models trained and tested with Reddit News Dataset showed that RNN with word embedding and LSTM layers performed better as compared to conventional models. We also identified that type of news headlines that are used to classify stock trends is a crucial factor to identify stock trends. In our case, we used stock data of Google which is a technology-based company. Using UCI News Dataset and further filtering technology-based news to train and test RNN model gave significant improvement on performance metrics.

In our perspective, there is a large future scope for the project. News Headlines for longer date ranges and specific to companies should be incorporated to train and test RNN models for better predictions of stock trends for that company. Sequences of Stock Data with engineered features can be added along with news sequences to make the model more robust and accurate.

V. References

[1] https://nlp.stanford.edu/pubs/lrec2014-stock.pdf - On the Importance of Text Analysis for Stock Price Prediction

[2] http://ijecm.co.uk/wp-content/uploads/2016/06/4614.pdf - forecasting stock market trends by logistic regression and neural networks

[3] https://medium.com/@nilimeshhalder/multi-class-classification-using-decision-tree-random-forest-and-extra-trees-algorithm-in-python-ec2428d576ec - Multi-class Classification using Decision Tree, Random Forest and Extra Trees Algorithm in Python: An End-To-End Data Science Recipe — 016

[4] https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/ - Recurrent neural networks and LSTM tutorial in Python and TensorFlow

[5] https://www.kaggle.com/aaron7sun/stocknews - Daily News for Stock Market Prediction

[6] https://www.kaggle.com/uciml/news-aggregator-dataset - News Aggregator Dataset

[7] https://pandas-datareader.readthedocs.io/en/latest - *pandas_datareader* library

[8] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 - Understanding AUC - ROC Cruve

[9] https://machinelearningmastery.com/gentle-introduction-bag-words-model/-A Gentle Introduction to the Bag-of-Words Model

[10] https://pythonspot.com/nltk-stop-words/ - NLTK Stop words

[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html-sklearn CountVectorizer

[12] https://machinelearningmastery.com/what-are-word-embeddings/ - What are word Embeddings for text

[13] https://nlp.stanford.edu/projects/glove/ - GloVe Vector Embeddings
[14] https://www.ibm.com/blogs/research/2018/11/word-movers-embedding/
[15] https://austingwalters.com/classify-sentences-via-a-recurrent-neural-network-lstm/ - Classify sentences using Recurrent Neural Network
[16] https://www.youtube.com/watch?v=6niqTuYFZLQ - Recurrent Neural Networks
[17] http://colah.github.io/posts/2015-08-Understanding-LSTMs/ - Understanding LSTM Networks
[18] https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57
[19] https://keras.io/models/model/ - Keras Functional API

VI. Appendix

Appendix 1

Presentation:
https://github.com/smitkiri/stock-trend-prediction/blob/master/presentation/Predicting%20Stock%20Market%20Trend%20using%20Daily%20News%20Headlines.pptx

Code: https://github.com/smitkiri/stock-trend-prediction

Appendix 2

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 2048)	167936
dense_20 (Dense)	(None, 2048)	4196352
dropout_9 (Dropout)	(None, 2048)	0
dense_21 (Dense)	(None, 1024)	2098176
dense_22 (Dense)	(None, 1024)	1049600
dropout_10 (Dropout)	(None, 1024)	0
dense_23 (Dense)	(None, 512)	524800
dense_24 (Dense)	(None, 512)	262656
dropout_11 (Dropout)	(None, 512)	0
dense_25 (Dense)	(None, 256)	131328
dense_26 (Dense)	(None, 256)	65792
dropout_12 (Dropout)	(None, 256)	0
dense_27 (Dense)	(None, 3)	771

Total params: 8,497,411
Trainable params: 8,497,411
Non-trainable params: 0

Model Summary for Deep Neural Network under Conventional Approach

Appendix 3

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 512)	4096512
dropout_4 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 256)	131328
dropout_5 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 64)	16448
dropout_6 (Dropout)	(None, 64)	0
dense_8 (Dense)	(None, 3)	195

Total params: 4,244,483
Trainable params: 4,244,483
Non-trainable params: 0

Model Summary for Deep Neural Network under Bag-of-Words Approach

Appendix 4

	Date	news
0	2008-08-18	tornado throws bus poland captured one passengers
1	2008-08-18	great resource war already underway mainly mid...
2	2008-08-18	britain terror laws left family shattered
3	2008-08-18	president says resigning avoid impeachment bat...
4	2008-08-18	190000 us contractors working iraq

	Date	news_words_indexes
0	2008-08-18	[0, 1, 2, 3, 4, 5, 6]
1	2008-08-18	[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, ...
2	2008-08-18	[26, 25, 27, 28, 29, 30]
3	2008-08-18	[31, 32, 33, 34, 35, 36, 37, 38, 31791, 39]
4	2008-08-18	[31791, 40, 41, 42, 43]

Textual sentences converted to sequence of word indexes

Appendix 5

Date	High	Low	Open	Close	Volume	...	month_9	month_10	month_11	month_12	change_1
2008-08-08	246.949371	236.956833	239.178497	246.580750	7506500.0	...	0	0	0	0	1
2008-08-11	253.489853	244.971786	245.315491	249.484863	8510300.0	...	0	0	0	0	1
2008-08-12	252.119995	248.070175	250.062698	250.366562	5532000.0	...	0	0	0	0	0
2008-08-13	250.829834	246.017868	249.863449	249.081375	7278100.0	...	0	0	0	0	1
2008-08-14	252.857224	247.218369	247.920731	251.801193	5859000.0	...	0	0	0	0	1
2008-08-15	254.376541	251.806168	252.548386	254.122482	7117800.0	...	0	0	0	0	2
2008-08-18	254.047760	246.829819	253.968063	248.219604	6692700.0	...	0	0	0	0	2
2008-08-19	248.209656	242.406403	244.299301	244.334183	6115700.0	...	0	0	0	0	2
2008-08-20	247.417618	240.383987	246.436295	241.594452	7993900.0	...	0	0	0	0	2
2008-08-21	244.035294	238.740143	240.558334	242.356583	7054500.0	...	0	0	0	0	1

Example of stock sequences where 7 days' stock data is used to predict the label

Appendix 6

Layer (type)	Output Shape	Param #	Connected to
input1 (InputLayer)	(None, 308)	0	
embedding_1 (Embedding)	(None, 308, 300)	9537900	input1[0][0]
conv1d_1 (Conv1D)	(None, 308, 3)	3603	embedding_1[0][0]
input2 (InputLayer)	(None, 7, 81)	0	
dropout_1 (Dropout)	(None, 308, 3)	0	conv1d_1[0][0]
lstm_2 (LSTM)	(None, 7, 256)	346112	input2[0][0]
lstm_1 (LSTM)	(None, 128)	67584	dropout_1[0][0]
lstm_3 (LSTM)	(None, 128)	197120	lstm_2[0][0]
concatenate_1 (Concatenate)	(None, 256)	0	lstm_1[0][0] lstm_3[0][0]
dense_1 (Dense)	(None, 64)	16448	concatenate_1[0][0]
aux_output1 (Dense)	(None, 3)	387	lstm_1[0][0]
aux_output2 (Dense)	(None, 3)	387	lstm_3[0][0]
main_output (Dense)	(None, 3)	195	dense_1[0][0]

Total params: 10,169,736
Trainable params: 10,169,736
Non-trainable params: 0

Model summary for neural network using Keras Functional API to combine two different RNN models