NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AI for Finance

By: Phoe Chuan Bin (U1821679J)

Supervisor: A/P Erik Cambria

Examiner: Prof. Mo Li

Submitted in Partial Fulfilment of the Requirements for the Degree of Bachelor of Engineering in Computer Science of Nanyang Technological University

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

2021/2022

Abstract

The rise in prominence of cryptocurrencies have led to increased volatility and trading in cryptocurrency exchanges. Financial institutions are now embracing the use of alternative data especially towards social media commentary to increase their investment returns. The rationale is based on behavioural finance which proved that financial decisions are significantly driven by emotion and mood. As such, sentiment analysis of financial microblogs have been getting increased attention.

In this project, I will be leveraging on the use of Text Mining and NLP techniques to better predict the financial sentiment of social media cryptocurrency content. We will take both Symbolic and Sub Symbolic approaches in tackling this problem using lexicons and learning-based language models respectively. Our results show that the proposed final hybrid architecture outperforms individual lexicons in the current literature and state-of-the-art deep learning methods for this sentiment classification problem.

Acknowledgements

I would like to take this opportunity to express my utmost gratitude towards my project supervisor, **Associate Professor Erik Cambria**, for his guidance, his helpful suggestions and also his valuable advice during the project despite his busy schedule. His rich experience in Natural Language Processing and Sentiment Analysis have greatly benefitted me in completing this project.

I would also like to thank my examiner, **Professor Mo Li,** for grading this Final Year Project.

I am also thankful towards the **SenticNet Team**, for providing various Application Programming Interfaces (APIs) to assist in my experimentation and research in this project.

Lastly, I would like to thank my family and friends who have consistently supported and encouraged me throughout the project.

Table of Contents

Table of Figures

1. Introduction

Background

Rise in Cryptocurrency Trading

In March 2022, the 2 global cryptocurrencies, measured in terms of market capitalisation, had a combined markets value of $1.96 Trillion with a daily traded volume of ~$100 Billion. [1] It was estimated that the 221 million people traded cryptocurrency last year which is a stark contrast to the early 2010s when Bitcoin, the first cryptocurrency created, was still an obscure financial product. [2]

As an emerging market, cryptocurrency trading has seen considerable progress and a notable upturn in interest and activity. [3] The huge volatility in prices – for example in 2017, the value of Bitcoin increased 2000% and by 8 weeks later the price had been more than halved [4], is ideal for traders as it presents a huge opportunity for them to capitalise on short-term price movements. [5]

Social Media as source of Alternative Data

Research has shown that in addition to information, emotions play a significant role in human decision-making [6], [7], [8]. Behavioural finance has provided further proof that financial decisions are significantly driven by emotion and mood. [9] As such, financial institutions have started using Alternative Data, defined as non-traditional data that can provide an indication of future performance of Financial Markets, to enhance their investment returns. [10] Social Media Commentary is one of the key sources of Alternative Data, in addition to satellite imagery and GPS data. [11]

Text Mining from the Web

Text mining which usually manifests in the form of Web Scraping, refers to a technique in which a computer program extracts data from extract valuable information from a website. [12] This way, huge amounts of social media data can be collected without much usage of user's time, effort and money since it's an automatic process performed by bots. [13]

Emergence of Deep Learning Solutions

In recent years, Deep Neural Network (DNN) models have seen significant success in the performance of NLP-related tasks. In particular, with the advent of pre-trained generalized language models, we now have methods for transfer learning to new tasks with massive pre-trained models like GPT-2, BERT, and ELMO. These models are doing real work in the world, such as language translation and email filters. [14]

Project Objective and Scope

The main aim of this project is the leverage on the use of Text Mining and NLP techniques to better predict the financial sentiment of social media cryptocurrency content. This can be split into 2 smaller subtasks:

1. Text Mining of useful Social Media Content
2. Financial Sentiment Classification

Report Organisation

The report is organised as follows:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: Datasets
- Chapter 4: Lexicon Construction
- Chapter 5: Symbolic Approach
- Chapter 6: Sub Symbolic Approach
- Chapter 7: Sub Symbolic Approach
- Chapter 8: Overall Conclusion and Future Works
- Chapter 9: References
- Chapter 10: Appendix

2. Literature Review

Lexicon-based vs Deep learning methods

There are 2 main methods used in sentiment analysis – lexicon-based methods and deep learning methods. The increased performance of sentiment analysis methods has greatly increased in recent years due to the use of various models based on the Transformer architecture. However, it is known that these black-box models are difficult to train and are not easily interpretable. On the other hand, lexicon-based methods are fast, requires no training and are well-interpreted albeit not performing as well as deep learning models. [15]

Lexicon-based method

In lexicon-based sentiment analysis, documents are assigned labels by calculating the overall sentiment from the semantic orientation of word or phrases that occur in the text. With this approach a dictionary of positive and negative words is required, with a positive or negative sentiment value assigned to each of the words. Different approaches to creating dictionaries have been proposed, including manual [16] and automatic [17] approaches. Generally speaking, in lexicon-based approaches a piece of text message is represented as a bag of words. Following this representation of the message, sentiment values from the dictionary are assigned to all positive and negative words or phrases within the message. A combining function, such as sum or average, is applied in order to make the final prediction regarding the overall sentiment for the message.

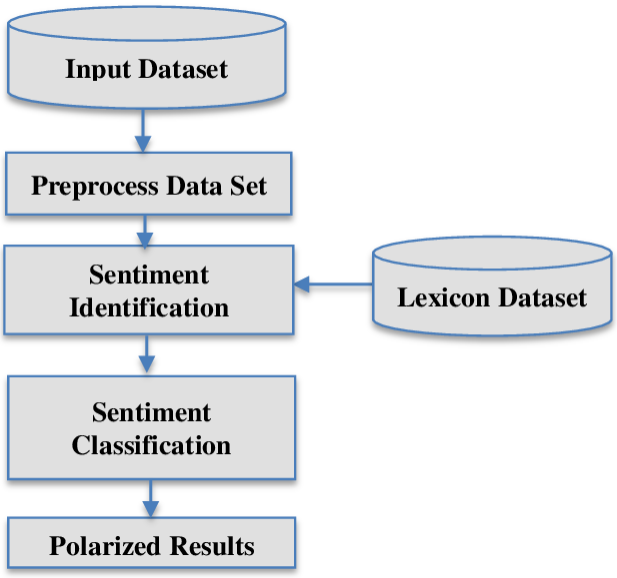

Figure 1: Lexicon-based approach for Sentiment Analysis [18]

SenticNet Polarity Classification API

This API leverages neurosymbolic AI to assign contextual polarity to concepts in text using SenticNet and to flux such polarity through dependency arcs in order to assign a final polarity label to each sentence via sentic patterns. The expected input of this API is a piece of text (a sentence or a paragraph) and the output is one of the following labels: POSITIVE, NEGATIVE, or NEUTRAL. [19]

