Assignment 2

CPSC 488: NATURAL LANGUAGE PROCESSING
10/14/2024
DR. CHRISTOPHER RYU
CALIFORNIA STATE UNIVERSITY, FULLERTON (CSUF)

Assignment Information

OPTIONAL: This assignment can be completed individually or by a team with up to 5 members.
Total score: 60
Due date: 12/09/2024 11:59PM
Members: Hammad Sheikh, Matthew Do, Ryan Avancena
Work distribution: Equal

Abstract - Machine learning (ML) and Artificial Intelligence (AI) worlds are experiences tremendous growth. Their applications are uncountable. One realm of applications for ML and AI is Natural Language Processing (NLP). In this aspect, there are various classification models that assist with text classification and processing. Our goal for this assignment is to utilize NLP techniques in analyzing news data for ETF FNGU to build a ML model that would maximize our investment returns. The stock data can be obtained via the Python libraries that Dr. Ryu [1] defined. News articles will be scraped from the web.
Keywords - Stock Trading, ETF, FNGU, Natural Language Processing, Machine Learning

Contents

1 Introduction **2**

2 Approach **2**
2.1 Data Acquisition 3
2.2 Data Pre-Processing 4
2.2.1 Method A [2] 4
2.2.2 Method B [3] 4
2.3 Sentiment Analysis and Model Development 4
2.3.1 Method A [2] 4
2.3.2 Method B [3] 5
2.4 Model Evaluation 5
2.4.1 Method A [2] 5
2.4.2 Method B [3] 5

3 Conclusion and Future Work **6**

4 Acknowledgements **7**

1 Introduction

There is a lot of stock data available in the world. All the stocks are impacted by the environment and the economical status of the world. This information is shared via news articles, which are in text format. We need to able to analyze news data, identify patterns and sentiment, and utilize it to predict impact on stocks. However, due to the sheer size of data available and the amount of convolution, it is not an easy task for a human. If we can utilize the computing power and technology available in computers, we may be able to make an impact. This may not have been possible a few decades ago. However, with the expansion of ML and AI technologies, numerous applications become available. With the use of NLP, we can perform text classification and sentiment analysis.
Without generalizing too much, let us look at an ETF, ticker FNGU, with the goal of applying existing text classification and sentiment analysis algorithms on news and stock data, and analyzing their results for performance and accuracy to maximize our investment returns.

2 Approach

The generalized approach is shown below in Figure 1. (Data Acquisition $\rightarrow$ Data Pre-Processing $\rightarrow$ Sentiment Analysis and Model Development $\leftrightarrow$ Model Evulation.) Further details will be discussed in the following sections.

