Chapter 1

Introduction

1.1 Introduction

Stock prices are very fluctuating in nature. They depend on various factors like the previous stock prices, current market trends, financial news, competitor's performance etc. The Efficient Market Hypothesis (EMH) states that all relevant information is fully and immediately reflected into stock prices, hence the price fluctuations are unpredictable. Many researchers have used Machine Learning techniques to better predict the stock prices, however most of the research is about the international market which is comparatively much stable as compared to the Indian stock market.

Figure 1 compares Bombay Stock Exchange with Standard & Poor's 500, NASDAQ and DOW.

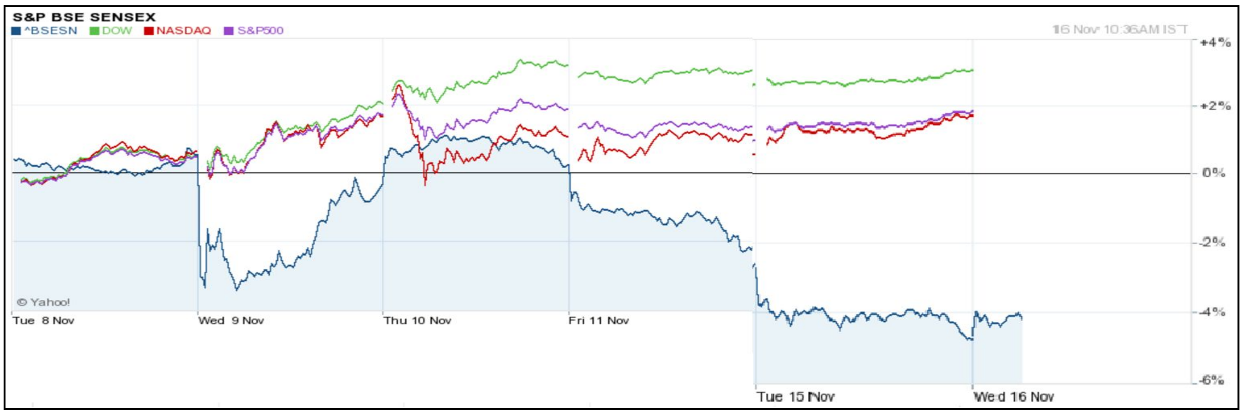


Figure 1.1

Comparison of stock indices

This leads to the question: How can Machine Learning be used to better predict such fluctuating markets like the Indian stock market?

Among the sectors: I.T., Financial, FMCG, Banking, Pharma and Media, this project focuses on I.T. sector.

1.2 Problem Statement

In this project, we proposed to predict the stock prices of some of the I.T. companies which are listed in National Stock Exchange (NSE), Bombay Stock Exchange (BSE) by correlating the factors like financial news, expert's opinion and public sentiment from Twitter with historical stock market data and international market trend using Machine Learning techniques.

1.3 Objectives

- To gather data about stock prices, financial events, and public sentiments from various data sources and build a predictive model.
- Use this model to predict the stock prices of some companies in the Indian I.T. domain.

Chapter 2

Literature Survey

We studied various papers describing the research done in this domain and also the comparison of various models. This chapters summaries some of those research papers.

2.1 Papers

2.1.1 Sentiment Analysis of Twitter Data for Stock Market Prediction[1]

Rupawari Jadhav, M.S. Wakode discussed the prediction of future stock prices with the help of sentiment score. The paper presents two different textual representations, Word2vec and N-gram, for analyzing the public sentiments in tweets. The author applied sentiment analysis and supervised machine learning principles (such as logistic regression, random forest, Sequential Minimal Optimization) to tweets extracted from Twitter and analyzing the correlation between stock market movement of company and sentiments in tweets. It proposes a hybrid approach which combines unsupervised learning to cluster the tweets and then performing supervised learning methods for Classification. The author applied different machine learning techniques: Naive Bayes(NB), Maximum entropy, support vector machine (SVM) etc. and concluded that NB, SVM with 89.4% accuracy outperforms the other techniques in sentiment classification. They looked for a correlation between twitter sentiments with stock prices and determined which words in tweets correlate to change in stock price by doing a post analysis of price change and tweets.

2.1.2 Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation[2]

Paul D. Yoo, Maria H. Kim and Tony Jan compared and evaluated some of the existing ML techniques used for stock market prediction. After comparing simple regression, multivariate regression, Neural Networks, Support Vector Machines and Case Based Reasoning models they concluded that Neural Networks offer ability to predict market directions more accurately as compared to other techniques. Support Vector Machines and Case Based Reasoning are also popular for stock market prediction. In addition, they found that incorporating event information with prediction model plays a very important role for more accurate prediction. The web provides the latest and latent event information about stock market which is required to yield higher prediction accuracy and to make prediction in a short time frame.

2.1.3 Stock Price Prediction Using Financial News Articles[3]

M.I. Yasef Kaya and M. Elef Karshgil analysed the correlation between the contents of financial news articles and the stock prices. News articles were labeled positive or negative depending on their effect on stock market. Instead of using single word as features, they used word couples as features. A word couple consisted of a combination of a noun and a verb. SVM classifier was trained with labeled articles to predict the stock prices.

2.1.4 Forecasting of Stock Market Indices Using Artificial Neural Network[4]

Dr. Jay Joshi, Nisarg A Joshi used artificial neural network (ANN) to forecast the daily returns of Bombay Stock Exchange (BSE) Sensitive Index (Sensex) because of its ability to discover non-linear relationship in the input dataset without a prior assumption of the knowledge of the input and the output. They compared the performance of the neural network with performances of random walk and linear autoregressive models. They reported that neural network outperforms linear autoregressive and random walk models by all performance measures in both in-sample and out-of-sample forecasting of daily BSE Sensex returns.The model forecasted the desired target with an average accuracy of 82%.

2.1.5 Stock Price prediction using ARIMA model[5]

Ayodele A. Adebiyi, Aderemi O. Adewumi and Charles K. Ayo used ARIMA model to predict the stock price on the data obtained from New York Stock Exchange (NYSE) and Nigeria Stock Exchange (NSE).

The data composed of four elements, namely: open price, low price, high price and close price respectively. In this research the closing price is chosen to represent the price of the index to be predicted. Closing price is chosen because it reflects all the activities of the index in a trading day.

Used Q-statistics and correlogram methods to show that there is no significant pattern left in the autocorrelation functions (ACFs) and partial autocorrelation functions (PACFs). If the data series is not stationary, it was made stationary with the help of differencing techniques. Results obtained revealed that the ARIMA model has a strong potential for short-term prediction and can compete favourably with existing techniques for stock price prediction.

2.2 Techniques studied

2.2.1 Autoregressive Model (AR)

The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term (an imperfectly predictable term); thus the model is in the form of a stochastic difference equation. The notation $AR(p)$ indicates an autoregressive model of order p. It is defined as:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t$$

Where $\phi_1, ..., \phi_i$ are the parameters of the model, c is constant, and ε_t is white noise.

2.2.2 Moving Average Model (MA)[5]

The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic (imperfectly predictable) term.

A moving average term in a time series model is a past error (multiplied by a coefficient).

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q}$$

where μ is mean of the series, $\theta_1, ..., \theta_q$ are the parameters of the model and the $\varepsilon_t, \varepsilon_{t-1}, ..., \varepsilon_{t-q}$ are white noise error terms.

2.2.3 Autoregressive Integrated Moving Average (ARIMA)[5]

In this model, non-stationary data can be made stationary by differencing the series, Y_t.

The general model for Y_t is written as,

$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} \ldots \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots \theta_q \epsilon_{t-q}$

Where, Y_t is the differenced time series value, ϕ and θ are unknown parameters and ϵ are independent identically distributed error terms with zero mean. Here, Y_t is expressed in terms of its past values and the current and past values of error terms.

The ARIMA model combines three basic methods:

- Autoregressive (AR) – In auto-regressive model the values of a given time series data are regressed on their own lagged values, which is indicated by the "p" value in the model.
- Differencing (I-for Integrated) – This involves differencing the time series data to remove the trend and convert a non-stationary time series to a stationary one. This is indicated by the "d" value in the model. If d = 1, it looks at the difference between two time series entries, if d = 2 it looks at the differences of the differences obtained at d =1, and so forth.
- Moving Average (MA) – The moving average nature of the model is represented by the "q" value which is the number of lagged values of the error term.

2.2.4 Artificial Neural Network (ANN)[2][4]

An (artificial) neural network is a network of simple elements called neurons, which receive input, change their internal state (activation) according to that input, and produce output depending on the input and activation. The network forms by connecting the output of certain neurons to the input of other neurons forming a directed, weighted graph. The weights as well as the functions that compute the activation can be modified by a process called learning which is governed by a learning rule.

