Stock Price Prediction with Twitter Sentiment Analysis using LSTM

Ann Sara Sajee, Gunjan Ratan Lalwani, Divya Priya Emmanuel
{sajee.a,lalwani.g,emmanuel.di}@husky.neu.edu
CYSE 7245, Spring 2018, Northeastern University

Abstract -- Stock prices are totally random and unpredictable. It has attracted many attentions from academic as well as business. However, it is a challenging research topic, in which many advanced computational methods have been proposed, but not yet attained a desirable and reliable performance. There are several time-series forecasting techniques like auto regression (AR) models, moving average (MA) models, Holt-winters, ARIMA etc., to name a few but, what is the need for yet another model like LSTM-RNN to forecast time-series? This will be the conclusion. The proposed method is a model independent approach. Here we are not fitting the data to a specific model, rather we are identifying the latent dynamics existing in the data using deep learning architectures.

I. Introduction

RNN"s (LSTM"s) are pretty good at extracting patterns in input feature space, where the input data spans over long sequences. It can almost seamlessly model problems with multiple input variables (but the input need to be transformed to 3D input vector). The other best feature is its flexibility to use several combinations of seq2seq LSTM models to forecast time-series: many to one model (useful when we want to predict at the current timestep given all the previous inputs), many to many model (useful when we want to predict multiple future time steps at once given all the previous inputs) and several other variations. This study proposes model that implement both linear (AR, MA, ARIMA) and Neural Network (LSTM) algorithms [1], but they focus on predicting the stock open price for next day by sentimental measures calculated from twitter movement or price forecasting for a single company using the daily closing price. The results will be compared and tuned to provide a best model with our analysis.

II. Related Work

This section briefly surveys related work on stock market prediction using sentiment measures.

One area of research concentrates on predicting the stock market using econometrics models. Lee et al. used the investors intelligence (II) sentiment index and employed a generalized autoregressive conditional heteroscedasticity-in-mean (GARCH) specification to test the impact of noise trader risk on both the formation of conditional volatility and expected return [2]. Baker and Wurgler used the principal component analysis to construct an investor sentiment index, and predicted that a group of investor sentiment had larger effects on securities whose valuations were highly subjective and difficult to arbitrage [3]. Brown and Cliff investigated investor sentiment and its relation to near-term stock market returns and found that although sentiment levels and changes were strongly correlated with contemporaneous market returns, the test showed that sentiment had little predictive power for near-term futures stock returns [4]. Another area of research is that of capturing media influence on stocks and bridging some connections based on artificial intelligence and natural language processing techniques. Most research based on deep learning are only using stock history data and market trading indicators as input variables. In this paper, we integrate sentiment measures and related market data into a deep learning model to forecast a future stock price and examine its result with other models.

III. Dataset

The paper implements three different models and in two different ways in each model. We are using two different data: one consists of stock data with following features - open, close, high, low and volume for five years and tweets from tweepy with calculated sentimental measures. Below are the variables description:

open: the opening price of an exchange on a given trading day

close: the opening price of an exchange on a given trading day

high: the highest price at which a stock traded during the course of the day

low: the lowest price at which a stock traded during the course of the day

volume: the number of shares during a given day

prevday_open: the previous day opening price for the given date

IV. Methodology

As shown in Fig. 1 we proceed in three phases. In the first phase, we employ Aylien API of twitter which gives us tweets which are classified into one of three categories: positive, negative, or neutral. In the second phase, we construct a tweets sentiment index to measure the daily mood of stock market. In the third phase, we deploy a Long Short-Term Memory model to test the hypothesis that the prediction accuracy of stock market prediction models can be improved
by including measurements of investor
sentiment.

V. Twitter Analysis

Tweets sentiments analysis plays a major role in understanding the changes in the stock market rates. The tweets extracted from Tweepy API were pre-processed before applying Naïve Bayes algorithm.

Data Pre-Processing

Tweets consist of many acronyms, emoticons and unnecessary data like pictures and URL"s. So, tweets are pre-processed to represent correct emotions of public. For pre-processing of tweets, we employed a number of stages of filtering:

a) Stop word Removal: Words that do not express any emotion are called Stop words. After splitting a tweet, words like a, is, the, with etc. are removed from the list of words.

b) Lower case: Converting the tweet into lower case.

c) Hash tags: Some words in tweets are prefixed with hash symbol (#). It is used to show the current trending topic. Replace all the hash tag words with the words itself.
Example: #Apple replaced by Apple.

d) Handles: Every user in Twitter has username which is prefixed by „@". To pre-process data, we need to remove it by replace it by USER.
Example: @Apple replaced by USER.