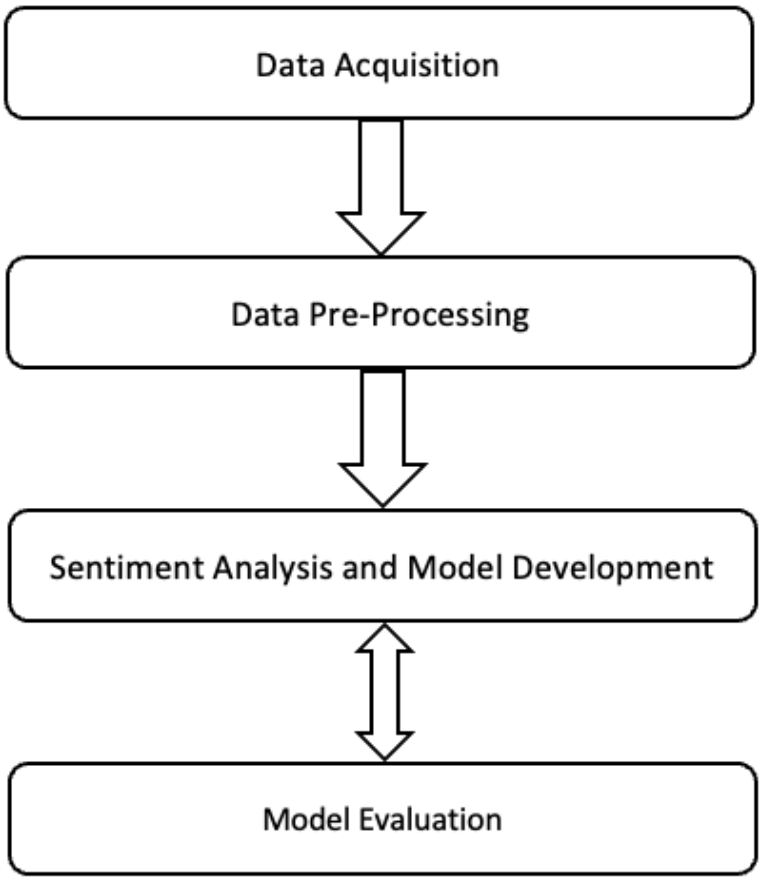

Figure 1: Generalized Approach

We will be relying on existing libraries as much as possible. This will help to focus on the classification and sentiment analysis task instead of building code from scratch for things that already exist. For example, train_test_split from sklearn.model_selection [4] will be utilized to split the data into training and testing datasets. We also tried different methodologies in order to test and assist each other. There are two main methodologies; one primarily developed by Matthew Do, that will now be referenced as "Method A" [2], and one primarily developed by Ryan Avancena, that will now be referenced as "Method B" [3].

2.1 Data Acquisition

The first step in data processing, regardless of whether it is for our assignment or any other project, is data acquisition. We need to have a dataset that can be used to train need based models. Our assignment needs two main datasets; one for stock information and one for news articles. For the ETF information, we utilized Python library yfinance [5] to download the data and stored it locally in JSON files. For the news articles, we utilized get_news() function of the yfinance library [5] to grab the URLs and then grabbed the articles text and information via HTML parser, modified from Dr. Ryu's [1] code. All the information grabbed is stored in appropriate JSON files to be utilized in model development. This enables us to have a defined set of data, which we can preprocess and utilize in our model development of sentiment analysis and investment returns. Figure 2 shows a sample snapshot of FNGU market data acquired from yfinance [5], and Figure 3 shows a sample snapshot of news data scraped from the web via our HTML scraper.

Bulk data tickers info:

Price	Adj Close								...	Volume							
Ticker	AAPL	AMZN	AVGO	CRWD	GOOG	META	MSFT	NFLX	...	AVGO	CRWD	GOOG	META	MSFT	NFLX	NOW	NVDA
Date																	
2013-01-02 00:00:00+00:00	16.705698	12.865500	2.395675	NaN	17.969599	27.915949	22.451811	13.144286	...	36716000	NaN	102033017	69866400	52899300	19431300	1525500	478836000
2013-01-03 00:00:00+00:00	16.494843	12.924000	2.408191	NaN	17.980036	27.686640	22.151047	13.798571	...	23295000	NaN	93075567	63140600	48294400	27912500	863100	298888000
2013-01-04 00:00:00+00:00	16.035383	12.957500	2.392730	NaN	18.335327	28.673666	21.736479	13.711429	...	27113000	NaN	110954331	72715400	52521100	17761100	1034400	524968000
2013-01-07 00:00:00+00:00	15.941059	13.423000	2.379478	NaN	18.255327	29.331686	21.695829	14.171429	...	16822000	NaN	66476239	83781800	37110400	45550400	2517000	610732000
2013-01-08 00:00:00+00:00	15.983952	13.319000	2.363281	NaN	18.219299	28.972767	21.582024	13.880000	...	20969000	NaN	67295297	45871300	44703100	24714900	2826800	466424000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2024-10-28 00:00:00+00:00	233.399994	188.389999	172.020004	301.320007	168.339996	578.159973	426.589996	749.119995	...	13195800	3464800.0	20858300	10925100	14882400	2862400	1238500	173586700
2024-10-29 00:00:00+00:00	233.669998	190.830002	179.240005	310.940002	171.139999	593.280029	431.950012	759.440002	...	23702300	3454800.0	28916100	13019100	17644100	3660400	1641100	157593600
2024-10-30 00:00:00+00:00	230.100006	192.729996	176.639999	307.450012	176.139999	591.799988	432.529999	753.739990	...	17911500	2083800.0	49698300	26864900	29749100	1722000	950800	179418100
2024-10-31 00:00:00+00:00	225.910004	186.399994	169.770004	296.869995	172.690002	567.580017	406.350006	756.030029	...	26086600	3475700.0	32801900	26838400	53971000	3057700	1333300	270039600
2024-11-01 00:00:00+00:00	222.910004	197.929993	168.919998	303.130005	172.649994	567.159973	410.369995	756.099976	...	17825100	2974300.0	21626900	15267400	24209400	2995800	985300	202737100