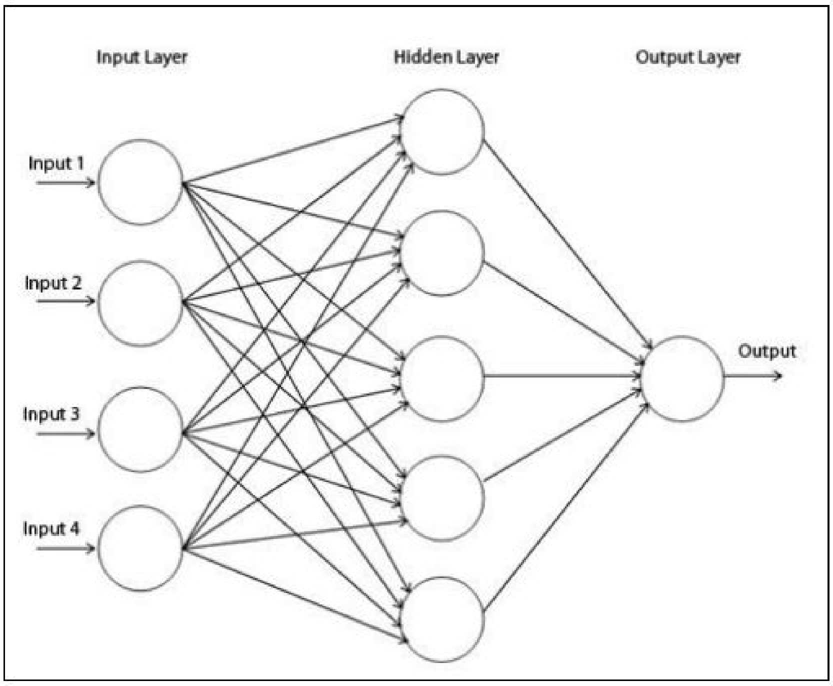


Figure 2.1

Artificial Neural Network

2.2.4 Long Short Term Memory (LSTM)

LSTM is form of Recurrent Neural Network(RNN) which is capable of holding long term dependencies. LSTM can remember the information for long period of time.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

- Cell: It is used to remember the values over arbitrary time intervals.
- Input Gate: It decides which information to keep in the cell.
- Output Gate: It is used to decide which part of cell state should be given as an output.
- Forget Gate: It is used to decide which information to throw away from the cell.

2.2.5 Convolutional Neural Network (CNN)[7]

A convolutional neural network (CNN) is a class of deep, feed-forward networks, composed of one or more convolutional layers with fully connected layers (matching those in typical ANNs) on top. It uses tied weights and pooling layers.

Chapter 3

Requirements and Analysis

This chapter describes the various requirements for the project along with the analysis performed.

3.1 Requirements

3.1.1 Functional Requirements

- The system should be able to gather data about historical stock prices, financial news, public sentiment and expert's opinions from various sources.
- The system should be able to predict the stock price of selected companies using the data gathered.

3.1.2 Nonfunctional Requirements

- Maintainability: The maintenance of the system would require training of the software by recent data so that there commendations are up to date. The database has to be updated with recent values.
- Scalability: As the amount of data increases, the system should be able to handle such huge amount of data.
- Reliability: The reliability of the system is dependent on the reliability of the data sources. Thus, data from various sources should be correct.

3.2.2 Software Requirements

- Python 3: Python is an interpreted high-level programming language for general-purpose programming.
- Keras: Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
- Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.
- NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- NLTK: The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language
- Gensim: Gensim is a robust open-source vector space modeling and topic modeling toolkit implemented in Python. It uses NumPy, SciPy and optionally Cython for performance.
- Tweepy: An easy-to-use Python library for accessing the Twitter API.
- Statsmodels: Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

3.2 Analysis

3.2.1 Selecting the prediction model

Based on the evaluation done by Paul D. Yoo, Maria H. Kim and Tony Jan comparing the various models including simple regression, multivariate regression, Support Vector Machines, Artificial Neural Networks, Case Based Reasoning it was concluded that Artificial Neural Networks can achieve higher accuracy as compared to other models. The ability of

ANNs to learn nonlinear relationships from the training input/output pairs enables them to model nonlinear dynamic systems such as stock markets more precisely.[2] Moreover, combining qualitative factors like international events, financial news, Twitter sentiments, etc. can achieve much higher accuracy.

3.2.2 Analysis of LSTM

Since there is a trend in a stock market, Neural Network has to consider the previous day's stock prices. Simple Neural Network is not able to remember the previous values. Thus, Long Short Term Memory (LSTM) Network should be used.

3.2.3 Analysis of CNN

CNNs are better adaptive in recognizing the hidden and different feature combination pattern than compared to other neural models. The same property makes it more useful in Image processing. And our project does require the recognition of frequently words combination pattern. Thus, we made use of CNN layer in our proposed model.

Chapter 4

Approach Taken

In this chapter we present the basic architecture of our model and complete description about the dataset and the preprocessing steps that were undertaken.

4.1 Basic Architecture

4.1.1 Overall data flow

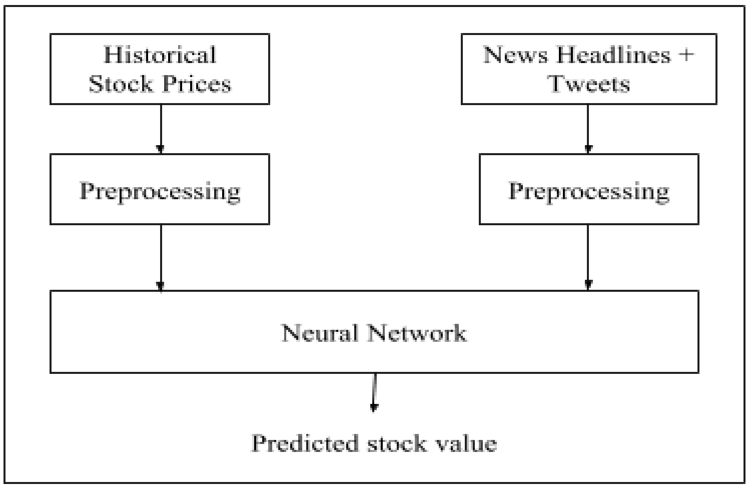


Figure 4.1

Data Flow Diagram

- Historical Stock Prices: This is the time series data which includes the stock prices of the companies. This data had to be pre-processed in order to handle missing values in the time series and remove all invalid data.
- Financial News: This includes all latest events in the financial market. The news headlines were preprocessed and converted to multidimensional vectors suitable for input to the neural network.
- Neural Network: The Artificial Neural Network is trained to find the correlation between all the factors and predict the stock price based on the input.

4.2 Dataset

Three different types of datasets have been used in this project for 2 different stocks (TCS and Infosys) each:

- Historical Stock Data: Daily price data from Yahoo Finance.
  Attributes: date, opening price, closing price, maximum, minimum, shares traded
- News Headlines:
  - News Headlines scraped using BeautifulSoup in Python from http://www.moneycontrol.com/ and https://stockarchitect.com/.
  - News Headlines collected over the last 2 years from Indian business, finance, and market news websites aggregated by Pulse.

  Attributes: timestamp, headline
- Twitter Data: Company specific tweets were collected using Tweepy.
  Attributes: timestamp, username, tweet, retweet count

Historical Stock Data and News Headlines data include records from January 01, 2009 till February 25, 2018. Twitter Data was collected from September 01, 2017 till February 25, 2018. Since Twitter search API allows to collect tweets published in past 7 days, historical tweets were not available.
Keywords used for collecting tweets: TCS, Infosys, Natarajan Chandrasekaran, Tata, Vishal Sikka, Sensex, Nifty.

4.3 Preprocessing Of Dataset

4.3.1 Historical Stock Price Data

Historical Stock Prices needed to be normalized before directly applying the values to the model since the parameters used were of different scales.

Two normalization techniques were used:

- Min-max scalar

  This converts the prices in the range 0 to 1.

  Normalized Value = (Current Value - Min_Value) / (Max_Value - Min_Value)

  where Min_value, Max_value are the minimum and maximum value of the data to be normalized.

  This technique does not retain the information about prices increasing or decreasing as compared to the previous day.

- Percentage change from previous value

  This converts the prices in the range -1 to 1.

  Normalized value = (Current Value - Previous Value) / Previous Value

  After normalization, negative values indicate decrease in stock price and positive values indicate increase in stock price.

Change in percentage as input gave better results than min-max scalar.

4.3.2 News Headlines and Twitter Data

Preprocessing of financial headlines and twitter data was essential to remove the noises such as stopwords, redundant words and punctuations as they are not important for financial analysis.

- Punctuations and characters were removed, contractions were replaced by their original form by using a dictionary (eg. “ain’t” replaced by “am not”, “can’t” replaced by “cannot”).

  Link to the dictionary:

  http://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python

- News headlines and tweets were tokenized and stopwords were removed using the NLTK’s stopword corpora.

- To convert the inflectional forms of words to the base form stemming was performed. For example: “banking” converted to “bank”.
- Common emojis were replaced by their signature words with the help of Emoji Sentiment file.