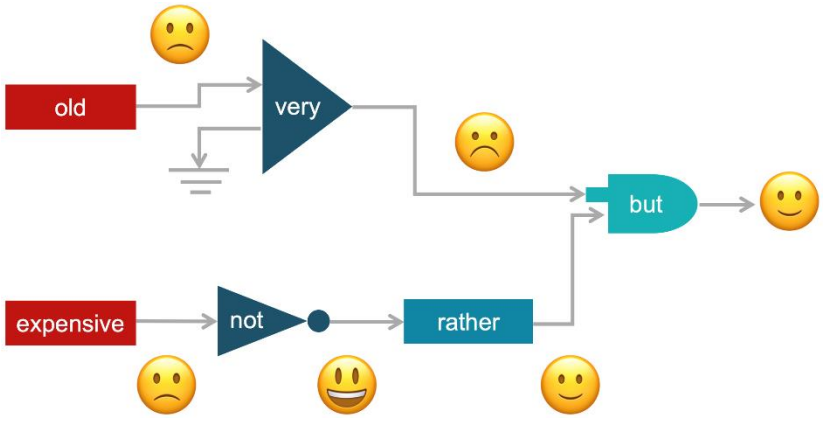

Figure 2: SenticNet Polarity Classification API [19]

NTUSD-Fin

NTUSD-Fin is a Market Sentiment Dictionary for Financial Social Media Data Applications which comprise of 8,331 words. It provides various scoring methods including frequency, CFIDF, chi-squared value, market sentiment score and word vector for the tokens.[20] The market sentiments score is calculated by:

$$\text{MarketSentimentScore(s)} = \text{BullishPMI(s)} - \text{Bearish PMI(s)}$$

Bullish						Bearish					
Word		Hashtag		Emoji		Word		Hashtag		Emoji	
dominant	1.22	buy	1.27	☼	1.14	bagholders	3.46	mnkdsecinvestigation	3.53	▼	4.42
bully	1.21	early	1.27	🎄	1.14	junk	3.45	pennymikey	3.53	🔻	3.68
updates	1.21	gainers	1.27	🐂	1.14	overvalued	3.42	noshamenate	3.53	💩	3.53
runner	1.21	mattel	1.27	💎	1.14	pumpers	3.37	sanofiwasright	3.53	🐻	2.59
binance	1.21	oprah	1.27	$	1.14	scam	3.34	scambags	3.53	🎯	2.01
blast	1.21	shortsqueeze	1.27	📍	1.14	garbage	3.32	short	3.39	😒	1.82
floater	1.21	analyis	1.27	🚁	1.14	pig	3.25	scam	3.29	😐	1.82
undervalued	1.21	biotech	1.27	🤘	1.14	trash	3.25	forex	3.04	😢	1.67
accumulating	1.20	blocks	1.27	🤓	1.14	turd	3.19	market	2.75	😬	1.57
blackberry	1.20	boolish	1.27	🤑	1.14	puts	3.11	elliottwave	2.40	😤	1.52

Figure 3: NTUSD-Fin Top 10 tokens ranking with PMI for Bullish and Bearish sentiments [20]

SentiWordNet

SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset, group of synonyms, of WordNet three sentiment scores: positivity, negativity, objectivity. It has a wide coverage (all synsets are tagged) and provides fine-grained numerical scores. [21]

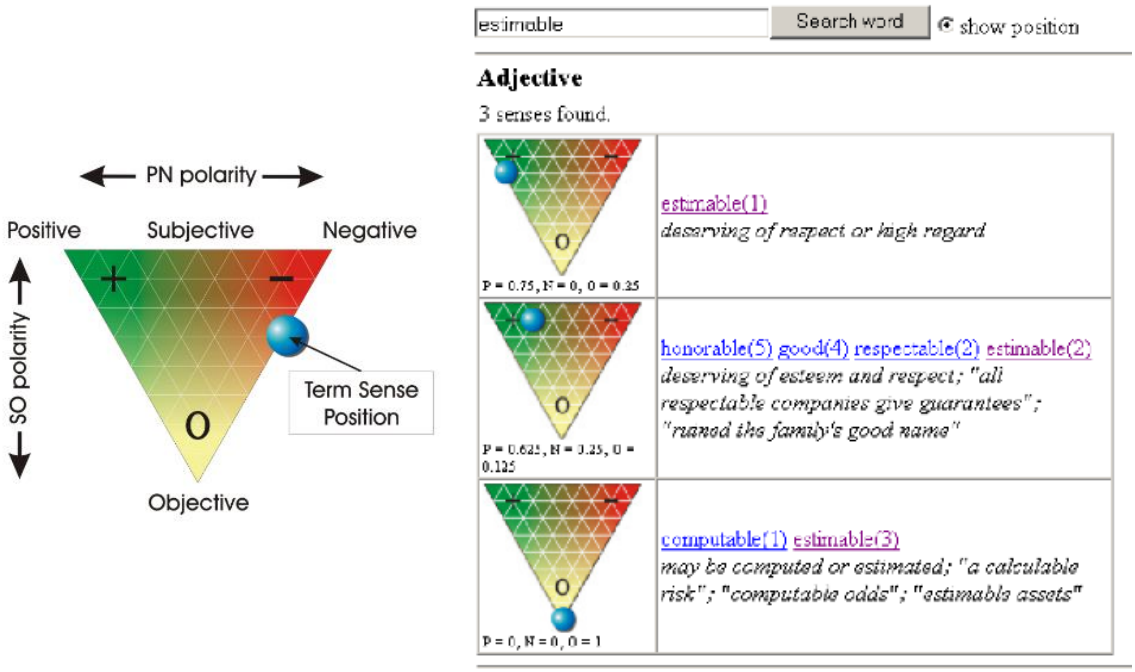

Figure 4: SentiWordNet graphical representation for representing opinion-related properties [21]

Afinn

The AFINN lexicon is a list of 2,477 English terms manually rated for valence with an integer between -5 (negative) and +5 (positive) by Finn Årup Nielsen between 2009 and 2011. There are 878 positive and 1,598 negative terms. It is specifically constructed for the language prevalent in microblogs. [22]

Vader

Valence Aware Dictionary and sEntiment Reasoner (Vader) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. The lexicon is a simple rule-based model for general sentiment analysis and is empirically validated by multiple independent human judges. [23]

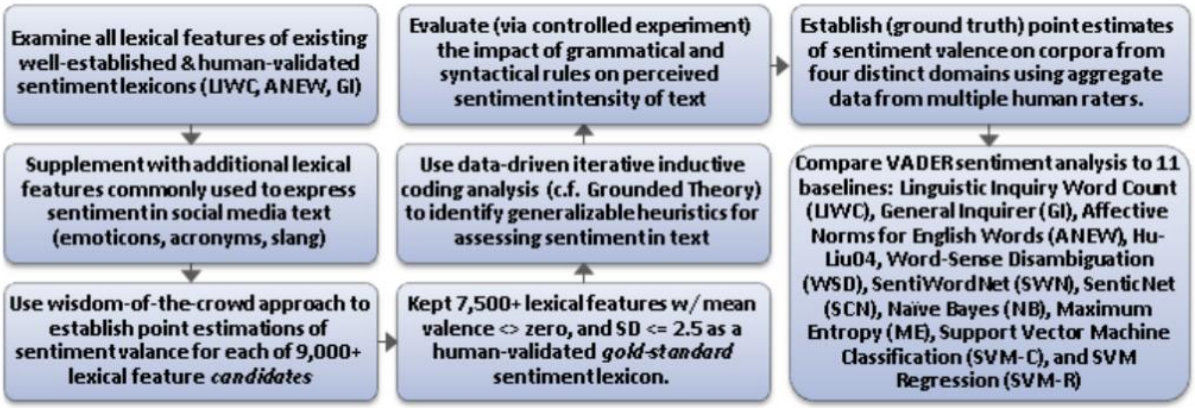

Figure 5: Vader methods and process approach overview [23]

