Sentiment analysis on news articles to forecast EUR-USD exchange rates

Gopinath Dayalan and Xiaobai Liu

June 29, 2020

Publication Number: CSRCR2020-03

Computational Science & Engineering Faculty and Students Research Articles

Database Powered by the Computational Science Research Center Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

Sentiment analysis on news articles to forecast EUR-USD exchange rates

Gopinath Dayalan[1], Xiaobai Liu[2]

Computational Science Research Center
San Diego State University
[1]gdayalan1814@sdsu.edu
[2]xiaobai.liu@sdsu.edu

***Abstract*— In recent times, text mining, topic modeling, and sentiment analysis have gathered huge attention due to the vast availability of data in social media, news articles, and other digital sources. Even though digitized, articles are typically unstructured and organized based on date and time rather than subject. Collecting a month data would result in extracting thousands of articles which leads to a big data challenge. In this paper, we present data extraction, preprocessing, normalization, feature engineering, sentiment analysis, natural language processing (NLP), LSTM to discover the relation between the news articles and EUR-USD exchange rate**

I. INTRODUCTION

The amount of data we produce every day is truly mind-boggling. There are 2.5 quintillion bytes of data created each day at our current pace, but that pace is only accelerating with the growth of the Internet of Things (IoT). Over the last two years alone 90 percent of the data in the world was generated. Data and information or knowledge are often used interchangeably, however data becomes information when it is viewed in context or in post-analysis. While the concept of data is commonly associated with scientific research, data is collected by a huge range of organizations and institutions, including businesses (e.g., sales data, revenue, profits, stock price), governments (e.g., crime rates, unemployment rates, literacy rates) and non-governmental organizations.

Information published in the news articles are reliable source and helps forecast currency rate. News articles provides us a new dimension for predicting the currency rate. Sentiment from the news articles becomes a key factor in predicting the rise and fall of the currency rate. It is important to analyse the information as soon as possible so it can help predict the currency rate. The task of analysing the vast number of articles manually and predicting the currency rate is very tedious. This means that an automated system is needed to extract, process, structure and predict. News articles are extracted from websites, processed to a structural format and stored for prediction. Not every article talks about our topic related matters. So, a filter is placed after data extraction to neglect articles which are not related to the currency rate.

The time series prediction problem involves textual data so, several pre-processing steps are used in these kinds of works. Natural Language Processing (NLP) tool kits are used of these purposes. Sentiment analysis is performed on these cleaned and processed textual data. Finally, in order to train the AI model, historical currency rate is also used along with the sentiment of each article to forecast the EUR-USD currency rate.

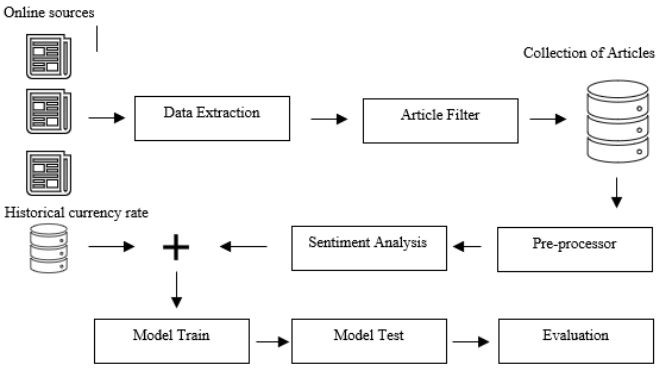

Fig 1. System Architecture

In this paper, we discuss on how the articles are extracted, the techniques used for pre-processing, visualizing, extracting features from the news articles and develop a model training and testing with historical currency rate and features from the news article to forecast the currency rate. The model is evaluated in three levels. i.e. a base line model such as Autoregression Integrated Moving Average (ARIMA) which is a time series forecasting model with historical currency rate as input. Next, a Long Short-Term Memory (LSTM) model with historical currency rate without sentiment score and finally, a LSTM model with historical currency rate and sentiment score.