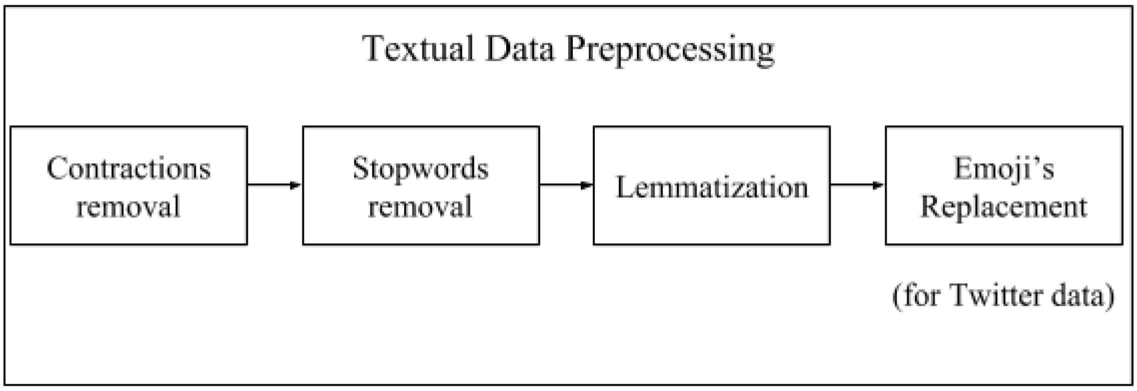


Figure 4.2

Textual Data Preprocessing

Chapter 5

Proposed Models

This chapter describes the different proposed models.

5.1 Model 1 (Historical stock prices)

Since the stock prices mainly depend on the previous stock prices, autoregressive models can be used to predict the stock prices given previous stock prices, and Long Short Term Memory Networks (LSTM) can be used to find long term dependencies in the stock prices.

5.1.1 Autoregressive Model

The model takes closing stock prices of all days and finds the relation between them and also the best values for parameters of the AR model.

5.1.2 Seasonal Autoregressive Integrated Moving Average

Seasonality in a time series is a regular pattern of changes that repeats over S time periods, where S defines the number of time periods until the pattern repeats again. In a seasonal ARIMA model, seasonal AR and MA terms predict x_t using data values and errors at times with lags that are multiples of S (the span of the seasonality).

Seasonal ARIMA models are usually denoted ARIMA(p,d,q)(P,D,Q)$_m$, where m refers to the number of periods in each season, and the uppercase P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

Multiple values of p,d,q and P,D,Q were compared using Akaike information criterion (AIC) to find the best values of these parameters, the one with minimum AIC was chosen.

5.1.3 Long Short Term Memory Network

Input: Given models took previous day's stock prices with opening, maximum, minimum and last trading price as input parameters. Input was in the form of percentage change.
Output: Output of the model was percentage change in the closing price of the next day's stock.
LSTM network consisted of 4 input neurons with each neuron representing a single parameter from the input dataset. A lookback value was set for training the model. This value represents the number of rows which should be looked back for predicting the next value. i.e. if lookback value is set to 3, then model should consider the stock prices of previous 3 days to predict the next day's price. The model was trained on various lookback values. Hyperbolic tangent was used as activation function at the output layer.

5.2 Model 2 (Historical stock prices and News headlines)

Along with the historical prices, financial news has some impact on the stock prices. In this model, we considered financial news headlines and historical prices to predict the stock price.
Input: News headlines were converted into the word vectors of size 300. These word vectors along with 4 historical prices parameters (opening, maximum, minimum, last trading price) were given as an input to the model.
Preprocessed news headlines were converted into word vectors by using Google's pre-trained word embeddings. Average of the word vectors was taken to form sentence vector of a news headline.
Output: Output of the model was percentage change in the closing price of the next day's stock.

5.2.1 Long Short Term Memory Network

In this model, a single layer LSTM which takes word vectors of news headlines and historical prices as an input was used. The LSTM layer consisted of 304 neurons in the input layer. 300 neurons represented word vectors and 4 neurons represented historical prices parameters. This model was trained on various lookback values. Activation function used for the output layer was hyperbolic tangent.

5.2.2 Merged Layer

In this merged model, two different LSTM layers were merged together to give the output.
One LSTM layer took historical prices in the input layer. A lookback value for this layer of LSTM was set to 15.
Another LSTM layer took word vectors of news headlines in the input layer. A lookback value for this layer of LSTM was set to 30.

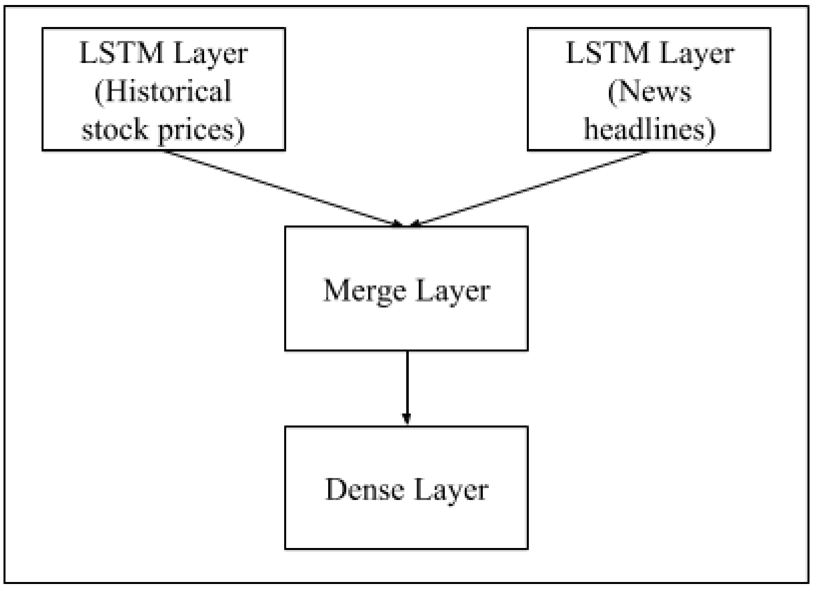

LSTM Layer (Historical stock prices)
LSTM Layer (News headlines)
Merge Layer
Dense Layer

Figure 5.1

Merge Layer

5.3 Model 3 (Combined CNN and LSTM)

Grabbing the sentiments from short texts is very challenging as it contains very limited contextual information and also the information is semantically sparse. Thus we made use of CNN which is good for extracting local features and the output of the same is passed on to the LSTM which is able to learn and remember the long-term dependencies. After preprocessing, we mapped the change in opening prices with the previous days news headlines. All the headlines of the same date were taken and embedded using the Glove words. To create the weights that it is to be used for the model's embeddings, we created a matrix consisting of the embeddings relating to the words in our vocabulary. If a word is found in GloVe's vocabulary, then we used it as it is in the pre-trained vector, otherwise given a random embedding for it which later updated as the model trained so the new 'random' embeddings were more accurate by the end of training. A fixed size of 300 for embedding dimensions was used to match the actual Glove Vectors. If the output dimensions is less than 300 then padding was applied to the headlines embeddings.