Deep learning

Neural networks are trained via the Backpropagation algorithm using Gradient Descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights and repeatedly adjusts the weights of the connections to minimize the cost function. [24]

Recurrent Neural Networks

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. [25]

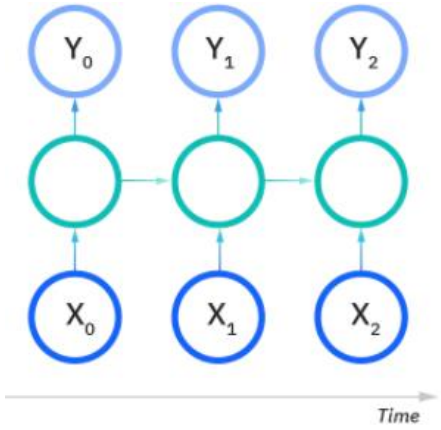

Figure 6: Recurrent Neural Network (Unrolled) [25]

While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network.

LSTM

Vanishing Gradient Problem

The LSTM model is unique as it avoids the problem vanilla RNNs face - the Vanishing Gradient Problem. There are 2 main steps in training the model:

1. Feed forward to generate a prediction based on the input and weights on each node connection
2. Backpropagation to use the error (Difference between the predicted and actual signals) to tune the various above-mentioned weights

The Vanishing Gradient Problem occurs in step 2 where earlier layers are unable to “learn”. This can be visualised in the diagram below. After multiple layers of backpropagation of weights preceding it, the weight received in the earlier layers will be too small and negligible to be tuned. As a result, RNNs have “short term” memories.

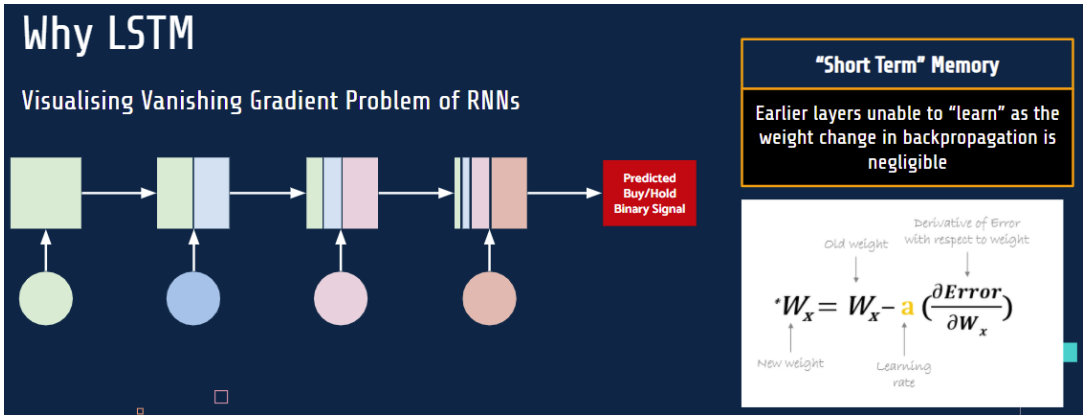

Figure 7: Vanishing Gradient Problem

To bypass this problem, LSTM adds on a cell-state highway connection between cells that acts as the "long term" memory of the network in addition to a 3-gated cell structure to affect the cell-state and the hidden-state at each timestep.

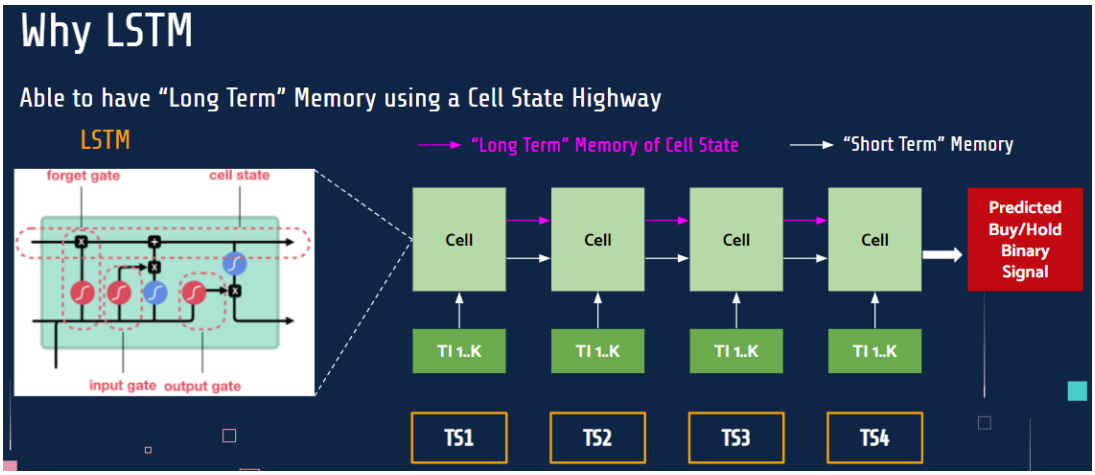

Figure 8: LSTM cell-state highway

An LSTM cell structure has five essential components consisting of three gates and two cell states. Each gate contains a sigmoid activation function that will either remove or keep the data. This is summarised in the diagram below.

Gates

1. The forget gate selects the information that is required for learning. It determines the amount of information to be allowed from the current input and previous cell state to the current cell state.
2. The input gate determines the additional information from the current input to be added to the cell state.
3. The output gate passes the information from the previous input stage to determine what information should be carried over to the next hidden stage.

States

1. The cell state contains the short-term memory and long-term memories of the cell.
2. A hidden state is an output state that is formulated from the current input and previous hidden state. It retrieves the memory stored in the cell state to be used for the next prediction.

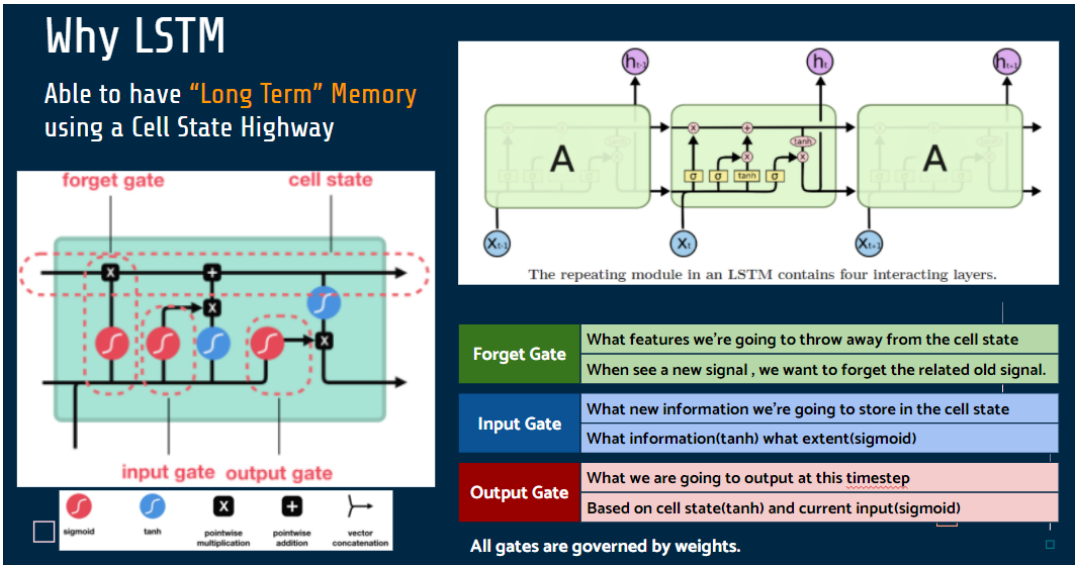

Figure 9: LSTM gates

Due to its unique cell structure, LSTM is relatively unaffected by the vanishing gradient problems.