II. LITERATURE REVIEW

(Gholampour & van Wincoop, 2019) has discussed on how data from twitter can be used to predict the currency rate

between euro and dollars. Their research uses the data from vast financial tweet accounts and rate them positive, negative or neutral based on the underlying sentiment. Word embedding technique has been used to prepare a large list of possible words that can be in the tweets and categorized them into positive, negative and neutral. This type of embedded set preparation can be made possible using n-gram approach where n denotes the number of words in 1 set. N-gram provides the list of most frequently co-occurring words from the entire sample of data. (Gholampour & van Wincoop, 2019) approaches the problem, like a supervised machine learning technique where the prepared financial lexicons with labels are compared with the tweets and decide on the probability along with the google machine learning tool. Then, the system predicts whether it will a bullish or a bearish day. i.e., bullish denotes the rates will go high, bearish meaning the rates will go down. The author also talks about the bad actors, where an actor believes a currency will appreciate in a medium run, post negative tweets to make other to sell it. But our model uses news article where such possibility is very low. This model is validated with mean squared error metrics.

(Huang, Lehkonen, Pukthuanthong, & Zhou, 2018) research involves predicting not only the currency rate, but also stocks bonds, commodities and housing. This paper uses data from news articles and social media and evaluates the sentiment underneath. Principal component analysis(PCA) and Partial least squares(PLS) based sentiment index has been used for prediction. The evaluation is based on the returns from individual market.

(Mohan, Mullapudi, Sammeta, Vijayvergia, & Anastasiu, 2019) research work is on predicting stock based on the news article. Authors have filtered the data using relevance of the company, which is being monitored, but have not considered cleaning the data samples by removing non-lexical terms and other stop words. The authors use autoregression models like ARIMA and neutral network RNN for prediction. These models are evulated based on Mean absolute percentage error (MAPE).

III. Our Approach

A. Overview

A web crawler & scrapper is built with python library scrapy to collect news articles from the web. These news articles are filtered for articles to have only articles related to currency rate, which is accomplished using NLTK. From this collection of articles, preprocessing is carried out to remove the stop words and custom words are added which doesn't provide any useful information or being repetitive in every article. The polarity of each article is calculated and classified as positive and negative. The number of positive and negative articles are calculated and grouped for each day. This count is used as the input for the model along with the historical currency rate.

The sentiment analysis is carried out using TextBlob and Vader. A segment of previous currency rate is used as the input, where the segment size is defined a parameter called window size. For example, the currency rate of a week or two is used as input along with the positive and negative sentiment count. The model is developed in LSTM architecture – a special type of Recurrent Neural Network (RNN).

B. Data Collection

News articles are extracted from the web using scrapy sitemap spider, a python library. For example, let us see the implementation for Reuters. Articles of Reuters are indexed daily, and the index link is available in the robot txt file of the site. A web-crawler is developed to generate the sitemap links according to the pattern of the index. The sitemap links generated provides an XML output which has the list of URLs of the articles published on that day. Next, the crawler fetches each article and scraps the address, the title of the article, actual content and the time of publishing. It ignores the rest, which is of the repeated header, footer, and advertainments. This process of eliminating the redundant contents is achieved using the class name of HTML elements. The class name "ArticleHeader_headline" fetches the title, "StandardArticleBody_body" fetches the body and "ArticleHeader_date" fetches the time of publishing. Similarly, data extractors can be built for other web sites by understanding the webpage structure of different sites.

Extraction of news articles from a web site takes humongous amount of time. For this research, 6 months of news articles which sums of to 94726 is collected, which has a daily average around 526. These articles are then filtered to get the relevant news articles and rest are ignored. The exchange rate for EUR USD is also collected using pandas data reader API for about the same period as the news articles

C. Data Preprocessing

The raw data is cleaned from special characters and numbers by using a regular expression and tokenized. Tokenization is the step by which articles are split into an array of words called tokens. A larger chunk of the article is tokenized into words and token of each word called a corpus. Tokenization is also referred to as text segmentation or lexical analysis. The raw data consists of a lot of noise like the commonly used words such as "the", "a", "an", "in", symbols and numbers. These are useless data and contributes to inefficient topic modeling. NLTK library provides the list of stop words which are matched and removed. Then, with the cleaned data, EUR-USD currency rate related keywords are matched and the articles which have at least a word from the keywords are selected for the dataset. Below are the set of keywords used.

keywords = ['usd', 'us dollar', 'us dollars', 'eurusd', 'eur/usd', 'usdeur', 'usd/eur', 'eur', 'euro', 'european dollar', 'european dollars']