Figure 2: Sample snapshot of FNGU market data


```
{
    "0": {
        "ticker": "META",
        "date_published": "2024-11-02",
        "title": "Opinion: The 3 Best Tech Stocks to Own Ahead of 2025",
        "summary": "Don't sleep on these 3 stocks heading into 2025.",
        "link": "https://finance.yahoo.com/news/meta-platforms-third-quarter-2024-133838526.html?.tsrc=rss",
        "first_p": "Revenue: US$40.6b (up 19% from 3Q 2023).",
        "last_p": "Performance of the American Interactive Media and Services industry.",
        "whole_article": "Meta Platforms (NASDAQ:META) Third Quarter 2024 Results\nKey Financial Results\nReve
        in line with analyst estimates. Earnings per share (EPS) surpassed analyst estimates by 14%.\nLooking
        warning sign for Meta Platforms that you need to be mindful of.\nHave feedback on this article? Concer
        any stock, and does not take account of your objectives, or your financial situation. We aim to bring
    }
},{
    "1": {
        "ticker": "META",
        "date_published": "2024-11-02",
        "title": "Opinion: The 3 Best Tech Stocks to Own Ahead of 2025",
        "summary": "Don't sleep on these 3 stocks heading into 2025.",
        "link": "https://finance.yahoo.com/news/tech-earnings-fail-fire-traders-113001142.html?.tsrc=rss",
        "first_p": "(Bloomberg) -- Investors had hoped earnings from five of the world\u2019s biggest companie
        "last_p": "How to End the Backlog of Asylum Cases? Take Them Out of the Courts",
        "whole_article": "(Bloomberg) -- Investors had hoped earnings from five of the world\u2019s biggest co
        Hike, Breaking Campaign Vow\nInstead, in many cases they were left wanting.\nWhile Microsoft Corp., Ap
        Alphabet and Amazon ending the week in the green. The S&P 500 fell 1.4%, weighed down by the big-tech
        September, which would beat the estimate of 18% at the start of earnings season, according to data com
        intelligence dominated Magnificent Seven earnings this season. Amazon, Microsoft, Alphabet and Meta pu
        related demand is picking up steam, the message from management teams was that investors will need to
        significant costs. Microsoft\u2019s commercial cloud margin will narrow in the current quarter as capi
        told, technology, media and telecom stocks collectively saw the largest net selling in five weeks, acc
        course. Amazon, which has been dogged by concerns about pressure on margins from big spending, soothed
        to see,\u201d he said in an interview. \u201cThe AI winners will continue to perform.\u201d\nHowever,
        cheaper S&P 500 sectors that have outperformed big-tech stocks since July.\n\u201cThe story is still t
        Inflation Even Worse\nHow to End the Backlog of Asylum Cases? Take Them Out of the Courts\nHow Elon Mu
    }
```


Figure 3: Sample snapshot of scraped News data

2.2 Data Pre-Processing

2.2.1 Method A [2]

In this method, we started with utilizing word2vec [6] for embedding the articles' text. However, this did not work for us. That is, the model trained really poorly. We believe the reason to be that we are giving every word (or word $\rightarrow$ vec, which is deterministic) the same value across an article, which would freak out the training and converge everything to 0. This is not helpful.
We changed our direction after experimenting with word2vec [6]. We passed the scraped articles as input, while padding and truncating them at 580 characters. We pass this input set through bert-large-uncased tokenizer [7]. This allowed us to process the data and utilize it with existing functionalities of Bert [7] to develop a sentiment analyst model via bert-large-uncase [7]. More on this will be discussed in Section 2.3.