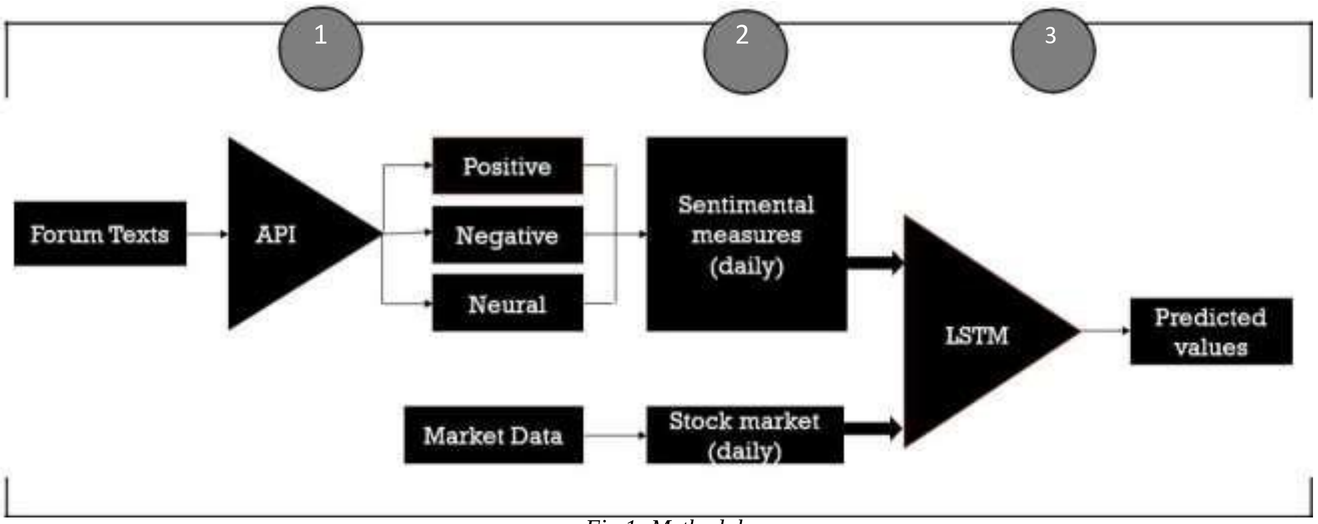


Fig 1: Methodology

e) URLs: A lot of tweets have hyperlinks in them. These must be replaced by URL. Example:
http://t.co/FCWXoUd8 replaced with URL.

f) Emoticons: Emoticons are new ways to express your emotions. Hence, a lot of people today use emoticons to express in their tweets. These emoticons have to be replaced by their correct emotions.
Example: SMILE, LAUGH, LOVE etc.

g) Punctuations: There is a lot of punctuation marks which are used in tweets are replaced by the correct words.
Example: DOT, EXCL, QUES etc.

h) Repeating words: Words which are prolonged to show emotion. These words have to be replaced with accurate words.
Example: Cooooooooool is replaced with cool.

i) Tokenizaton: A tokenizer that divides a string into substrings by splitting on the specified string (defined in subclasses).

Feature Vector:

Feature Vector is generated of all the tweets after this step. This is a crucial step before applying classification algorithms to these tweets. [5] It determines how successful your classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

Naïve Bayes Classification:

Naïve Bayes classifier is a supervised classification algorithm. It is based on "Bayes" theorem with a "naïve" assumption of independence between every pair of feature in feature vector. Naïve Bayes helps us to classify tweets using Tweepy into three categories: positive, negative and neutral.

VI. Different models

For the models, we tried different datasets for better result. At last took a dataset of Apple stocks from "https://www.nasdaq.com" for 5 years. Feature engineering was performed to add a new column "***prevday_open"*** that was created using "*open*" price column. In-order for analysis its importance, we created two different dataset with and without "***prevday_open"*** column.