GRU

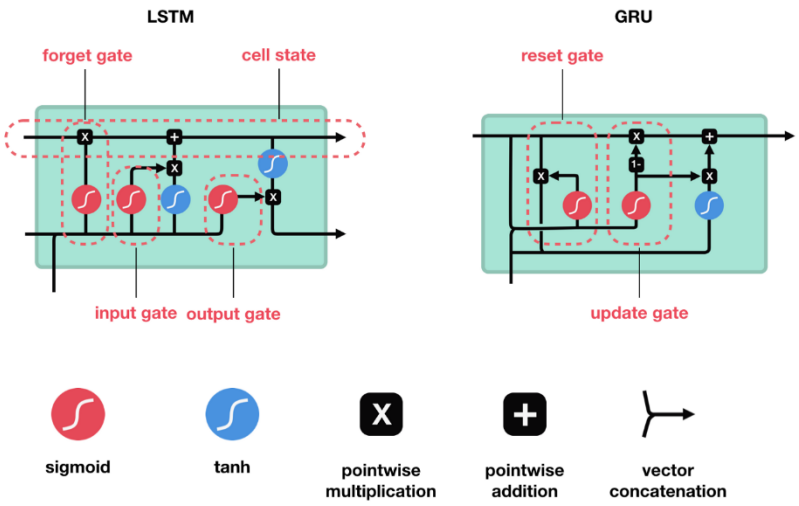

Figure 10: Comparison between LSTM and GRU [26]

The GRU is the newer generation of Recurrent Neural networks and is very similar to an LSTM in avoiding the Vanishing Gradient Problem. GRU's got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.

Gates

1. The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.
2. The reset gate is another gate is used to decide how much past information to forget.

Since GRU has fewer tensor operations; therefore, they are a little faster to train then LSTM.

Transformer Architecture

Attention

The transformer architecture relies of a self-attention mechanism to convert input sequences into output sequences without the need to be sequentially aligned like LSTM. Specifically, neither the encoder nor the decoder used any recurrence or looping, like traditional RNNs. Instead, they used layers of "attention" through which the information passes linearly. It

didn’t loop over the input multiple times – instead, the Transformer passes the input through multiple attention layers.

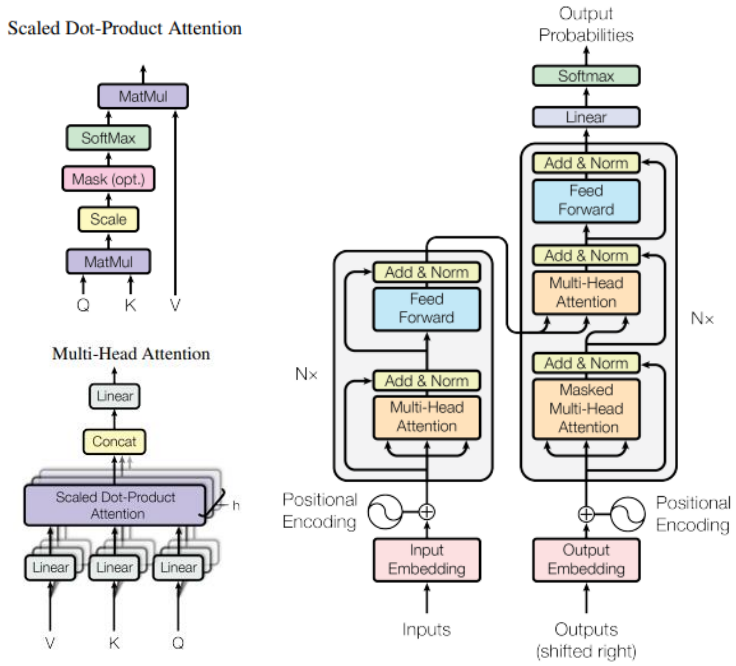

Figure 11: Transformer Architecture [27]

Encoder and Decoder

Encoder: Processes the input text, looks for important parts, and creates an embedding for each word based on relevance to other words in the sentence. (Left portion)

Decoder: Takes the output of the encoder, which is an embedding, and then turns that embedding back into a text output (Right portion)

BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers. Historically, language models could only read text input sequentially -- either left-to-right or right-to-left -- but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. The transformer is the part of the model that gives BERT its increased capacity for understanding context and ambiguity in language. The transformer does this by processing any given word in relation to all other

words in a sentence, rather than processing them one at a time. By looking at all surrounding words, the Transformer allows the BERT model to understand the full context of the word. [28]

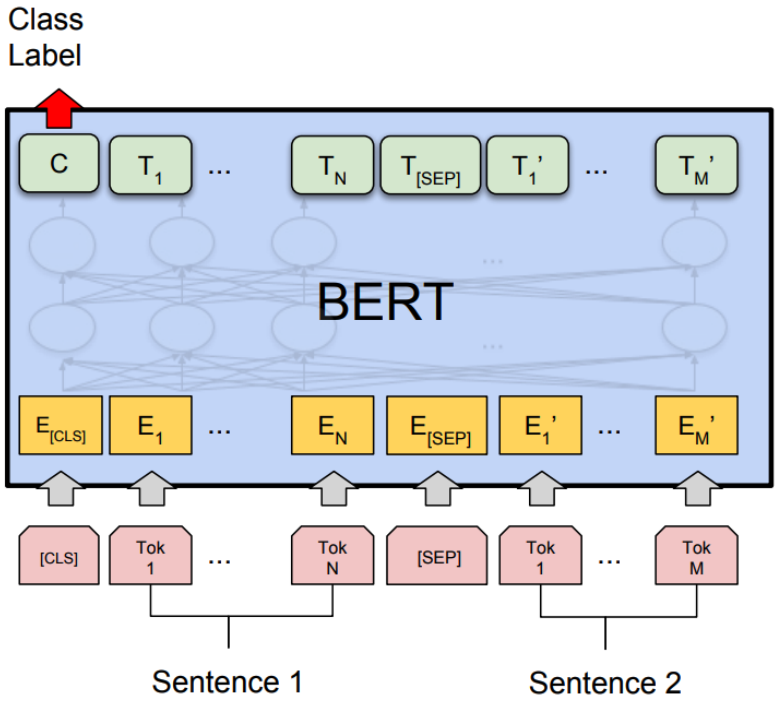

Figure 12: Bert Architecture [29]

Performance Metrics

The following are the various performance metrics that will be used in the following sections to compare the results across various proposed methods in a standardised way.