2.2.2 Method B [3]

In this method, local computing resources were insufficient. Hence, we worked and developed in Google Colab [8].This allowed us to utilize cloud computing capabilities of Google Infrastructure, reducing limitations introduced by local computing resources. We experimented with a different technique from "Method A" [2]. Main differences include, but are not limited to, focusing on a singular stock to start (ServiceNow), leaning into scikit-learn's MLP model [9], and utilizing Gensim's CBOW model [10] to generate text embeddings rather Bert [7]. We also utilized NLTK [11] to remove stopwords, cleaning up the web scraped news articles. We then split the cleaned articles' data via TimeSeriesSplit [12]. This data set was then tokenized via NLTK tokenizer [11], and then embedded via Gensim's CBOW [10].

2.3 Sentiment Analysis and Model Development

2.3.1 Method A [2]

In Section 2.2 we had tokenized our news articles (input data) via bert-large-uncased tokenizer [7]. We processed this tokenized data through bert-large-uncased [7] with an added layer of 256 neurons, 0.3 dropout as the last head, and early stopping enabled. We froze all weights except for the 256 neurons. The model trained for 129 epochs with 3.33 training loss and 0.053 validation loss, as shown in Figure 4.

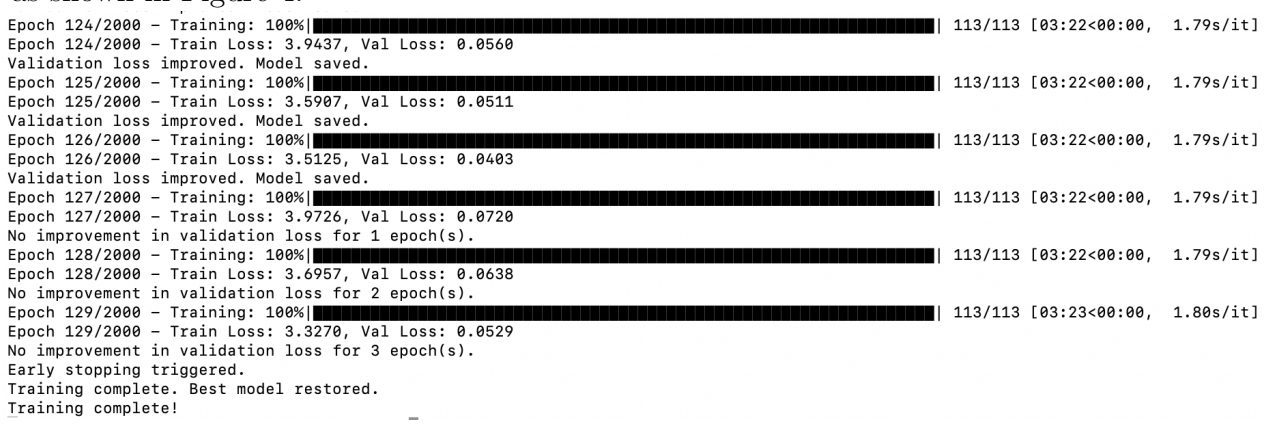

Figure 4: Bert Model Training

The output of the model is a prediction of % increase in returns from previous day, with labels of current open and yesterday open. The calculation for rate of return being $\frac{\text{current open - yesterday open}}{\text{yesterday open}}$.

2.3.2 Method B [3]

In Section 2.2 we had tokenized our dataset via NLTK tokenizer [11] and embedded it via Gensim's CBOW [10]. We processed this data through scikit-learn's MLP model [9], with two hidden layers, with the first hidden layer having 100 neurons and the second hidden layer having 50 neurons. We then set the model to train, and normalized open values to obtain a regression value. We will discuss metrics in Section 2.4.