(a.) Time Series Forecasting:

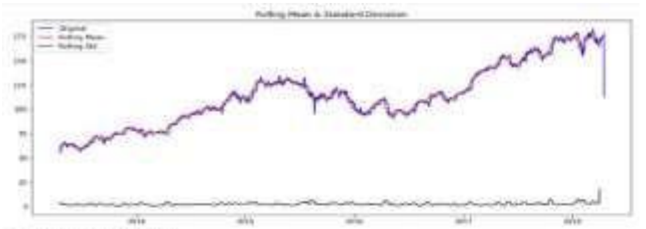
Time series(TS) is a collection of data points collected at constant time intervals. These are analysed to determine the long-term trend so as to forecast the future or perform some other form of analysis. But what makes a TS different from say a regular regression problem? There are 2 things:

- It is time dependent. So, the basic assumption of a linear regression model that the observations are independent doesn"t hold in this case.
- Along with an increasing or decreasing trend, most TS have some form of seasonality trends, i.e. variations specific to a particular time frame.

A TS is said to be stationary if its statistical properties such as mean, variance remain constant over time. But why is it important? Most of the TS models work on the assumption that the TS is stationary. Intuitively, we can use that if a TS has a behaviour over time, there is a very
high probability that it will follow the same in the future. Also, the theories related to stationary series are more mature and easier to implement as compared to non-stationary series.

Stationarity is defined using very strict criterion. However, for practical purposes we can assume the series to be stationary if it has constant statistical properties over time, ie. the following:

1. constant mean
2. constant variance
3. an autocovariance that does not depend on time.



```
Results of Dickey-Fuller Test:
Test Statistic                   [illegible]
p-value                          [illegible]
#Lags Used                       [illegible]
Number of Observations Used      [illegible]
Critical Value (1%)              [illegible]
Critical Value (5%)              [illegible]
Critical Value (10%)             [illegible]
dtype: float64
```

Fig.2

So, more formally, we can check stationarity using the following:

- Plotting Rolling Statistics: We can plot the moving average or moving variance and see if it varies with time. By moving average/variance I mean that at any instant „t‟, we‟ll take the average/variance of the last year, i.e. last 12 months. But again this is more of a visual technique.

- Dickey-Fuller Test: This is one of the statistical tests for checking stationarity. Here the null hypothesis is that the TS is non-stationary. The test results comprise of a Test Statistic and some Critical Values for difference confidence levels. If the „Test Statistic‟ is less than the „Critical Value‟, we can reject the null hypothesis and say that the series is stationary.

Estimating & Eliminating Trend

One of the first tricks to reduce trend can be transformation. For example, in this case we can clearly see that the there is a significant positive trend. So, we can apply transformation which penalizes higher values more than smaller values. These can be taking a log, square root, cube root, etc.

It is easy to see a forward trend in the data. But it‟s not very intuitive in presence of noise. So, we can use some techniques to estimate or model this trend and then remove it from the series. There can be many ways of doing it and some of most commonly used are:

1. Aggregation – taking average for a time like monthly/weekly averages

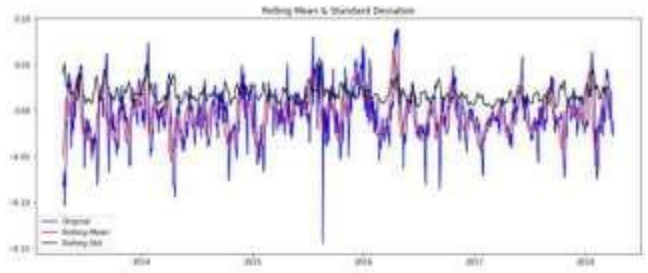
2. Smoothing – taking rolling averages
Polynomial Fitting – fit a regression model

But here we are using Smoothing which refers to taking rolling estimates, i.e. considering the past few instances.

Moving average

In this approach, we take average of „k‟ consecutive values depending on the frequency of time series. Here we can take the average over the past 12 days, i.e. last 12 values. Pandas has specific functions defined for determining rolling statistics.

The red line shows the rolling mean. Let‟s subtract this from the original series. Note that since we are taking average of last 12 values, rolling mean is not defined for first 11 values.



```
Results of Dickey-Fuller Test:
Test Statistic                 -7.7020[illegible]e+00
p-value                         [illegible]e-12
#Lags Used                      1.300000e+01
Number of Observations Used     1.2[illegible]e+03
Critical Value (1%)            -3.4[illegible]e+00
Critical Value (5%)            -2.86[illegible]e+00
Critical Value (10%)           -2.56[illegible]e+00
dtype: float64
```

Fig.3

This looks like a much better series. The rolling values appear to be varying slightly but there is no specific trend. Also, the test statistic is smaller than the 5% critical values, so we can say with 95% confidence that this is a stationary series.