Link to Glove Dictionary File: https://nlp.stanford.edu/projects/glove/

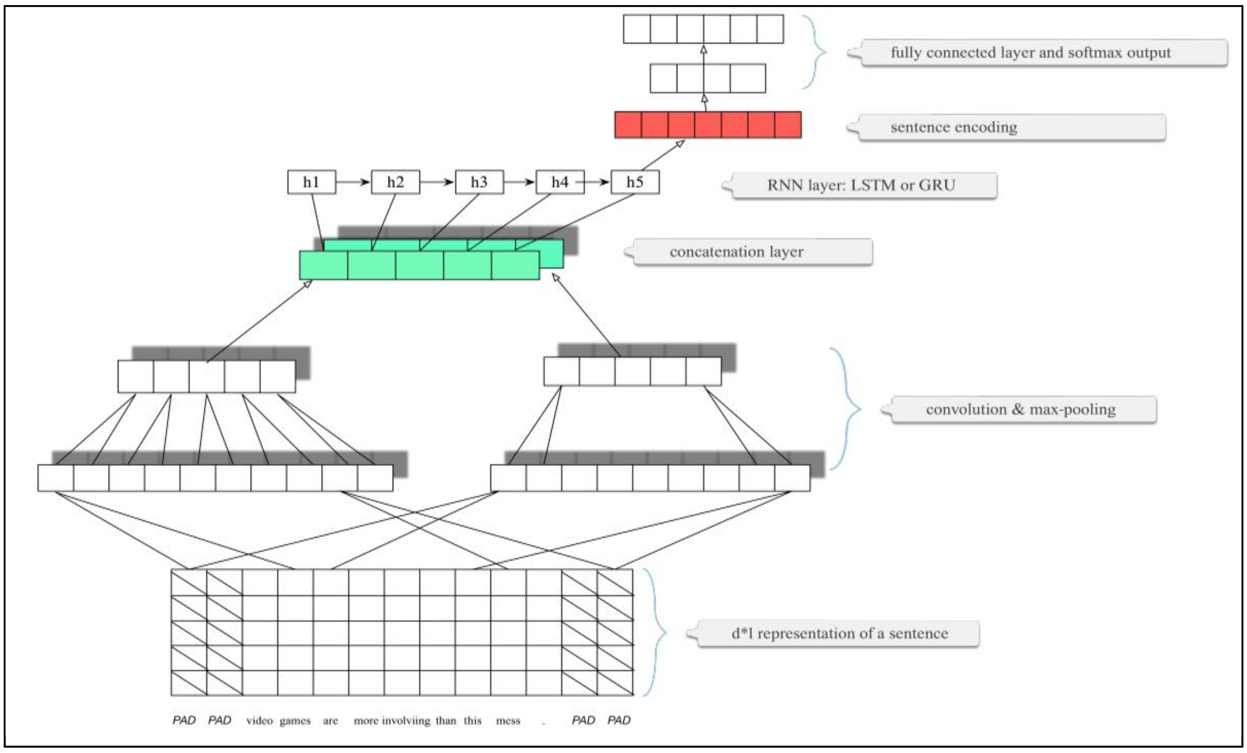

fully connected layer and softmax output
sentence encoding
h1
h2
h3
h4
h5
RNN layer: LSTM or GRU
concatenation layer
convolution & max-pooling
d*l representation of a sentence
PAD PAD video games are more involving than this mess . PAD PAD

Figure 5.2

Combined CNN and LSTM Model Architecture

Thus finally we had a matrix $\mathbf{Mat}_{r\times 300}$ where r is the no of rows in the dataset.And each row **S=[C_1 C_2 C_3 ......]** is the Sentence level representation where the column vector $\mathbf{C_i}$ corresponds to the embedding of the **i**th word in the sentence. The weighted matrix was then passed to the Convolution layer which applies a matrix-vector operation to each window of the sentence level representation with RELU (Rectified Linear Unit) activation function and some bias being added to get a feature mapping to extract the advanced n-gram feature vectors after applying max-pooling operation with multiple filter lengths. We used max-pooling as it gave better result than compared to average pooling. This feature vectors are given as input to the LSTM which maps the change in opening price and the vectors and tries to find the correlation between them. The model uses Adam Optimization Algorithm to update the network weights while training and reduces the mean squared error.

Chapter 6

Results and Evaluation

This chapter presents the evaluation of the each model described above along with the final results and the website screenshots.

6.1 Evaluation

6.1.1 Evaluating AutoRegressive Model

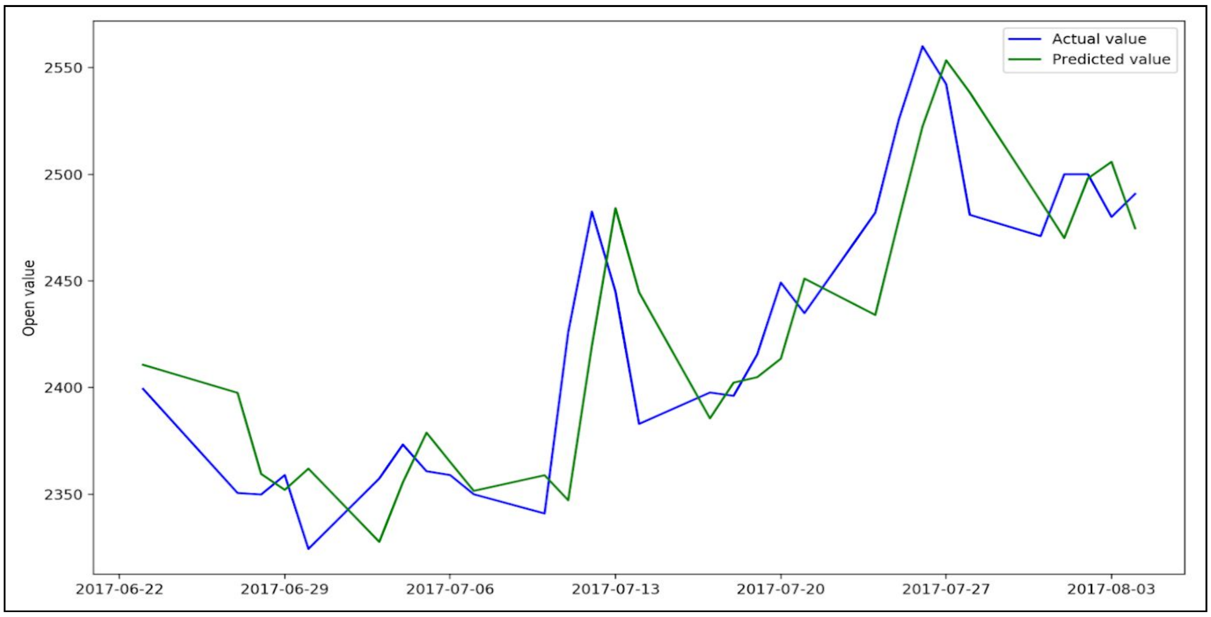


Figure 6.1

Prediction using AR Model for TCS

Figure 6.1 shows the differences between the actual stock prices and stock prices predicted by AR model for TCS. The RMS difference between the actual and predicted values is 34.13

6.1.2 Seasonal Autoregressive Integrated Moving Average Output

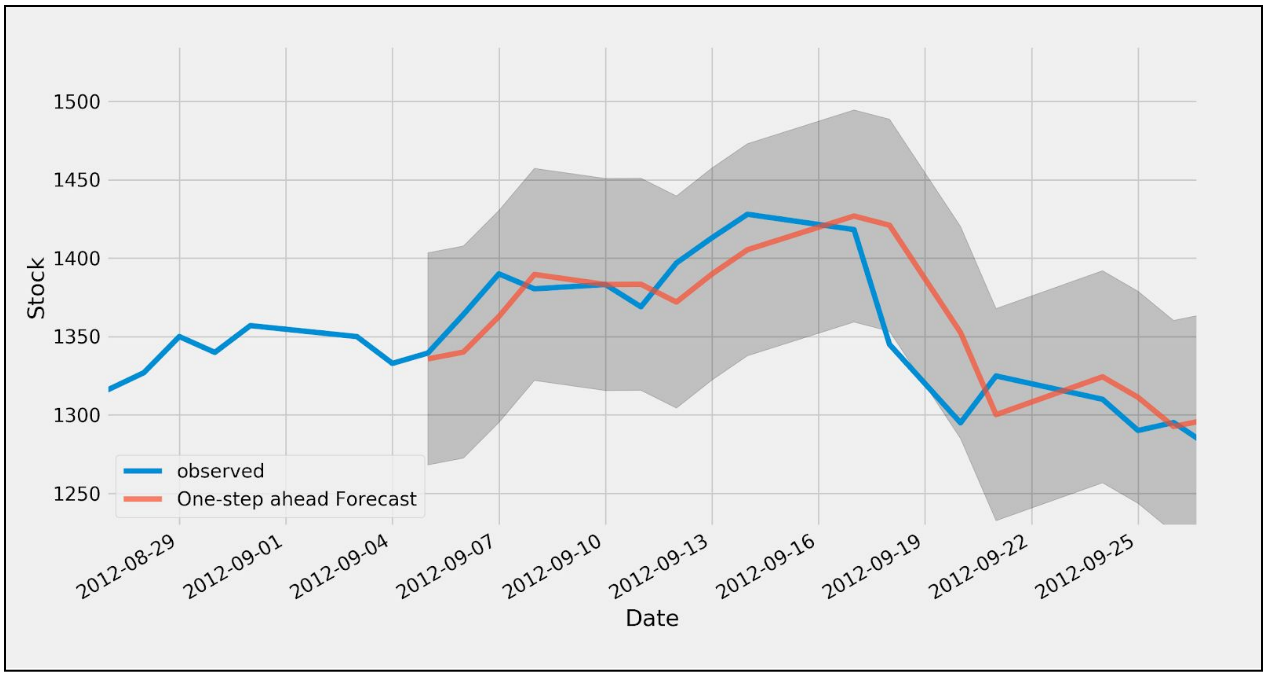


Figure 6.2

Prediction using Seasonal ARIMA for TCS

Figure 6.2 shows the actual stock prices and predicted stock prices using Seasonal ARIMA. The grey regions show the confidence interval of the predictions.