Fig 2. Word cloud of data before filtering

Even though we have removed stop words, there are some words that are not in the stop word vocabulary, but nevertheless appear in a large portion of our documents. Similarly, some words have a very low count in the corpus. These words could be anomalies that will not contribute to our model and removing them will speed up and improve our model. News articles will usually have days and months in their content, which needs to be removed as they don't add any value. Along with those few more custom stop-words have been added to the stop words list.

custom = ["company", "said", "trump", "donald", "white", "house", "also", "would", "president", "including", "democrates"]

Fig 3. Word cloud of data after filtering

Normalization makes the data to be on the same space and allows processing to proceed uniformly. Stemming and lemmatization are the ways to extract the root word for the given word. The difference between these two methods is stemming cuts of the given word aggressively which may not be an actual word, compared to lemmatization. Stemming eliminates the affixes (suffixed, prefixes, infixes, circumfixes) from a word to obtain a word stem. Lemmatization capture canonical forms based on a word's lemma. e.g. better → good.

D. Sentiment Analysis

Each article must be classified as positive or negative. To do this, TextBlob sentiment analyzer is used. The TextBlob is a huge collection of data, where each word in the English language is given a score. This score helps in finding the polarity and subjectivity of each statement. The document term matrix approach is not used here as the order matters. Eg, "happy" is positive and "not happy" is negative. The text blob returns a score for each article/input where the polarity ranges from -1 to 1 and subjectivity from 0 to 1.

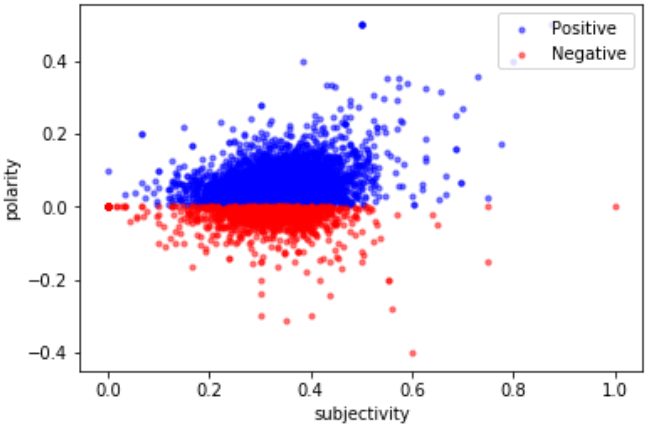

Fig 4. Positive and Negative samples plotted in terms of their polarity and subjectivity.

From literature and research, we could find that the sentence which has polarity greater than zero are positive and lesser than zero are negative. Some sentiment analyzers like "vader" can be used to classify data into positive, negative and neutral. Since neutral articles does contribute much in decision making, only TextBlob is used and vader is ignored.

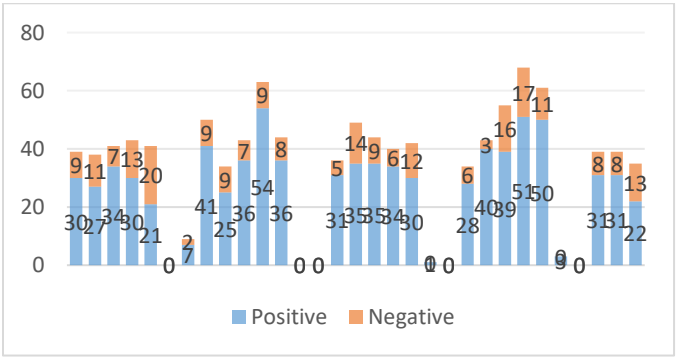

Fig 5. Illustrating the no. of positive and negative articles for a month

In the above, count of positive and negative articles is stacked up. We could also see that the articles relating to currency rates are nil or very few during the weekends, which indicates the filtering based on keywords is promising.

E. ARIMA Design

ARIMA is one of the most famous models used for time series analysis and forecasting. It uses information from the past to predict the future values. It is univariant, i.e., it depends only on one variable designed to analyze probability and stochasticity of the variable and is purely based on historical data without any economic theory. This model requires the input data to be stationary, where it should exhibit mean reversions, should have finite and time invariant variance. The data should depict/represent finite order or convergent autoregression approach. The only hyper parameter for this model is, p – the lag order, d – the degree of difference, q – the order of moving average. These are determined by autocorrelation function and partial autocorrelation function.