Eliminating Trend and Seasonality

There two ways of removing trend and seasonality:

1. Differencing – taking the difference with a time lag

2. Decomposition – modeling both trend and seasonality and removing them from the model.

Differencing

One of the most common methods of dealing with both trend and seasonality is differencing. In this technique, we take the difference of the observation at an instant with that at the previous instant. This mostly works well in improving stationarity.

Decomposing

In this approach, both trend and seasonality are modeled separately, and the remaining part of the series is returned

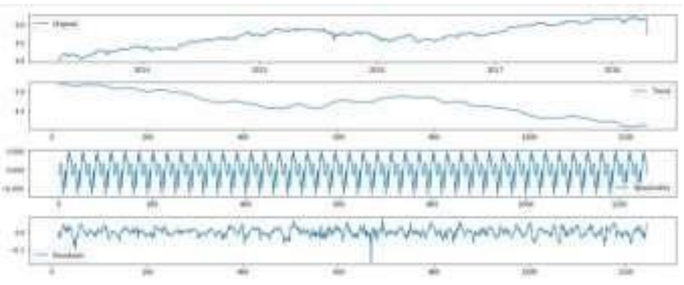
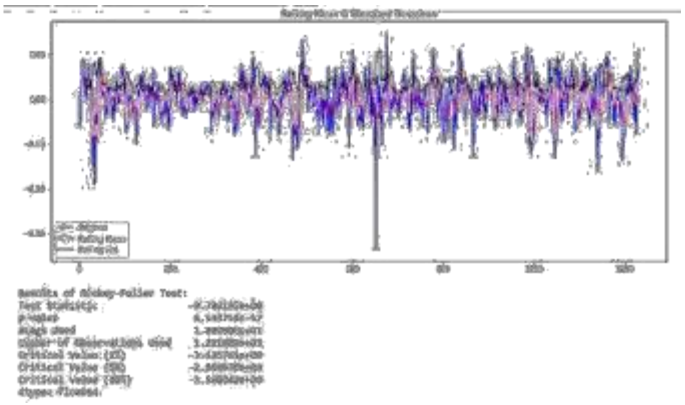


Fig.4



```
Results of Dickey-Fuller Test:
Test Statistic                  [illegible]
p-value                         [illegible]
#Lags Used                      [illegible]
Number of Observations Used     [illegible]
Critical Value (1%)             [illegible]
Critical Value (5%)             [illegible]
Critical Value (10%)            [illegible]
dtype: float64
```

Fig.5

The Dickey-Fuller test statistic is significantly lower than the 1% critical value. So, this TS is very close to stationary.

Forecasting a Time Series:

A strictly stationary series with no dependence among the values. This is the easy case wherein we can model the residuals as white noise. But this is very rare. A series with significant dependence among values. In this case we need to use some statistical models like ARIMA to forecast the data. Below is the brief introduction to ARIMA.

ARIMA stands for Auto-Regressive Integrated Moving Averages. The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p,d,q) of the ARIMA model.