		Actual	
		Positive	Negative
Predicted	Positive	**True Positive**	**False Positive**
	Negative	**False Negative**	**True Negative**

Figure 13: Performance Metrics in Confusion Matrix

Accuracy

Accuracy simply measures how often the classifier correctly predicts. [30]

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision

Precision explains how many of the correctly predicted cases actually turned out to be positive. [30]

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall explains how many of the actual positive cases we were able to predict correctly with our model. [30]

$$Recall = \frac{TP}{TP + FN}$$

F1 Score

F1 Score gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall. [30]

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Loss

Loss is a value that sums up the errors in the trained model. It measures how well or bad a particular model is doing. Log loss is a pretty common evaluation metric for binary classifiers, and it is sometimes the optimization objective for Neural Networks. [30]

$$Binary\ Log\ Loss = -(y\ log(p) + (1 - y) \log (1 - p)$$

Area under the Curve (AUC)

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes. The greater the AUC, the better is the performance of the model at different threshold points between positive and negative classes. [30]

Receiver Operating Characteristics Curve (ROC Curve)

The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR (True Positive Rate) against the FPR (False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'. [30]

Cohen's Kappa Coefficient

Cohen's Kappa Coefficient measures interrater reliability. [31]

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

$$where\ p_o\ is\ the\ observed\ agreement$$

$$and\ p_e\ is\ the\ expected\ agreement$$

3. Datasets

In this project, I will be using 4 datasets for the purposes of creating new lexicons, training our sub symbolic models and testing our various hypothesis.

All datasets used in this project can be found under *data* folder of the submitted codebase. A helper python library *data_reader.py* was also created for the ease of use for our downstream tasks. Its methods can be called in other Jupyter notebooks to read in and clean the data sources so that they are ready to be used.

Rationale of using multiple datasets

1. Using multiple datasets as test sets helps us to ensure that our proposed solution is generalised and not specific to a particular dataset.
    a. Data from StockTwits and Twitter may differ in length due to the limits of the platform (1000 and 280 characters respectively).
    b. Furthermore, users of different platforms may use different lingos that are not present in others.

2. Some of the datasets will be used to create new lexicons or train neural networks and as such cannot be used as test sets to ensure the legitimacy of our experiments.

Data 1

Source

Data 1 is sourced from the SemEval 2017, the 11th international workshop on semantic evaluation. More specifically, we will be using the Task 5 dataset which consists of financial-related microblog messages and their corresponding sentiment scores for the companies/stocks mentioned.


```
{
  "tweet": "downgrades $SON $ARI $GG $FLTX $WMC $MFA
    $IVR $CMI $PCAR $QLIK $AFOP $UNFI #stocks
    #investing #tradeideas",
  "target": "$PCAR",
  "snippet": "downgrade",
  "sentiment": -0.463
},
{
  "tweet": "$AMZN looking sexy this morning...$600
    break on volume and it's gone.",
  "target": "$AMZN",
  "snippet": [
    "looking sexy this morning",
    "break on volume"
  ],
  "sentiment": 0.678
},
```


Figure 14: SemEval 2017 Task 5 Dataset

The microblogs messages come from 2 sources – StockTwits and Twitter which have been annotated for fine-grained sentiment. Sentiment values are floating point values in the range of -1 (very negative/bearish) to 1 (very positive/bullish).

These files are stored as *training_set.json* and *test_set.json* under *data1* folder.

Data processing in data_reader.py

```
def read_data1(mode, to_clean= True):

    cwd = os.getcwd()
    train_path = cwd + DATA_1_FILEPATH[0]
    test_path = cwd + DATA_1_FILEPATH[1]

    with open(train_path, "r") as f:
        data = f.read()
        data1 = json.loads(data)

    with open(test_path, "r") as f:
        data = f.read()
        test_data = json.loads(data)

    data1.extend(test_data)

    if(to_clean):
        data1 = preprocess_data(pd.DataFrame(data1), "tweet")

    if(mode=="dataframe"):
        data1_X = [item.lower() for item in data1['tweet']]
        data1_y_class = [1 if float(item)>0 else 0 for item in data1["sentiment"]]

        data1 = pd.DataFrame()
        data1["text_cleaned"] = data1_X
        data1["Label"] = data1_y_class
        return data1

    elif(mode=="list"):
        data1_X = [item.lower() for item in data1['tweet']]
        data1_X = to_tokens(data1_X)
        data1_y_class = [1 if float(item)>0 else 0 for item in data1["sentiment"]]

        return data1_X, data1_y_class
```

Figure 15: Function to read Data 1

Both training and test data are combined, pre processed and returned to the caller as a DataFrame or X and y lists based on the input parameters. Data 1 consist of 2023 rows.

	text_cleaned	Label
0	downgrades stocks investing tradeideas	0
1	looking sexy this morning break on volume and ...	1
2	stock hasnt moved much since first few weeks a...	1
3	whole foods may feel price competition but wil...	1
4	apples iphone se could be doing better than ex...	1
...	...	...
2018	hold through all the media bs and you will be ...	1
2019	i love blood like a vampirekeep bleeding for me	0
2020	yea dude airplanes are still flying	1
2021	ten year yield weekly year yield still very be...	1
2022	the best scenario going forward is this stock ...	0

2023 rows × 2 columns

Figure 16: Data 1

Data 2

Source

Data 2 is sourced from IEEE Data Port and was submitted by Bruno Taborda. This data was downloaded from Twitter using the Twitter REST API search. Tweets were filtered out for the English language. The following Twitter tags were used at the search parameter.

Category	**Search Parameter**
S&P500 related	#SPX500, #SP500, SPX500, SP500, $SPX
Top 25 companies in the index	$MSFT, $AAPL, $AMZN, $FB, $BBRK.B, $GOOG, $JNJ, $JPM, $V, $PG, $MA, $INTC $UNH, $BAC, $T, $HD, $XOM, $DIS, $VZ, $KO, $MRK, $CMCSA, $CVX, $PEP, $PFE
Bloomberg tag	#stocks

Figure 17: Data 2 Search Parameters

We will be using the labelled subset (1,300 tweets) of the whole dataset (938 672 tweets) which was manually annotated by the author and reviewed by a second independent annotator.

```
id;created_at;text;sentiment
77522;2020-04-15 01:03:46+00:00;"RT @RobertBeadles: Yo
Enter to WIN 1,000 Monarch Tokens

US Stock Market Crashes &amp;amp; what we can LEARN from them PT3!

RETWEET, WATCH video…";positive
661634;2020-06-25 06:20:06+00:00;"#SriLanka surcharge on fuel removed!

The surcharge of Rs.26 imposed on diesel and petrol has been revoked with effect from midnight

#lka #FuelPrices #taxes #economy #stocks #StockMarket";negative
```

Figure 18: Data 2 Raw Form

The file is stored as *tweets_labelled_09042020_16072020.csv* which is separated by ";" under *data2* folder.