F. LSTM Design

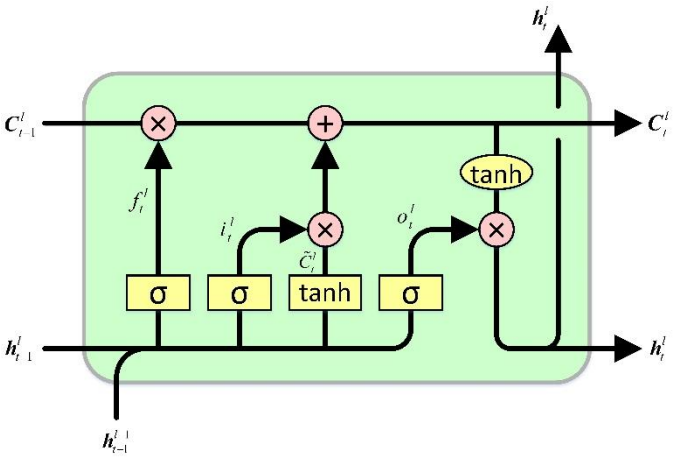

Fig 6. Architecture of LSTM, adapted from (Hochreiter & Schmidhuber, 1997)

LSTM overcomes the vanishing gradient problem. It has three gates; the forget gate depending on f_t ,the input gate depending on i_t and the output gate depending on o_t. These are sigmoid neural layers with point wise multiplication operation. The value of each gate ranges from 0 to 1, decides the level of information to passthrough them. This provides a control system to enable information from the past, control information from the input and likewise for output. The forget gate controlled by function of current state and previous state along weights and passed to a sigmoid function. And the input gate is controlled like forget gate with an addition of tanh to current and previous state. Finally, the output gate is controlled by the tanh function of forget gate and input gate with sigmoid of current and previous state.

G. Implementation

The currency rate is decomposed into observed, seasonal, trend and residual. Observed is the actual opening/closing currency rate for each day. Seasonal, trend and residual add up to observed value.

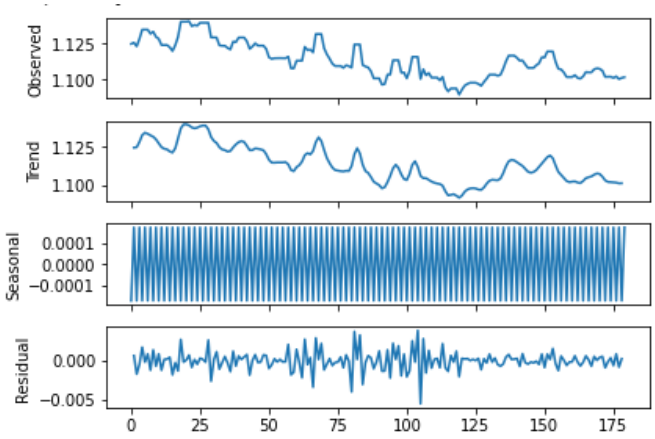

Fig. 7. Decomposition of the currency rate

The p, d and q value for ARIMA is chose from the autocorrelation factor where p can be in the range of 1-10 and d in range of 0-2 and q in range of 0-5. The data setup for ARIMA is historical data(residual).

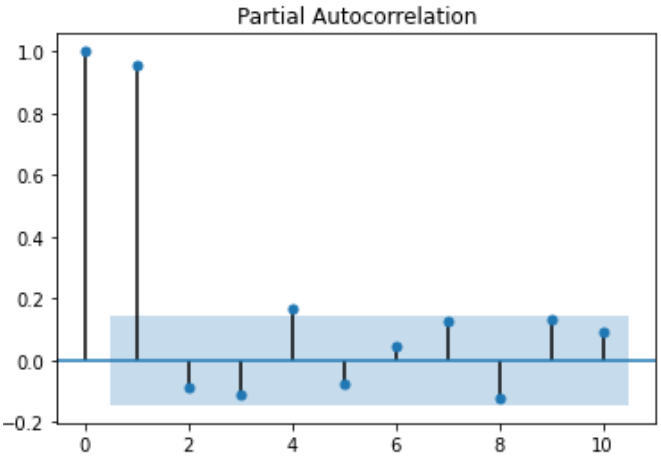

Fig 8. Partial Autocorrelation – relationship between two steps.