AR Model

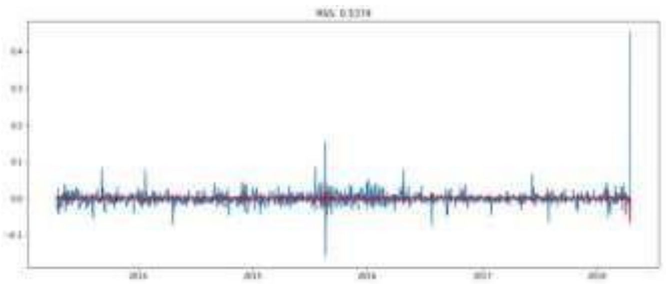


Fig.6

MA Model

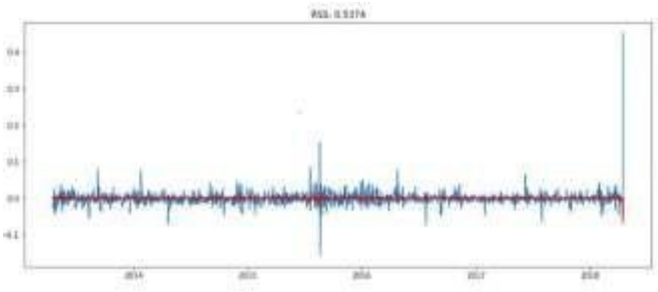


Fig.7

Combined Model

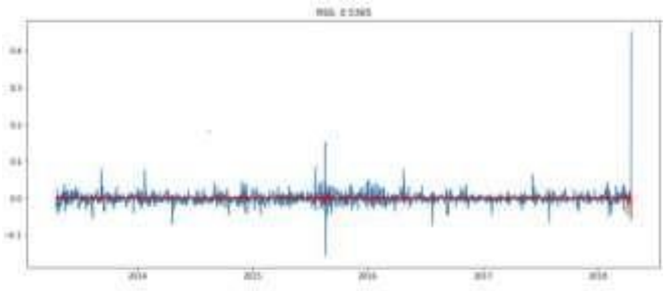


Fig.8

Here we can see that the AR and MA models have almost the same RSS but combined is significantly better. Now, we are left with 1 last step, i.e. taking these values back to the original scale.

Taking it back to original scale

Since the combined model gave good result, we scaled it back to the original values to see how well it performs there.

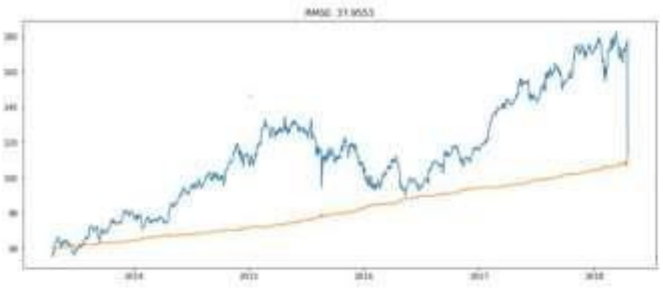


Fig.9

We conclude that this is not a very good forecast. We can keep on tuning the parameters to get the good forecast.

(b.) LSTM with stock data:

This model was constructed only with stock market data with no added feature. Below is the plot between predicted and test data.

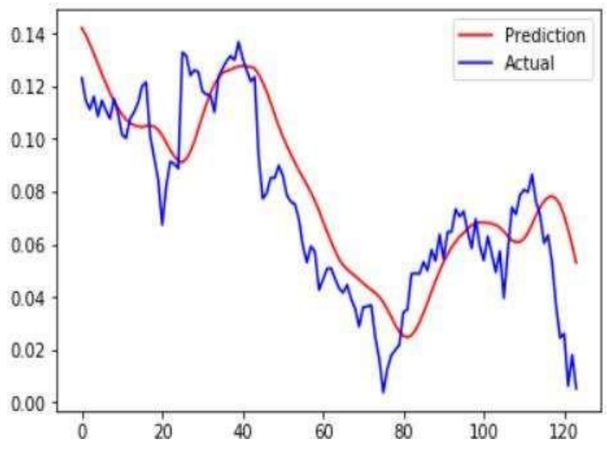


Fig.10

(c). LSTM with stock data and "*prevday_open*": This model as six features including "prevday_open". But below graph is evident that the added new feature has improved our model performance.

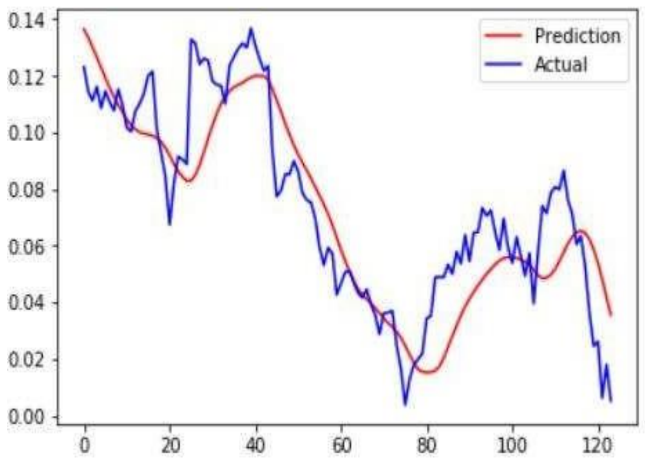


Fig.11

(d.) LSTM with stock data and sentimental measures:

In here, we have considered the sentimental measures from Alyien api along with the stock data. But still the results were not as expected.