6.1.3 LSTM Model Output

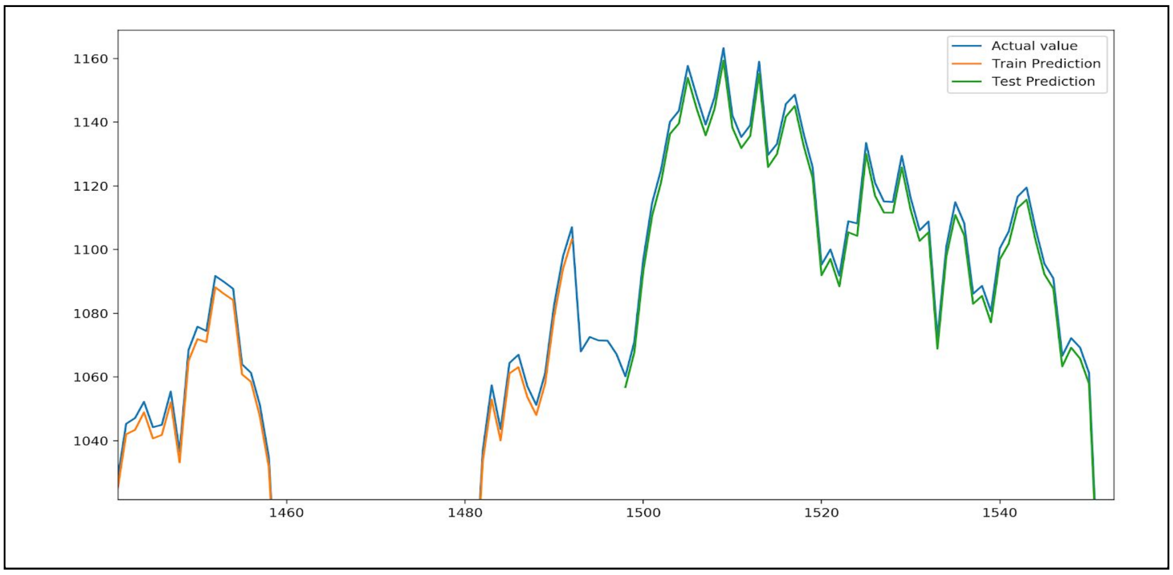


Figure 6.3

Prediction using LSTM for Infosys

Figure 6.3 shows the actual and predicted stock prices for Infosys on both training and testing data. The predictions are made using LSTM with a look back value of 5 days.

6.1.4 Combined CNN - LSTM Model Output

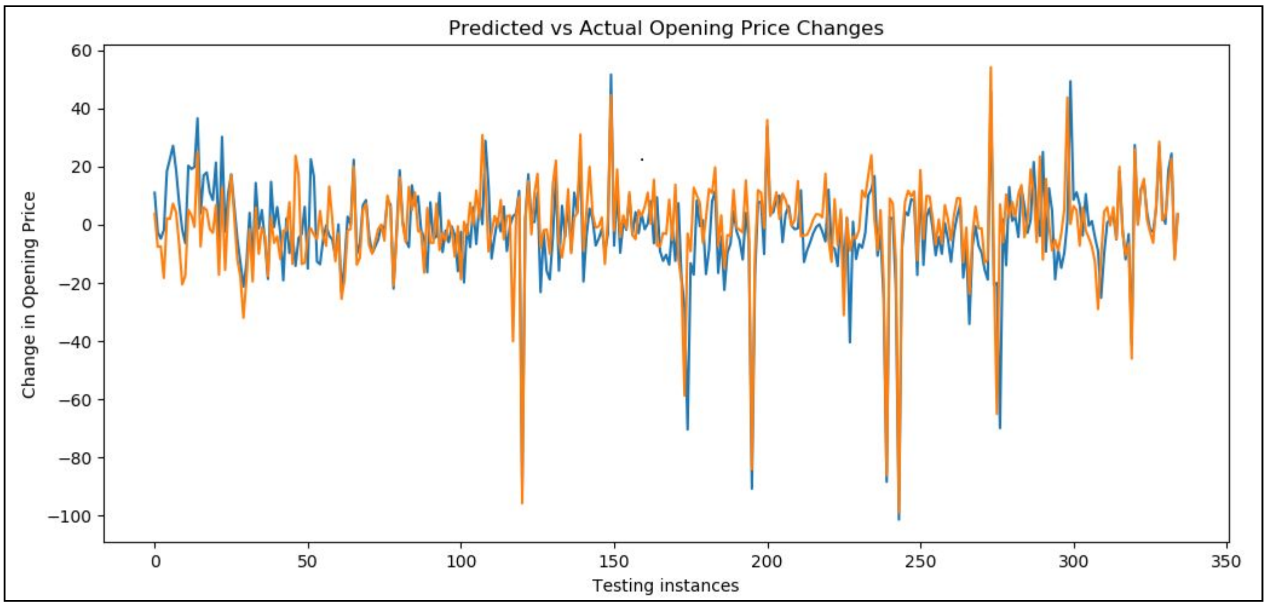
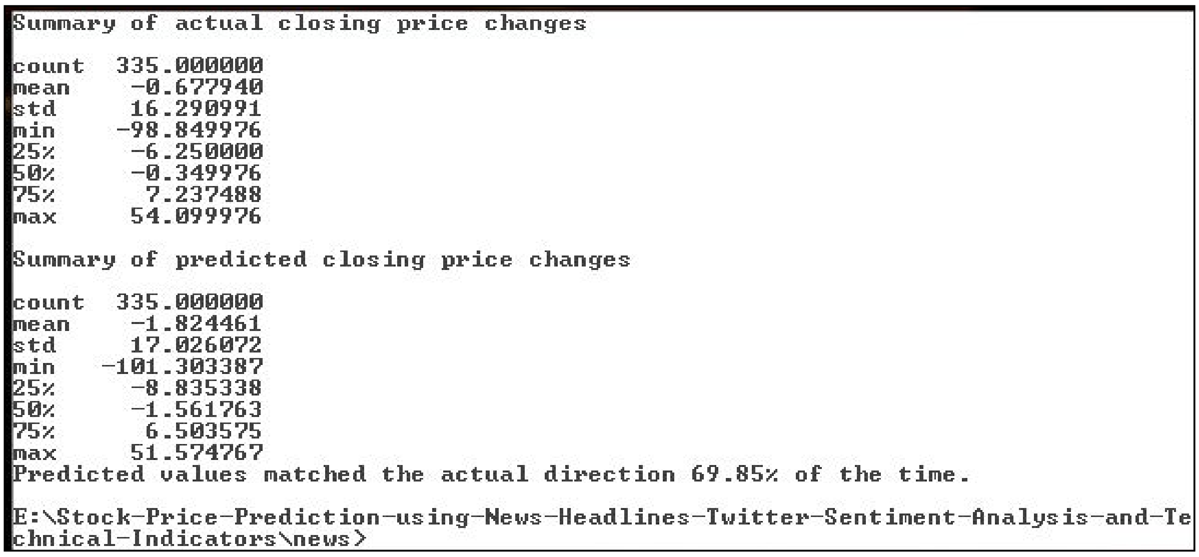


Figure 6.4

Combined CNN-LSTM Model output for Infosys

The output of this model seems to be much better than the above model as it maps the upwards and downward trends with more precision. However a sudden and a large change are not effectively marked up the model but that it is somewhat considerable.



```
Summary of actual closing price changes

count   335.000000
mean     -0.677940
std      16.290991
min     -98.849976
25%      -6.250000
50%      -0.349976
75%       7.237488
max      54.099976

Summary of predicted closing price changes

count   335.000000
mean     -1.824461
std      17.026072
min    -101.303387
25%      -8.835338
50%      -1.561763
75%       6.503575
max      51.574767
Predicted values matched the actual direction 69.85% of the time.

E:\Stock-Price-Prediction-using-News-Headlines-Twitter-Sentiment-Analysis-and-Te
chnical-Indicators\news>
```

Figure 6.5

CNN - LSTM model output on command prompt for Infosys

6.2 Results

Model No.	Model	Root Mean Square Error		Classification Accuracy	
		TCS	Infosys	TCS	Infosys
1.1	AR	34.13	13.70	48.17%	50.0%
1.2	Seasonal ARIMA	34.11	13.93	49.62%	49.34
1.3	LSTM (using historical stock prices)	34.91	16.08	50.56%	49.11%
2.1	LSTM (using news headlines and stock prices)	33.98	14.54	53.02%	53.79%
2.2	Merge Layer (LSTM + LSTM)	33.56	14.03	52.89%	52.57%
3	Combined CNN and LSTM	24.568	13.067	69.85%	74.86%

Table 6.1

Results of our Proposed Models with Other Methods

6.3 Code

Autoregressive model:

```
# Read input
filename = "stocks.csv"
file = open(data_directory + company + "/" + filename)

reader = csv.reader(file)
next(reader) # Skip headervalues = []
dates = []
for row in reader:
      values.append(float(row[1])) # Column 1 has closing price
      dates.append(datetime.strptime(row[0], "%Y-%m-%d")) # Column 
0 has dates

file.close()

model = AR(values, dates=dates, freq='D')a
predictions = model.fit().predict()

# Trim leading values used for first prediction
actual_values = values[len(values) - len(predictions):]
dates = dates[len(dates) - len(predictions):]

# Calculate Root mean squared error
print("Root mean squared error:", 
sqrt(mean_squared_error(actual_values, predictions)))
```