Three layers of LSTM has been used in the model and different hyper parameters such as the learning rate, dropout, activation function, optimization, bias, epoch and batch size are chosen to determine the performance of each model. The data setup for LSTM has two approach; without sentiment data and with sentiment data.

IV. Experiments

A. Evaluation

The data is split into training, validation and testing. Training set consists of dataset expect the forecast data, which in our case is two weeks. Each sample of data consist a X of window size and Y of forecast size. For model with sentiment data, the positive and negative percent of that window size is stacked up and the end of X.

X	Y
data of window size	data of forecast size

X	Sentiment Data	Y
data of window size	Postive & Negative percent for data of window size	data of forecast

Table 1 represents the data setup without sentiment data and table 2 represents the data setup with sentiment data.

Above table illustrates the dataset for with sentiment and without sentiment data for LSTM. Models are compared to each other with different hyper parameters and evaluated based on mean squared error(MSE).

B. Methods

1. Approach A.

Three ARIMA models are setup. 1. next day forecast for 2 weeks with observed values. 2. next day forecast for 2 weeks with residual values. 3. 2 weeks forecast with residual values. Best hyper parameters are chosen based on the lowest mean squared error. The data setup for this approach is where input is $rate_t$, $rate_{t-1}$, $rate_{t-2}$ … and output is $rate_{t+1}$

2. Approach B.

LSTM model to predict currency rate $rate_{t+1}$, $rate_{t+2}$, $rate_{t+3}$ …$rate_{t+f}$ with $rate_t$, $rate_{t-1}$, $rate_{t-2}$ …$rate_{t-w}$ where f is the forecast size w is the window size. A sliding window technique is used. Data normalization is applied to get the currency rate to the same scale. Data is transformed to range from -1 to 1. Training, validation and testing are carried out in residual values with different hyper parameters. Along with Adam optimizer, Stochastic gradient descent(SDG) optimizer is also used to experiment as the residual values are randomly changing. An inverse scalar transformation is applied to the predicted values. The predicted residual forecast is adding with the seasonal and trend and evaluated against the observed values.

3. Approach C.

In this approach, a multivariant model is created with the currency rate and positive negative sentiment percent. The input data has $rate_t$, $rate_{t-1}$, $rate_{t-2}$ …$rate_{t-w,}$ *positive percent, negative percent* for that window and output $rate_{t+1}$, $rate_{t+2}$, $rate_{t+3}$ …$rate_{t+f}$. Hyper parameter configuration and evaluation is like approach B.

C. Results

In this section, we discuss on the result obtaining on experimenting with 3 approaches briefed in the earlier section with different hyper parameters. The results are compared and evaluated based on mean squared error. Models results are different with every run, so experiments are carried out multiple times and best results of each model is considered for evaluation.

1. ARIMA (Approach A)

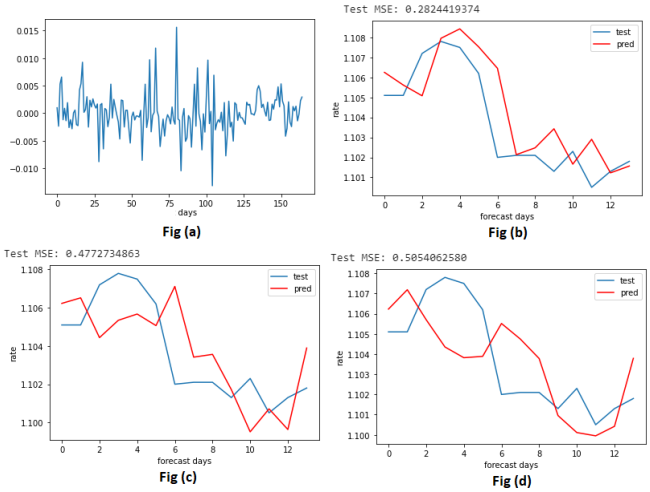

Fig 9. Results from ARIMA Model.