Data processing in data_reader.py

```
def read_data2(mode, to_clean= True):

    cwd = os.getcwd()
    path = cwd + DATA_2_FILEPATH

    data2 = pd.read_csv(path, sep=";")
    data2 = data2[data2['sentiment'].notna()]
    data2 = data2[data2['sentiment']!='neutral']

    if(to_clean):
        data2 = preprocess_data(data2, "text")

    if(mode=="dataframe"):
        data2_X = [item.lower() for item in data2['text']]
        data2_y_class = [1 if item=="positive" else 0 for item in data2["sentiment"]]

        data2 = pd.DataFrame()
        data2["text_cleaned"] = data2_X
        data2["Label"] = data2_y_class
        return data2

    elif(mode=="list"):
        data2_X = [item.lower() for item in data2['text']]
        data2_X = to_tokens(data2_X)
        data2_y_class = [1 if item=="positive" else 0 for item in data2["sentiment"]]

        return data2_X, data2_y_class
```

Figure 19: Function to read Data 2

Data that was not labelled or labelled as “neutral” were removed before returning to the caller as a DataFrame or X and y lists based on the input parameters. Data 2 consist of 875 rows.

	text_cleaned	Label
0	rt yo enter to win monarch tokens us stock mar...	1
1	srilanka surcharge on fuel removed the surchar...	0
2	net issuance increases to fund fiscal programs...	1
3	rt how much of amazons traffic is served by fa...	1
4	ryzen desktop cpus looking great and on track ...	1
...	...	...
870	q eps estimates for ball co increased by analy...	1
871	stocks back from the recovery room fair value ...	1
872	rt breadth expanding last weeks discussion mor...	1
873	top may now be in	1
874	glg partners lp short position in hilton food ...	0

875 rows × 2 columns

Figure 20: Data 2

Data 3 and 4

Source

Data 3 and 4 are StockTwits that have been scrapped manually by calling the StockTwit API. The code can be found under *data_scraper.ipynb*.

Reason of choosing StockTwits

Since 2013, users of the platform were able to add sentiment tags to the message that they post. Although it might not be entirely accurate, I believe that this presents 2 advantages:

1. Lending the concept from the Law of Large Numbers, an observed sample average sentiment value of a token from a large sample will be close to the true population average. As such, by scraping huge amounts of sentiment-tagged data, we are able to aggregate a pool of natural language syntax to build our proposed lexicons on with a good chance of reflecting the true "sentiment" of various tokens.

2. Manual annotations require huge amount of time and human effort which are not easily available. Furthermore, sentiments are subjective in general and thus manual annotations needs to be reviewed to prevent bias, adding to the costs. This solution of scraping data will allow for the automatic collection of big data from a large pool of independent "annotators" and alleviate the problems mentioned above.

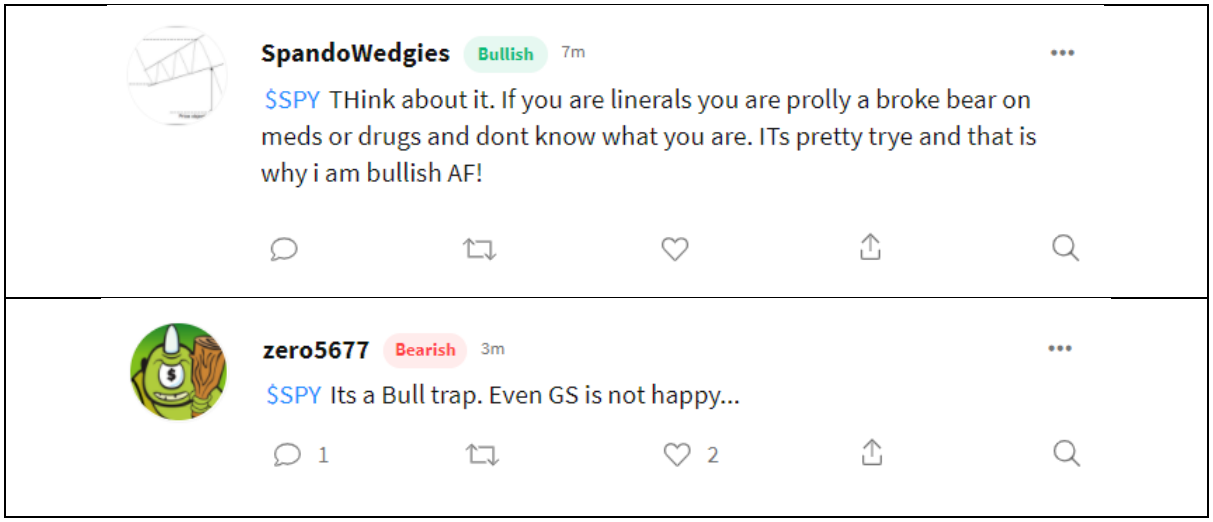

Figure 21: Stocktwits sentiment tags

Tickers to Scrape

Since the API can only return results based on searched symbols, a list of suitable financial tickers have been chosen to collect appropriate StockTwits for our task. This consists of tickers that track the General Market and Cryptocurrencies sentiment.

Category	**Tickers**
General Market	SPY – ETF that tracks S&P500 DJIA – Dow Jones Industrial Average QQQ – ETF that tracks NASDAQ VIX – Volatility Index
Cryptocurrency	BTC.X – Bitcoin ETH.X – Ethereum XRP.X – Ripple ADA.X – Cardano SOL.X - Solana

Figure 22: Tickers to scrape

Since the United States has the world's largest stock markets, we have chosen the 3 major indexes to get a gauge on the General Market sentiment. In addition, we added in the CBOE Volatility Index which represents the market's expectations for the relative strength of near-term prices changes of the S&P500. This is used by investors to quantify the level of risk, fear, or stress in the market to influence their investment decisions.

Cryptocurrency	**Market Cap**	**Watch Count**
BTC.X	$815,054,267,576	471,267
ETH.X	$362,296,101,136	247,670
XRP.X	$40,752,748,447	137,286
ADA.X	$32,154,033,473	109,620
SOL.X	$29,402,629,996	34,191

Figure 23: Statistics of Top 5 cryptocurrencies

We chose the top 5 cryptocurrencies based on their coin market cap and number of users who watch the coin on the StockTwits platform. We excluded coins that are pegged to the US Dollar.

Scraping Process

To perform the scraping process, we call URL_START using *requests.get()*, inputting the desired symbol into the curly braces.

URL	*https://api.stocktwits.com/api/2/streams/symbol/{}.json?max={}&filter=top*
URL_START	*https://api.stocktwits.com/api/2/streams/symbol/{}.json?filter=top*

Figure 24: URLs to send get requests to

We process the json response using the function *process_messages()* where we only store messages where sentiments are present. Additional details like User_id, Date, Time were also processing and collected. We will return the last User_id back to the caller for the reason below.

Since results are returned in batches of 30, we have to iteratively call URL to get all the data we want. To prevent receiving StockTwits which we have seen before, we have to store the last seen StockTwit ID and pass it into the *max* parameter in the URL for the next iteration. This process is carried out until we do not have any more results left to receive or our target count of StockTwits per symbol has been collected.

Duplicates were dropped post-scraping via *pandas.dropna()*.