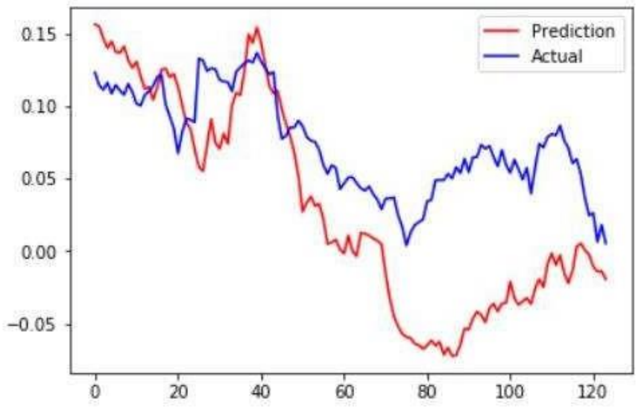


Fig.12

(e.) LSTM with stock data, "*prevday_open*" and sentimental measures

This model considered seven features including extra added feature. We still need to train the model with more tuned parameters.

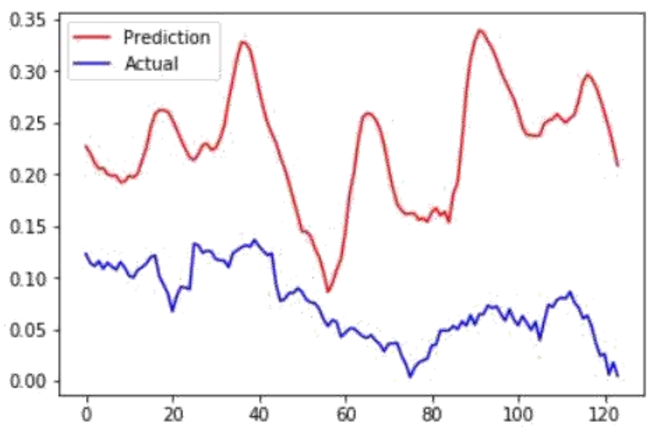


Fig.13

Result

Below are the error metrics we obtained so far, we"ll try to train the model for better result.

Models	MAE	MAPE	RMSE	R^2
ARIMA	42.75	22.9	35.6	-1.73
LSTM – Model1	0.015	50.25	0.019	0.68
LSTM – Model2	0.014	36.40	0.017	0.73
LSTM – Model3	0.15	364.45	0.16	-20.9
LSTM – Model4	0.05	122.91	0.06	-2.4

Table.1

VII. Conclusion

We started are analysis with simple ARIMA model and researched further to LSTM to see the difference between both. LSTM proved practically that its best than ARIMA for stock prediction. The model with dataset that had extra feature was considered best after various tuning of hyperparameters. We took the same dataset along with tweeter sentiment measure but the model responded with bad results. Extracting tweets randomly for a single day will not have much difference or any impact on a stock price of a company on any given day. As it was discussed earlier, stock price rise and fall is random. It might or might not include environment changes, sudden breaking news, unexpected statement from a famous personality or even might be a weather climate change. Hence, we conclude that Stock prediction is not as easy as predicting any other data. It requires lot of data which might not even directly be relevant to it. The model will give better result if any furthermore features has been added as its predictor.

References

[1] Sreelekshmy Selvin, Vinayakumar R,Gopalakrishnan E.A, Vijay Krishna Menon, Soman K.P. “Stock Price Prediction using LSTM,RNN and CNN-sliding window model”. IEEE(2017): 1643-1647

[2] Jiahong Li, Hui Bu*, Junjie Wu. “Sentiment-Aware Stock Market Prediction: A Deep Learning Method”. National Natural Science Foundation of China (71471009,71531001, 71373001, 71671012)

[3] Lee, Wayne Y., Christine X. Jiang, and Daniel C. Indro. "Stock market volatility, excess returns, and the role of investor sentiment." Journal of banking & Finance 26.12 (2002): 2277-2299.

[4] Baker, Malcolm, and Jeffrey Wurgler. "Investor sentiment and the cross-section of stock returns." The Journal of Finance 61.4 (2006): 1645-1680.

[5]http://colah.github.io/posts/2015-08 Understanding-LSTMs/

[6]http://www.jakobaungiers.com/articles/a/LSTMNeural-Network-for-Time-Series-Prediction

[7]https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7995302

[8]https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8256643&tag=1

Code with Documentation:
https://github.com/NEU-BDIA-ADG/Stock_Market_Prediction-using-LSTM
(along with local web application predicting model with LSTM)