Fig 9(a) represents the residual data decomposed from the 6 months currency rate. ARIMA model 1(fig 9-b) shows the forecast for two weeks where the prediction is basically for next day and iterated for two weeks. This model uses the observed values as the input. ARIMA model 2(fig 9-c) uses the residual data as input and the forecast is for two weeks with next day prediction. ARIMA model 3(fig 9-d) uses the residual data as input and forecasts for 2 weeks.

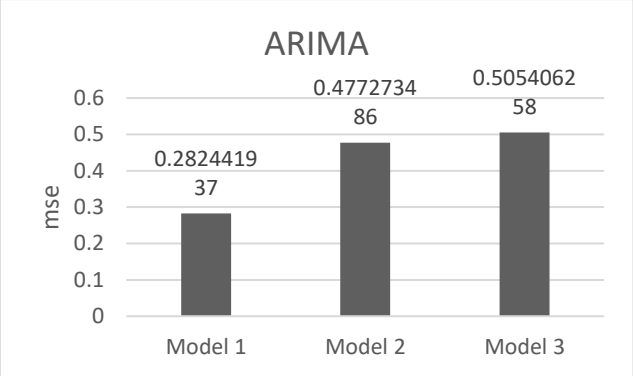

Fig 10. ARIMA model comparison

MSE is in the range of 1e-6 so they are scaled by multiplying with 1e6 to intepret better. From model 1 and model 3, we could see that as we increase the no. of forecast days, the mse increases. The mse for next day prediction will be lesser than two weeks forecast. ARIMA model best suits stationary data.

2. LSTM (Approach B & C)

Approach B & C are tested side by side with different hyper parameters for LSTM.

Param Configuration I.

```
neurons = 50
Optimizer = SDG
learning_rate = 0.003
dropout = 0.2
activation = 'relu'
n_output = forecast
return_seq = True
batch_size = 10
n_epochs = 50
```

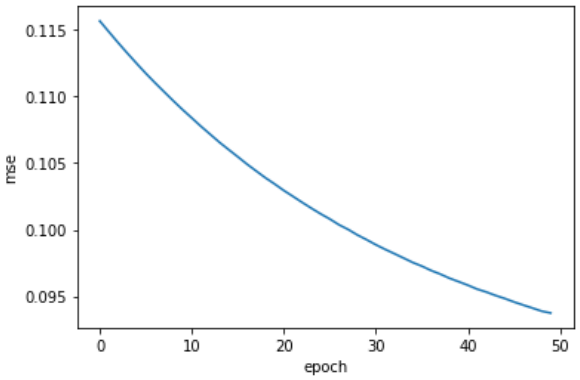

Fig 11. Training loss

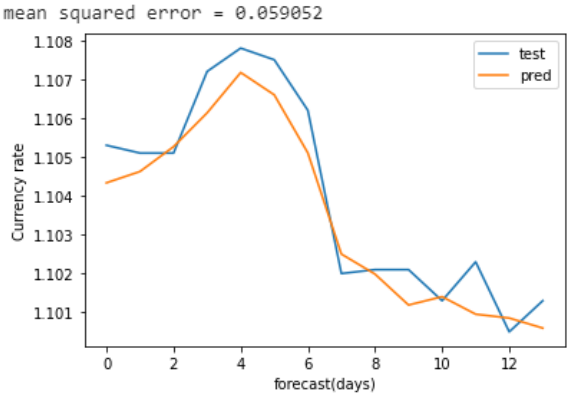

Fig 12. Two weeks forecast without sentiment data

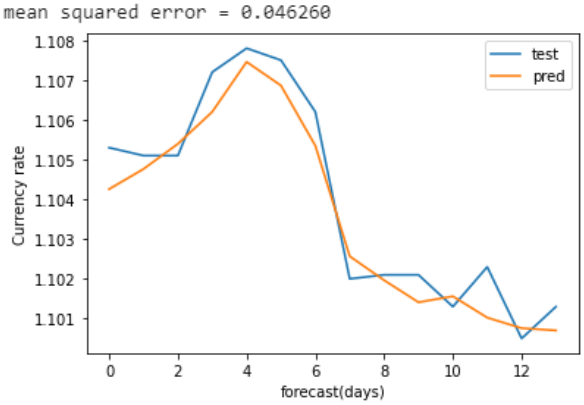

Fig 13. Two weeks forecast with sentiment data

The mean squared error is scaled to the same range as in ARIMA model. This model uses stochastic gradient descent for optimization and is better than ARIMA model.