Data 3

We collect approximately 200 StockTwits per symbol. Then, I manually annotate each tweet's sentiment values to be 0 (negative) or 1 (positive) and removed those that are deemed as neutral. Data 3 is saved as *data3_final.csv* under *data 3* folder. *The* total row count is 1161.

	text_cleaned	Label
0	im in this because it just doesnt seem logical	1
1	why saudi is dumping the petrodollar for petro...	0
2	bulls put up a good fight	0
3	fed is going to need bps rate hikes every week...	0
4	approaching midday and only at a third of norm...	0
...	...	...
1156	while all other cryptos are rising and shining...	0
1157	glad i sold this crap hbar	0
1158	bull trap	0
1159	team bear	0
1160	looks like a long weekend for the hodlers	0

1161 rows × 2 columns

Figure 25: Data 3

As a proxy to measure the inter-rater reliability of Data 4, I calculated the Cohen's Kappa Coefficient based on the tweet's actual sentiment values (Bearish/Bullish) with what I actually annotated them with. According to *statogy.org*, a score of 0.855 puts the annotations to be "near perfect agreement" which proves the reliability of using the crowd-sourced sentiment values from StockTwits.

Cohen's Kappa	Interpretation
0	No agreement
0.10 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 0.99	Near perfect agreement
1	Perfect agreement

Annotator	**Annotator**	**Cohen's Kappa Coefficient**
StockTwit's Sentiment	Manual Annotation	0.8554340

Figure 26: Cohen's Kappa Coefficient

Data 4

We collect approximately 110,000 StockTwits per symbol and saved in the format of *stocktwits_data_<symbol>.csv* under *data4* folder. They are then combined and saved as *stocktwits_data_ALL.csv*.

	Sentiment	User_id	Message	Date	Time	Symbol
0	Bearish	442859953	$SPY crypto is even worthless if there is no w...	2022-03-10	08:39:18Z	SPY
1	Bearish	442859939	$SPY Why is it crashing so fast? \nAnyone know?	2022-03-10	08:39:09Z	SPY
2	Bearish	442859938	$SPY BULLS GOT TRAPPED?? AGAIN?!?!?!! HOW CAN ...	2022-03-10	08:39:08Z	SPY
3	Bearish	442859889	$SPY 405 by Friday… c'mon make me rich daddy!!	2022-03-10	08:38:22Z	SPY
4	Bearish	442859884	$SPY algo last night predicts 409	2022-03-10	08:38:19Z	SPY
...	...	...	...	...	...	...
707945	Bullish	235142330	$SOL.X Bullish RSI	2020-08-10	18:46:14Z	SOL.X
707946	Bullish	234896323	$SOL.X time to back up the truck and load up!	2020-08-09	22:37:22Z	SOL.X
707947	Bullish	231400354	$SOL.X this things going to pump harder than A...	2020-07-28	01:25:14Z	SOL.X
707948	Bullish	231399519	$SOL.X Daddy wants gains 🚀🚀🚀	2020-07-28	01:19:47Z	SOL.X
707949	Bullish	231397503	$SOL.X Solana will Jill ETH and do 1,000x	2020-07-28	01:06:35Z	SOL.X

707950 rows × 6 columns

Figure 27: Data 4 before processing

Data processing of Data 3 and 4 in data_reader.py

	Sentiment	User_id	Message	Date	Time	Symbol
0	Bearish	442859953	crypto is even worthless if there is no world	2022-03-10	08:39:18Z	SPY
1	Bearish	442859939	why is it crashing so fast anyone know	2022-03-10	08:39:09Z	SPY
2	Bearish	442859938	bulls got trapped again how can it be l m f a o	2022-03-10	08:39:08Z	SPY
3	Bearish	442859889	by friday cmon make me rich daddy	2022-03-10	08:38:22Z	SPY
4	Bearish	442859884	algo last night predicts	2022-03-10	08:38:19Z	SPY
...	...	...	...	...	...	...
680928	Bullish	235142330	bullish rsi	2020-08-10	18:46:14Z	SOL.X
680929	Bullish	234896323	time to back up the truck and load up	2020-08-09	22:37:22Z	SOL.X
680930	Bullish	231400354	this things going to pump harder than arnold s...	2020-07-28	01:25:14Z	SOL.X
680931	Bullish	231399519	daddy wants gains	2020-07-28	01:19:47Z	SOL.X
680932	Bullish	231397503	solana will jill eth and do x	2020-07-28	01:06:35Z	SOL.X

680933 rows × 6 columns

Figure 28: Data 4 after processing

Like Data 1 and 2, Data 3 is also pre-processed is done on-the-fly under *data_reader.py* since it is small. However, due to the large size of 700,000+ StockTwits, Data 4 was pre-processed beforehand and saved as *stocktwits_data_ALL_cleaned.csv*.

```python
def read_data3(mode, to_clean= True):

    cwd = os.getcwd()
    path = cwd + DATA_3_FILEPATH

    data3 = pd.read_csv(path)

    if(to_clean):
        data3 = preprocess_data(data3, "Message")

    if(mode=="dataframe"):
        data3_X = [item.lower() for item in data3['Message']]
        data3_y_class = [int(sentiment) for sentiment in data3['Annotator1']]

        data3 = pd.DataFrame()
        data3["text_cleaned"] = data3_X
        data3["Label"] = data3_y_class
        return data3

    elif(mode=="list"):
        data3_X = [item.lower() for item in data3['Message']]
        data3_X = to_tokens(data3_X)
        data3_y_class = [int(sentiment) for sentiment in data3['Annotator1']]

        return data3_X, data3_y_class
```

Figure 29: Function to read Data 3

```python
def read_data4(mode, to_clean= True, sample_size=10000, seed=100):

    cwd = os.getcwd()

    if(to_clean):
        path = cwd + DATA_4_FILEPATH[1]
    else:
        path = cwd + DATA_4_FILEPATH[0]

    data4 = pd.read_csv(path)

    if(mode=="lexicon"):
        return data4

    elif(mode=="bert"):
        data4['Label'] = [1 if sentiment=="Bullish" else 0 for sentiment in data4['Sentiment']]
        data4['text_cleaned'] = data4['Message']
        data4 = data4[["text_cleaned", "Label"]]

        data4 = data4.sample(n=sample_size)
        train_df = data4.sample(frac=0.8,random_state=seed)
        val_df = data4.drop(train_df.index)
        return train_df, val_df
```

Figure 30: Function to read Data 4

Data 3 is returned to the caller as a DataFrame or X and y lists based on the input parameters. Data 4 is returned as a DataFrame to the caller. Additionally, if it is to be used for sub symbolic training, it is further sampled and split into train and test DataFrames.

Data Pre-processing

All the data mentioned above were pre-processed and cleaned with the following steps. Detailed implementation can be found under *preprocess_data()* in *data_reader.py*.