Seasonal Autoregressive Integrated Moving Average:

```
# Define the p, d and q parameters to take any value between 0 and 
2
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q 
triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in 
list(itertools.product(p, d, q))]
```

```
# Find best fitting model
for param in pdq:
      for param_seasonal in seasonal_pdq:
      try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                                 order=param,

seasonal_order=param_seasonal,

enforce_stationarity=False,

enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param,
param_seasonal, results.aic))
            if results.aic < minAIC:
                  p = param
                  sp = param_seasonal
                  minAIC = results.aic
      except:
            continue
```

Long Short Term Memory (using Historic Prices):

```
def create_dataset(dataset, input_columns, output_column,
look_back=1):
      dataX, dataY = [], []
      for i in range(len(dataset)-look_back):
      a = dataset[input_columns][i:(i+look_back)].values
      dataX.append(a)
      dataY.append(dataset[output_column][i + look_back])
      return numpy.array(dataX), numpy.array(dataY)

# Set which columns to use for input and predictions
input_columns = ['Open']
predict_column = 'Close'

# Split into train and test sets
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
```

```
train, test = dataset[0:train_size],
dataset[train_size:len(dataset)]
test.reset_index(drop=True, inplace=True)

# Reshape into X=t and Y=t+1
look_back = 5
trainX, trainY = create_dataset(train, input_columns,
predict_column, look_back)
testX, testY = create_dataset(test, input_columns, predict_column,
look_back)

model = Sequential()
model.add(LSTM(10, input_shape=(look_back, len(input_columns)),
activation='sigmoid'))
model.add(Dense(15, activation='sigmoid'))
model.add(Dense(5, activation='sigmoid'))
model.add(Dense(1, activation='tanh'))

loss_function = 'mean_squared_error'
optimizer = 'adam'
epochs = 20
batch_size = 10

model.compile(loss=loss_function, optimizer=optimizer)
model.fit(trainX, trainY, epochs=epochs, batch_size=batch_size,
verbose=2)
```

Long Short Term Memory (using news headlines and stock prices):

```
embeddings_binary = 'GoogleNews-vectors-negative300.bin'
embeddings_path = data_directory + embeddings_binary
word_vectors = KeyedVectors.load_word2vec_format(embeddings_path,
binary=True)

# Tokenization and punctuation removal
tokenized_news = []
for sentence in headlines:
      sentence = ''.join(c for c in sentence if c not in
punctuation)
      tokenized_news.append(nltk.word_tokenize(sentence))

# Sentence vector by taking average of word vectors
sentence_vector = np.empty((300))
```

```
input_vectors = np.empty((300))

for sentence in tokenized_news:
      for word in sentence:
      if word not in stopwords:
            if word in word_vectors.vocab:
                  sentence_vector = np.add(sentence_vector,
word_vectors[word])
      input_vectors = np.vstack((input_vectors, sentence_vector /
len(sentence)))

# Average of headlines_vectors with same date
news_vectors = headlines_vector.groupby('Date')
news_vectors = news_vectors.apply(lambda x: x.vectors.mean())
```

Combined CNN LSTM Model:

Preprocessing:

```
def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def clean_text(text, remove_stopwords = True):
'''Remove unwanted characters and format the text to create fewer
nulls word embeddings'''

    # Convert words to lower case
    text = text.lower()

    # Replace contractions with their longer forms
    if True:
        text = text.split()
        new_text = []
        for word in text:
            if word in contractions:
                new_text.append(contractions[word])
            else:
                new_text.append(word)
        text = " ".join(new_text)
    # Format words and remove unwanted characters
    text = re.sub(r'&amp;', '', text)
    text = re.sub(r'0,0', '00', text)
    text = re.sub(r'[_"\-;%()|.,+&=*%.,!?:#@\[\]]', ' ', text)
    text = re.sub(r'\'', ' ', text)
    text = re.sub(r'\$', ' $ ', text)
    text = re.sub(r'j k ', ' jk ', text)
    text = re.sub(r' s ', ' ', text)
    text = re.sub(r' yr ', ' year ', text)
    text = re.sub(r' l g b t ', ' lgbt ', text)

    # Optionally, remove stop words
    if remove_stopwords:
        text = text.split()
        stops = set(stopwords.words("english"))
        text = [w for w in text if not w in stops]
        text = " ".join(text)

    return text
```

Base Model Code :

```
if wider == True:
    nb_filter *= 2
    rnn_output_size *= 2
    hidden_dims *= 2

def build_model():

   model1 = Sequential()
   model1.add(Embedding(nb_words,
                        embedding_dim,
                        weights=[word_embedding_matrix],
                        input_length=max_daily_length))
   model1.add(Dropout(dropout)

    model1.add(Convolution1D(filters = nb_filter,
                             kernel_size = filter_length1,
                             padding = 'same',
                            activation = 'relu'))
    model1.add(Dropout(dropout))

    if deeper == True:
        model1.add(Convolution1D(filters = nb_filter,
                                 kernel_size = filter_length1,
                                 padding = 'same',
                                activation = 'relu'))
        model1.add(Dropout(dropout))

    model1.add(LSTM(rnn_output_size,
                     activation=None,
                 kernel_initializer=weights,
       dropout = dropout)

    model2 = Sequential()

    model2.add(Embedding(nb_words,
                          embedding_dim,
                          weights=[word_embedding_matrix],
                          input_length=max_daily_length))
   model2.add(Dropout(dropout)

    model2.add(Convolution1D(filters = nb_filter,
                             kernel_size = filter_length2,
                             padding = 'same',
                             activation = 'relu'))
    model2.add(Dropout(dropout))

    if deeper == True:
        model2.add(Convolution1D(filters = nb_filter,
                                 kernel_size = filter_length2,
                                 padding = 'same',
                                 activation = 'relu'))
        model2.add(Dropout(dropout)

    model2.add(LSTM(rnn_output_size,
                    activation=None,
                    kernel_initializer=weights,
                    dropout = dropout)

    model = Sequential()

    model.add(Merge([model1, model2], mode='concat'))
    model.add(Dense(hidden_dims, kernel_initializer=weights))
    model.add(Dropout(dropout))
    if deeper == True:
        model.add(Dense(hidden_dims//2,
kernel_initializer=weights))
        model.add(Dropout(dropout))

    model.add(Dense(1,
                    kernel_initializer = weights,
                    name='output'))

    model.compile(loss='mean_squared_error',
                  optimizer=Adam(lr=learning_rate,clipnorm=1.0))

    return model
```

Chapter 7

Conclusion

7.1 Conclusion

The project objective of adding news feeds, Twitter data to increase the prediction accuracy along with historical stock prices did give us better results compared to only historical stock prices. Along with this, using LSTM and CNN layers helped in mapping the events better. Though the root mean squared error value of our models were not substantially different from that of AR and ARIMA, but they had higher classification accuracy. Addition of Twitter data to the model didn't have a significant effect on the prediction value. This may be due to smaller dataset. Collecting more streaming tweets with the paid APIs and working with them might help in achieving better accuracy.

7.2 Future Scope

The future scope of the project would involve incorporating more stocks from different domains and finding out the correlation among them to predict the trend for an entire sector. Also, how much does one sector depend on another and incorporating that into our model. Making use of Twitter data in much better way with a large corpus. Looking out for some more features which can help us in getting more closer to the actual values.

Chapter 8

References

[1] Sentiment Analysis of Twitter Data for Stock Market Prediction https://www.ijarcce.com/upload/2017/march-17/IJARCCE%20129.pdf

[2] Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation http://ieeexplore.ieee.org/abstract/document/1631572/

[3] Stock Price Prediction Using Financial News Articles http://ieeexplore.ieee.org/document/5609404/

[4] Forecasting of Stock Market Indices Using Artificial Neural Network http://irjcjournals.org/ijieasr/Dec2015/2.pdf

[5] Stock market prediction using ARIMA model http://ijssst.info/Vol-15/No-4/data/4923a105.pdf

[6] Sentiment analysis of Twitter data for predicting stock market movements http://ieeexplore.ieee.org/document/7955659/

[7] Combination of Convolution and Recurrent Neural Networks for Sentiment Analysis of Short Texts https://www.aclweb.org/anthology/C/C16/C16-1229.pdf

[8] Prediction Model of the Stock Market Index Using Twitter Sentiment Analysis http://www.mecs-press.org/ijitcs/ijitcs-v8-n10/IJITCS-V8-N10-2.pdf

[9] Feature selection for stock market analysis. https://pdfs.semanticscholar.org/893d/0bdf90186ca2b6c7327813a240f2fd713ed5.pdf

[10] Correlation https://www.datascience.com/blog/introduction-to-correlation-learn-data-science-tutorials

[11] Feature selection methods and algorithms http://www.enggjournals.com/ijcse/doc/IJCSE11-03-05-051.pdf

[12] CNN and LSTM-based Claim classification in Online user Comments http://www.aclweb.org/anthology/C16-1258

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our final year project guide, Mrs. Asha Bharambe, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. We also appreciate the comments and advices given by the panels during our project presentations.