Param Configuration II.

```
neurons = 200
Optimizer = SDG & ADAM
learning_rate = 0.01
dropout = 0.2
activation = 'relu'
n_output = forecast
return_seq = True
batch_size = 10
n_epochs = 100
```

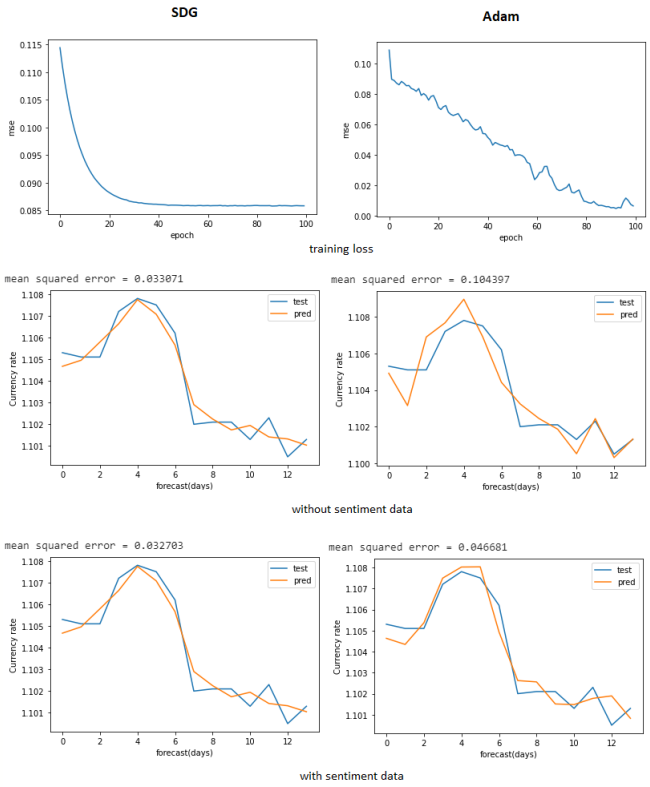

Fig 13. Plots for Param configuration II, with comparison of SDG and Adam optimizer.

From the loss plot, we could incur that the training is converge smooth with Adam optimizer and it doesn't stabilize. This creates a overfitting problem. The MSE model with sentiment data and without sentiment data is so close while using SDG as the optimizer but SDG is far better than Adam optimizer.

Param Configuration III.

```
neurons = 100
Optimizer = SDG
learning_rate = 0.001
dropout = 0.2
activation = 'tanh'
```

```
n_output = forecast
return_seq = True
batch_size = 10
n_epochs = 100
```