2.4 Model Evaluation

2.4.1 Method A [2]

The model trained for 129 epochs with 3.33 training loss and 0.053 validation loss, as shown in Figure 4. Further evaluation and testing is to be completed in future work, as discussed in Section 3.

2.4.2 Method B [3]

We reviewed loss values post training of the model. We got mean squared error (MSE) with the value 0.0214 and mean absolute error (MAE) with the value 0.1277, which are quite good, albeit only for ServiceNow stock [13]. Further information on metrics is shown in Figure 5.

```
Mean Squared Error (MSE): 0.021436616206938774
Mean Absolute Error (MAE): 0.12766417823161075
R-squared (R2): -4.698978789233457
MAPE: 0.30264557605320436
```

Figure 5: Method B Metrics

We also plotted a trends graph of normalized data, over time, as discussed in Section 2.3. This was done in order to be used a reference for future model enhancement and usage to determine whether our model is predicting buy vs sell in line with the trends. This graph is shown in Figure 6. We also attempted to run predictions on the built model through some test cases, for which the results are shown in Figure 7.
Though the model is not complete, it is showing promising results, with next steps being to expand the model to all the stocks in our dataset instead of just ServiceNow [13].

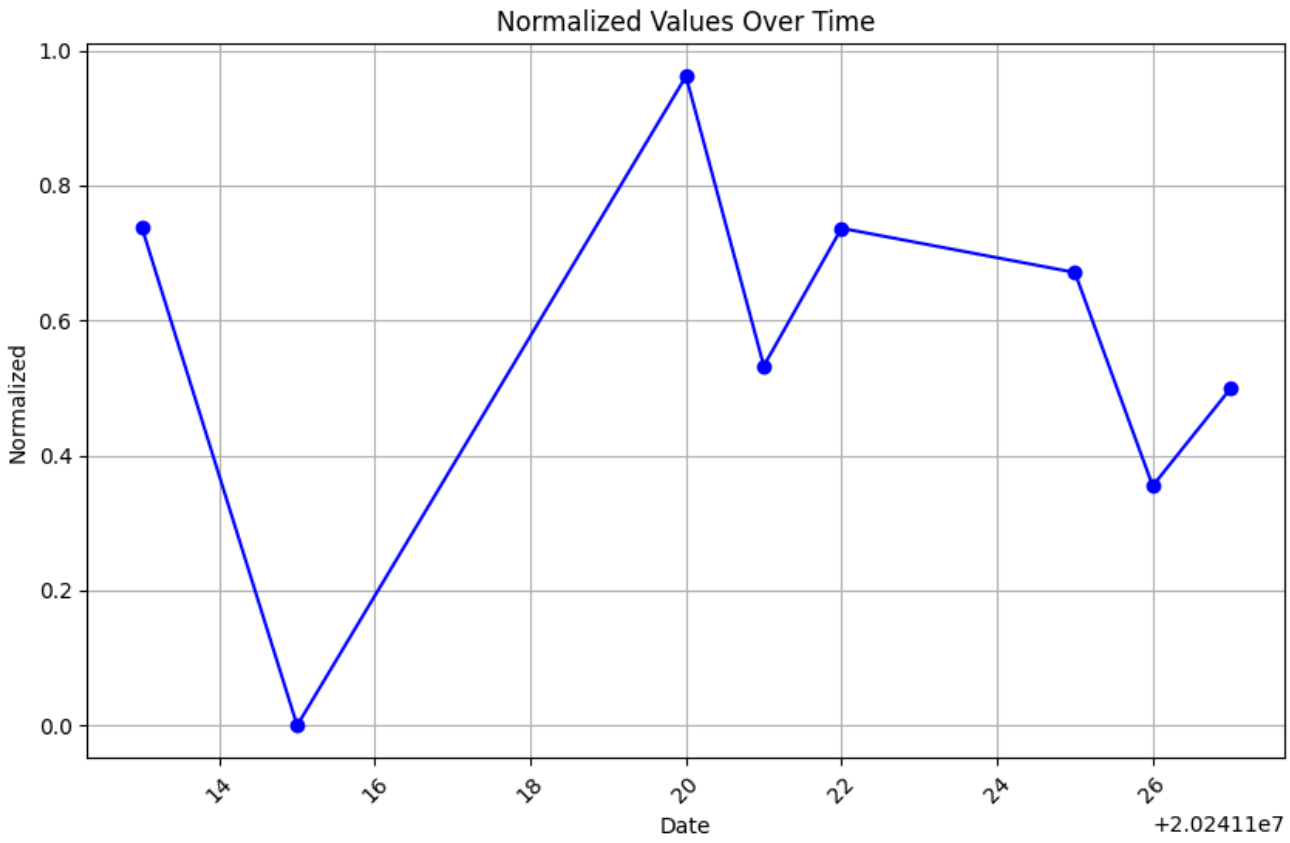

Figure 6: Method B Normalized Data Trends

```
# X_test is a list of embeddings
test_case = "ServiceNow, Inc. (NOW): Needham Maintains $1,150 Price Target, Highlights GenAI Growth Potential with Xanadu."
test_case = cleanArticle(test_case)
test_case_embedding = compute_embedding(test_case, model)
test_case_embedding = np.array(test_case_embedding).reshape(1, -1)

predicted_value = mlp.predict(test_case_embedding)
print(f"Predicted value: {predicted_value[0]}")
```

```
Predicted value: 0.6699533462524414
```

```
# X_test is a list of embeddings
test_case = "ServiceNow (NOW) Is Considered a Bad Investment by Brokers: Is That True?."
test_case = cleanArticle(test_case)
test_case_embedding = compute_embedding(test_case, model)
test_case_embedding = np.array(test_case_embedding).reshape(1, -1)

predicted_value = mlp.predict(test_case_embedding)
print(f"Predicted value: {predicted_value[0]}")
```

```
Predicted value: 0.6082693934440613