Abhinav Valecha

Anjali Wani

Rajeevkumar Yadav

Annexure

Published Paper:

Stock Price Prediction using News Headlines, Twitter Sentiment Analysis and Technical Indicators

Rajeevkumar Yadav
Student, VESIT
rajeevkumar.yadav@ves.ac.in

Abhinav Valecha
Student, VESIT
abhinav.valecha@ves.ac.in

Anjali Wani
Student, VESIT
anjali.wani@ves.ac.in

Asha Bharambe
Professor, VESIT
asha.bharambe@ves.ac.in

Abstract—Stock Prices are highly dependent on the behaviour of human investors and they build up their notion based on the stock specific news, public opinion on it and the various technical features of the stock. We have used the news headlines from different news sources, stock specific twitter data for getting the public sentiments and the historical stock prices to predict the stock price movements. In this paper, we present the jointed CNN (Convolutional Neural Network) and LSTM (Long Short Term Memory) architecture as it produced remarkable results with short texts data along with a separate LSTM network to capture the long term dependencies of the technical indicators like Simple, Open, High, Closing, 3 day Moving Averages, Volume, etc. We have been able to capture the trends and get a better accuracy by ensembling the above two models.

Index Terms — CNN, LSTM, Stock, Gradients, Hyperparameters, stopwords, Latent Dirichlet Allocation, Document Term Matrix, Gensim module, Activation functions (relu, elu), SentiWordNet Dictionary, Emoji Sentiment Dictionary, Ensembling.

1 Introduction

Stock prices are very fluctuating in nature and are influenced by a vast variety of sources. External as well as internal factors moves the stock market from something as small as a press release of a particular firm to something as big as laws passed by the government. The stock prices also changes with the fluctuations in the international market, impact on the sector also plays a key role. The Efficient Market Hypothesis (EMH) states that all relevant information is fully and immediately reflected into stock prices, hence the price fluctuations are unpredictable. This means that if we can incorporate various factors like the previous stock values, current national and international market trends, financial news impact, competitor's performance etc. in to our model then we can achieve a better insight of the changes. Many researchers have used Machine Learning techniques to better predict the stock values, however most of the research is about the international market which is comparatively much stable as compared to the Indian stock market. This leads to the question: How can we use Machine Learning and Big Data to better predict such fluctuating markets like the Indian stock market? Among the sectors: I.T., Financial, FMCG, Banking, Pharma and Media, for this project we are focusing on I.T. sector.

2 Literature Survey

2.1 Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation[1]

Paul D. Yoo, Maria H. Kim and Tony Jan compared and evaluated some of the existing ML techniques used for stock market prediction. After comparing simple regression, multivariate regression, Neural Networks, Support Vector Machines and Case Based Reasoning models they concluded that Neural Networks offer ability to predict market directions more accurately as compared to other techniques. Support Vector Machines and Case Based Reasoning are also popular for stock market prediction. In addition, they found that incorporating event information with prediction model plays a very important role for more accurate prediction. The web provides the latest and latent event information about stock market which is required to yield higher prediction accuracy and to make prediction in a short time frame.

2.2 Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts [2]

Xingyou Wang and the other co-authors used the jointed CNN and RNN architecture for extracting the sentiments of movie reviews. This architecture was inspired by the image classification problems more and paved a better model for analysis of short texts. The problem with this architecture was that it is difficult to RNN for capturing long term dependencies because the gradients tends to either vanish or explode.Thus in our architecture we solved this problem by using LSTM and to construct a better model we applied a grid search with the hyperparameter values.

2.3 Sentiment Analysis of Twitter Data for Stock Market Prediction[3]

Rupawari Jadhav, M.S. Wakode discussed two different textual representations, Word2vec and N-gram, for analyzing the public sentiments in tweets. The authors proposed a hybrid approach which combines unsupervised learning to cluster the tweets and then performing supervised learning methods for Classification. The author applied different machine learning techniques: Naive Bayes(NB), Maximum entropy, support vector machine (SVM) etc. and concluded that NB, SVM with 89.4% accuracy outperforms the other techniques in sentiment classification.

3 Approach taken

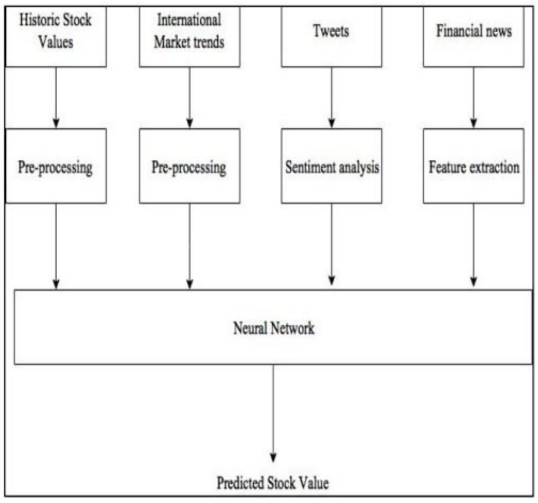


Fig 1: Basic Architecture Model

- Historical Stock Prices: This is the time series data which includes the stock prices of the companies. This data had to be pre-processed in order to handle missing values in the time series and remove all invalid data.
- Tweets: All tweets relevant to the companies were collected from Twitter and Sentiment Analysis was performed to get the overall sentiments about the company.
- Financial News: This includes all latest events in the financial market. The news headlines were preprocessed and converted to multidimensional vectors suitable for input to the neural network.
- Neural Network: The Artificial Neural Network is trained to find the correlation between all the factors and predict the stock price based on the input.

3.1 DataSets

Three different datasets have been used in this Project for 2 different stocks (Infosys and TCS) each:

1) **Historical Stock Data:** Daily price dataset from Yahoo Finance.
2) **News Headlines:** News headlines from moneycontrol.com.
3) **Twitter Data:** Company specific tweets from 1st September 2017 to 25 Feb 2018.

3.2 Preprocessing Of Data

3.2.1 Historical Stock Price Data

Historical Stock Prices need to be normalized before directly applying the values to the model since the parameters that we are using of are of different scales and units. NaN and missing values were looked before passing data to the model. We went for change in percentage from the previous day as an input to the traditional min-max scalar function.

3.2.2 News Headlines and Twitter Data

Preprocessing of financial headlines and twitter data was essential to remove the noises such as stopwords, redundant words and punctuations as they are not considered important for financial analysis. First of all the unwanted punctuations, characters are removed, contractions are replaced by their original form in the dataset by using a dictionary.(for eg. ain't is replaced by am not, can't replaced by cannot). After that, the news headlines/each tweet are taken and tokenized and stopwords are removed using the NLTK's stopword corpora. Then stemming is done to reduce the inflected terms to their base form.(for eg. bank, banking converted to bank).At the end only for tweets data common emojis have been replaced by their signature words with the help of Emoji Sentiment file which is readily available online.

5 Architecture

5.1 LSTM architecture with Technical Indicators and Tweets Sentiments (Model 1)

Model 1 is the multi stacked LSTM model which takes in the input the different parameters like Open, High, Low, Close values along with these the technical indicators like 3 day Open Simple moving average, 3 day Exponential moving average and the to include the sentiments of the common public on the stocks a column with intensity of positive negative polarity in the range (-1 to 1) is also added. All the input parameter are normalized with the help of Min Max Scalar and then passed to the model thereby reducing the chances of any single parameter getting higher importance because of the just range scale. This model with exponential linear unit function was better able to learn the long term dependencies much better than single and 3 stacked lstm model.

Historic Price Work:

Formula for calculating N-days Moving Average:

Simple N Days Moving Average = Previous N days stock prices / N.

Exponential N Days Moving Average (EMA) =
Current Day value - EMA(previous day)} x Multiplier + EMA(previous day)

where multiplier = (2/(N + 1)) and the EMA for the very initial records are simple to taken as the Simple Moving Average.

Tweets Work :

The preprocessed tweets are the passed on to the SentiWordNet Dictionary and sentiment value for each tokenized word in tweets are collected and the average sentiment of each day during the trading hour is calculated. Then the parameter column given to the model is generated with the help of this formula :

Final Positive Score : 40% * Positive Sentiment Score through SentiNetWord dictionary + 40% * Positive Sentiment Score through tweepy library + 20% Subjectivity obtained through tweepy library.

Final Negative Score : 40% * Negative Sentiment Score through SentiNetWord dictionary + 40% * Negative sentiment Score through tweepy library + 20% Subjectivity score obtained through tweepy library.

The one with the higher values is taken as the days overall sentiment on the stock and the intensity is calculated as the difference of the final positive score and final negative score.