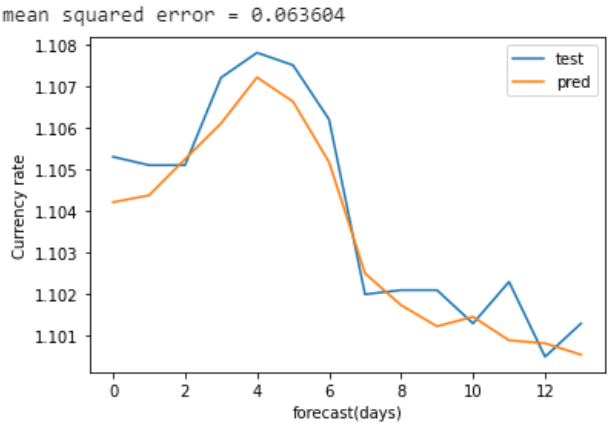

Fig 14. Forecast without sentiment data

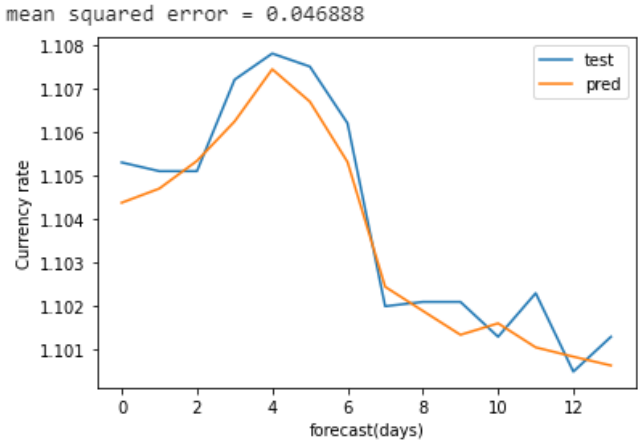

Fig 15. Forecast with sentiment data

The performance of this model is like the model with param configuration I. and increasing the number of neurons and epoch could result as same as model with param configuration II. The next difference in the activation function. We could also see that the MSE of model without sentiment is better than the model with sentiment.

V. Conclusion

In this research, we handled a time series problem by extracting news articles from websites, filtered, pre-processed, performed sentiment analysis and created a model to predict the currency rate. From the results, LSTM performs better than ARIMA model and the comparison between model with sentiment data and without sentiment shows connections between the sentiment of the news articles with EUR-USD currency rate.

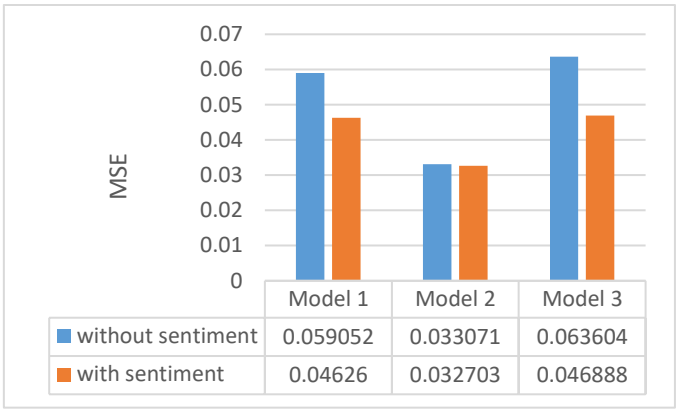

Fig 16. Comparison of LSTM models

From the comparison chart, we could see that there is large difference in the error in model 1 and 2 between using and not using sentiment data. This shows a strong connection between the textual news information available in the internet and the currency rate.

VI. Future Work

Even though LSTM performs well when compared to auto regression models. LSTM models have over fitting and under fitting problem. To overcome this problem, a better deep learning model should be explored. Executing the built script to scrap and store news articles and perform predicting is a tedious process, so in future this can be automated, and the model is continuously trained, validated and tested with new data.

REFERENCES

[1] Gholampour, V., & van Wincoop, E. (2019). Exchange rate disconnect and private information: What can we learn from Euro-Dollar tweets? *Journal of International Economics*, *119*, 111–132. https://doi.org/10.1016/j.jinteco.2019.04.007

[2] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[3] Huang, D., Lehkonen, H., Pukthuanthong, K., & Zhou, G. (2018). Sentiment Across Asset Markets. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3185140

[4] Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., & Anastasiu, D. C. (2019). Stock price prediction using news sentiment analysis. *Proceedings - 5th IEEE International Conference on Big Data Service and Applications, BigDataService 2019, Workshop on Big Data in Water Resources, Environment, and Hydraulic Engineering and Workshop on Medical, Healthcare, Using Big Data Technologies*. https://doi.org/10.1109/BigDataService.2019.00035

[5] Blei, D. M. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 993--1022.

[6] Harshman, S. D. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 391--407.

[7] Lee, D. D. (2000). Algorithms for Non-negative Matrix Factorization. *MIT Press*, 535--541.

[8] Loper, E. a. (2002). NLTK: The Natural Language Toolkit. *Association for Computational Linguistics*, 63--70. Retrieved from NLTK: The Natural Language Toolkit.

[9] Pedregosa, F. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from Scikit-learn: Machine Learning in Python.

[10] Peterson, E. J. (2001). *{SciPy}: Open source scientific tools for {Python}*. Retrieved from {SciPy}: Open source scientific tools for {Python}.

[11] Sojka, R. {. (2010, May 22). *Software Framework for Topic Modelling with Large Corpora*. Retrieved from Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks: http://is.muni.cz/publication/884893/en

[12] (n.d.). *Thomson Reuters: Reuters News Agency*.

[13] (chollet2015keras, 2015)