```

Figure 7: Method B Prediction Tests

3 Conclusion and Future Work

For this assignment, it took us quite a bit of time to gather data. Additionally, it took us time to understand what technology and tools to utilize to process the data and develop a model that could provide us sentiment analysis. Though we were able to develop a model, it is not deployment

ready. Our future work is to utilize the data, that we spent a lot of time to gather, and the existing development to better analyze news data in real-time. This work could then be further enhanced with the trends data and models that stock market analysts use in real-time to forecast market trends. Having such a model would provide tremendous value in better understanding and staying up-to-date with market trends.

4 Acknowledgements

We would like to acknowledge Dr. Jin [14] for ML, classification and probability distribution concepts, Professor Avery [15] for MLP concepts, and Dr. Ryu [1] for scraper, stock information downloading process and assignment information.

References

[1] C. Ryu, California State University, Fullerton, https://www.fullerton.edu/ecs/cs/faculty/.

[2] M. Do, "Method A", methodology Developed Primarily by Matthew Do.

[3] R. Avancena, "Method B", methodology Developed Primarily by Ryan Avancena.

[4] scikit learn, "train_test_split," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[5] Y. Finance, "yfinance," https://pypi.org/project/yfinance/.

[6] O. Source, "word2vec," https://www.tensorflow.org/text/tutorials/word2vec.

[7] G. R. Developers, "bert," https://github.com/google-research/bert.

[8] G. Developers, "Google Colab", https://colab.research.google.com/.

[9] scikit learn, "MLPClassifier", https://scikit-learn.org/dev/modules/generated/sklearn.neural_network.MLPClassifier.html.

[10] O. Source, "Gensim Embeddings", https://radimrehurek.com/gensim/models/word2vec.html.

[11] N. Team, "NLTK", https://www.nltk.org/.

[12] scikit learn, "TimeSeriesSplit", https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html.

[13] ServiceNow, https://www.google.com/finance/quote/NOW:NYSE?hl=en.

[14] R. Jin, California State University, Fullerton, https://www.fullerton.edu/ecs/cs/faculty/.

[15] K. Avery, California State University, Fullerton, https://www.fullerton.edu/ecs/cs/faculty/.