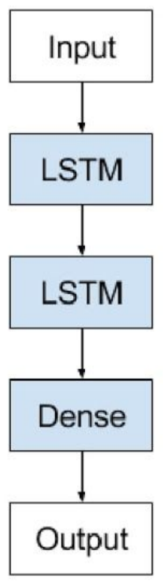


Fig 2: 2 Layered LSTM model for historical stock prices data.

5.2 Jointed CNN and LSTM architecture with News Headlines and Change in Opening Price(Model 2)

Grabbing the sentiments from short texts is very challenging as it contains very limited contextual information and also the information is semantically sparse. Thus we made use of CNN which is good for extracting local features and the output of the same is passed on to the LSTM which is able to learn and remember the long-term dependencies. After preprocessing, we mapped the change in opening prices with the previous days news headlines. All the headlines of the same date were taken and embedded using the Glove words. To create the weights that it is to be used for the model's embeddings, we created a matrix consisting of the embeddings relating to the words in our vocabulary. If a word is found in GloVe's vocabulary, then we used it as it is in the pre-trained vector, otherwise given a random embedding for it which later updated as the model trained so the new 'random' embeddings were more accurate by the end of training. A fixed size of 300 for embedding dimensions was used to match the actual Glove Vectors. If the output dimensions is less than 300 then padding was applied to the headlines embeddings.

Thus finally we had a matrix $\mathbf{Mat}_{\mathbf{r} \times \mathbf{300}}$ where r is the no of rows in the dataset.And each row S=[C_1 C_2 C_3 …...] is the Sentence level representation where the column vector $\mathbf{C_i}$ corresponds to the embedding of the **ith** word in the sentence.

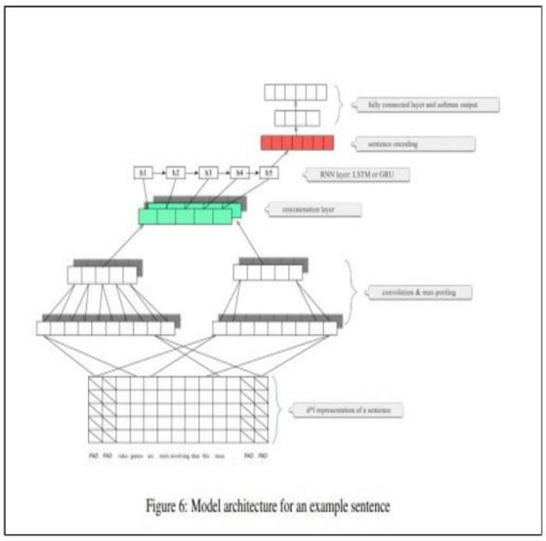

fully connected layer and softmax output
sentence encoding
RNN layer: LSTM or GRU
concatenation layer
convolution & max-pooling
d*l representation of a sentence
Figure 6: Model architecture for an example sentence

Fig 3 : Combined CNN - LSTM Model

The weighted matrix was then passed to the Convolution layer which applies a matrix-vector operation to each window of the sentence level representation with RELU (Rectified Linear Unit) activation function and some bias being added to get a feature mapping to extract the advanced n-gram feature vectors after applying max-pooling operation with multiple filter lengths. We used max-pooling as it gave better result than compared to average pooling. This feature vectors are given as input to the LSTM which maps the change in opening price and the vectors and tries to find the correlation between them. The model uses Adam Optimization Algorithm to update the network weights while training and reduces the mean squared error.

5.3 Overall Model

Inorder to take the advantage of both the model, the final predicted model is given by taking 40% of the Model 1 value and 60% of the Model 2 value.

6 Results

6.1 Model 1 (2 Layered Lstm Model)

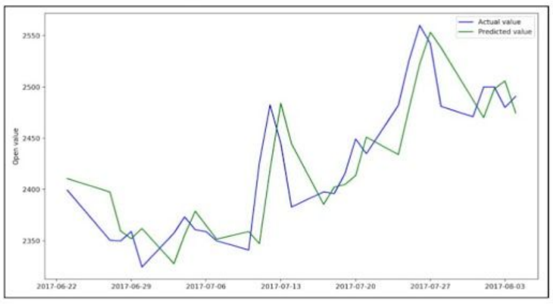


Fig 4 : Output of 2 layered LSTM model

Model graph seems to be better but still seems to be somewhat biased towards the upward trend.

6.2 Model 2 (Combined CNN- LSTM Model)

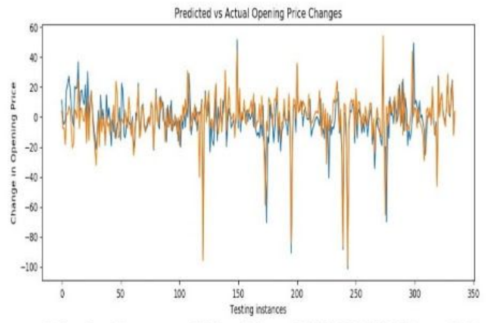


Fig 5 : Output of Combined CNN LSTM model

Output of the model looks satisfactory as it maps both upward and downward in a much better way. However a sudden and a large change are not effectively marked up the model but that it is somewhat considerable.

6.3 Ensembled Model

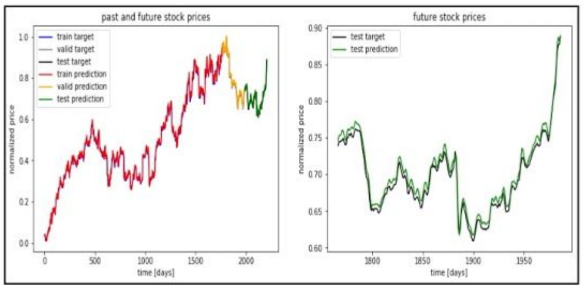


Fig 5 : Output of Combined CNN LSTM model

The ensembled model graph output seems to better map the target prediction.

Model	Stock	RootMean Squared Error	Up/Down Accuracy
Multi Stacked LSTM model	Infosys	14.231	73.21 %
	TCS	26.245	71.83%
Combined CNN LSTM model	Infosys	13.067	69.85 %
	TCS	21.457	75.83%
Ensembled Model	Infosys	12.759	76.54%
	TCS	18.451	77.86%

Table 1: Results of our different model with rmse error and classification(upward/downward trend) accuracy.

7 Conclusion

The project objective of adding newsfeeds, twitter data to increase the prediction accuracy along with historic stock prices does gave us a better result than compared to only historical stock prices. Along with this using LSTM, CNN layers helped in better mapping the events. Though the root mean squared error value of our models were not substantially different from that of AR, ARIMA but still they were able to outperform the classification accuracy to that of the traditional ones. Addition of twitter data to the model didn't had a very major effect on the prediction value but this maybe due to the smaller dataset (only 1 year data). Collecting more tweets and working out with them might help us achieving better accuracy and satisfaction.

8 Acknowledgment

We would like to express our deepest appreciation to Sankarson Banerjee, CTO,NSE India, to our mentor Mrs. Asha Bharambe for helping us throughout the project.We also to appreciate the comments and advices given by our college reviewers during our project presentations.

References

[1] Stock Price Prediction Using Financial News Articles
http://ieeexplore.ieee.org/document/5609404/

[2] Sentiment analysis of Twitter data for predicting stock market movements
http://ieeexplore.ieee.org/document/7955659/

[3] Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation
http://ieeexplore.ieee.org/abstract/document/1631572/

[4] Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts
https://www.aclweb.org/anthology/C/C16/C16-1229.pdf

[5] Sentiment Analysis of Twitter Data for Stock Market Prediction
https://www.ijarcce.com/upload/2017/march-17/IJARCCE%20129.pdf

[6] Stock market prediction using ARIMA model
http://ijssst.info/Vol-15/No-4/data/4923a105.pdf

[7] Forecasting of Stock Market Indices Using Artificial Neural Network
http://irjcjournals.org/ijieasr/Dec2015/2.pdf

[8] Prediction Model of the Stock Market Index Using Twitter Sentiment Analysis
http://www.mecs-press.org/ijitcs/ijitcs-v8-n10/IJITCS-V8-N10-2.pdf

[9] Feature selection for stock market analysis.
https://pdfs.semanticscholar.org/893d/0bdf90186ca2b6c7327813a240f2fd713ed5.pdf

[10] K. Elissa, "An Overview of Decision Theory," unpublished. (Unpublished manuscript)

[11] R. Nicole, "The Last Word on Decision Theory," *J. Computer Vision,* submitted for publication. (Pending publication)

Dear Author/Research Scholar,
This is to inform you that your paper has been accepted for publication with **IJRASET**, which will be published in our **Volume 6 Issue IV (April 2018).**

Paper Title : Stock Price Prediction using News Headlines, Twitter Sentiment Analysis and Technical Indicators

Paper ID : IJRASET16258
Paper Status : Paper Accepted

Note: You can Check status/ Download Certificates & Paper online from our website (Paper ID is required).
You are advised to complete the process for the publication of your research paper.