- Removing any cashtag and tickers. E.g. $SPY
- Converting to lowercase
- Converting HTML encoding to UTF-8 encoding
- Removing hashtags, mentions, page breaks and handles
- Removing any hyperlinks
- Removing any numerical digits
- Removing any emojis
- Removing any punctuations
- Removing additional whitespaces
- Removing the empty rows post-processing

4. Lexicon Construction

Need to construct new Lexicons

As discussed in the Literature Review, lexicon-based sentiment analysis plays a key role in opinion mining. If the lexicon is missing words that are important indicators of sentiment, or if it incorrectly assigns sentiment strengths to words, the accuracy of the resulting sentiment analysis will be negatively impacted. However, most commonly used lexicons are not created:

1. For the specific use in the Cryptocurrency and Financial domain and
2. To process social media microblog content;
3. Or updated to reflect the current economic/societal influences

I will discuss 2 weakness that current lexicons lack in below.

Lack of temporal awareness

After careful analysis on mislabelled StockTwits via the use of traditional lexicons, I noticed that the meaning and sentiment of words change over time.

For example, “Inflation” has a much stronger negative connotation today compared to a year ago due to changes in our macro environment especially in light on the consecutive higher CPI reports.

The word “Transitory” also has a negative connotation today compared to a year ago. Ever since FOMC Chairman Jerome Powell effectively “retired” the term to describe the inflation dynamics affecting our economy, the term is no longer considered a “dovish” term as interpreted by users but more likely used as an mockery to the past inaction of the FED which contributed to the high inflationary environment faced today.

Lack of ability to capture to social media lingo

A closer inspection of the StockTwits also revealed how many social media lingo terms were not present in existing lexicons. These terms tend not to be formal words or if so, present a different meaning in the financial cryptocurrency space. Since a word’s strength and

sentiment is dependent upon the context in which it is used, it is unlikely that a word will have a single score across multiple domains, fuelling the need to build a specific lexicon for our task. Some of these terms are shown below.

Domain	Term	Meaning	Sentiment
General/Financial	Stonks	A term to express a financial decision that resulted in financial gain.	Positive
	LFG	Common abbreviation for "Lets fucking go"	Positive
	HODL	A misspelling of "hold" and later said to imply "Hold On for Dear Life" which reinforces the financial concept that you don't sell in a Bear Market. It implies that investors should either ride it until the bitter end, or till the price comes back up.	Positive
	Rip	A dramatic upward move in the price of an asset, relative to surrounding price moves	Positive
	Tank	A dramatic drop in the price of an asset, relative to surrounding price moves	Negative
Cryptocurrency	Staking	The process of locking up crypto holdings in order to obtain rewards or earn interest	Positive
	Soldana	A mashup of the word “sold” and the cryptocurrency “Solana”	Negative

Figure 31: Uncaptured social media lingo

Application to other areas where data is lacking

Another benefit of building such lexicons is the ability to apply these lexicons in areas where there may not be enough information to do corpus-based approaches. For example, these lexicons can be applied to web media contents where each text content may be longer than microblogs, such as traditional blog posts or forum threads.

Pre-Analysis of SenticNet Polarity Classfication API

To understand how current lexicons are lacking in our sentiment analysis task, especially in the financial cryptocurrency domain, we will take a look at the shortcomings of our baseline method. As such, we scraped tweets using the Twitter API and *tweepy* library. From this, we processed the tweets using both the SenticNet Polarity Classification and Concept Parsing API. This data is saved as *baseline_analysis_twitter.csv* under the *results* folder. This pre-analysis is done in *baseline_senticnet_analysis.ipynb*.

	tweettextcleaned	sentic_polarity	sentic_concept
0	The Fed is going to fight inflation - and lose.	POSITIVE	['fight', 'inflation', 'lose']
1	Since August 20, 2021, - Lumber is up over 157% - Bitcoin is down over 13% - US national debt increased to over $30 trillion - Inflation jumped to over 7.5%, a 40 year high - The Fed remains clueless And stocks might soon crash. Few understand this. Click Follow for more	POSITIVE	['lumber', 'debt', 'increased', 'inflation', 'jumped', 'high', 'remain', NOT 'clue', 'stocks', 'crash', 'understand', 'click', 'follow']
2	For the first time during this crisis there are rumors the FED is internally considering forgoing the rate hike completely in March. If this happens markets will rip.	NEGATIVE	['time', 'crisis', 'rumors', 'internally', 'consider', 'forgo', 'rate', 'hike', 'complete', 'hap', 'rip']
3	Following FOMC today, market will likely realise 1) Fed will begin to taper - and soon! 2) Tapering is not tightening (bubble can inflate further - weeks/months. And down the road. 3) This is first step on tapering/hiking cycle which will eventually pop the Everything Bubble	POSITIVE	['follow', 'real', 'beg', 'taper', NOT 'tighten', 'bubble', 'inflate', 'hiking', 'cycle', 'eventually', 'pop']
4	"We should have moved earlier," says the man (Jerome Powell), who still hasn't moved and won't move as far as he could and should. Great.	POSITIVE	['man', NOT 'moved', 'early', NOT 'move', 'great']
5	FED Jerome Powell "inflation is transitory." Quote of the decade	POSITIVE	['inflation', 'transitory', 'quote', 'decade']
6	Russia's war on Ukraine is likely to raise inflation and lower investment, energy prices to affect US economy in form of higher inflation in short term, US Federal Reserve Chair Jerome Powell said	POSITIVE	['in form', 'war', 'raise', 'inflation', 'lower', 'investment', 'energy', 'affect', 'economy', 'short', 'federal', 'reserve']
7	Russia's war on Ukraine will worsen inflation, says Fed's Jerome Powell Consumer prices are already rising at their fastest pace in four decades, having jumped 7.5 per cent in January compared with 12 months earlier Top Stories by BusinessStandard	POSITIVE	['war', 'worsen', 'inflation', 'consumer', 'rising', 'fastest', 'pace', 'decade', 'jumped', 'cent', 'compare', 'early', 'top']

Figure 32: Pre-analysis of a selection of SenticNet's mislabelled tweets

I have picked out a few of the incorrectly labelled rows after a quick scan of the data. For example, the tweet *'The FED is going to fight inflation – and lose'* in row 0 and its concepts *['fight', 'inflation', 'lose']* is clearly indicative of a negative financial sentiment. However, it was labelled as POSITIVE by the Polarity Classification API.

Another example is in row 2 where although the sentence is positive and the concept of *'rip'*, which we explained earlier, was captured, the tweet was still labelled as NEGATIVE by the API.

In rows 6 and 7, we can notice many negative concepts being captured but the tweets were still labelled as POSITIVE by the API. We will perform a deeper analysis of the SenticNet API after creating our Lexicon Ensemble in Chapter 5.

StockTwitLexi (STL)

Inspiration

In the paper *Estimating Sentiment via Probability and Information Theory* [32], the authors proposed the calculation of domain-specific sentiment scores using(1) a probabilistic approach and (2) an information theoretic approach. It is a unique approach since:

1. The lexicon can generated using text mining with no *apriori* knowledge - without any seed words or lexicon adaptation, and that
2. It contained words from all parts-of-speech, rather than just adjectives.

Their results show that the generated lexicons perform well in the sentiment analysis task with accuracy ranging from 87.30% to 88.75% versus a baseline of 81.60% for a widely-used lexicon. The lexicons also achieve good recall, precision, and F1-Scores. The best result was observed in the ensemble method.

	Recall	Precision	F-Score	Acc.
P3	0.89	**0.96**	0.92	87.60%
I3	0.90	0.95	0.92	87.30%
Ensemble	**0.91**	**0.96**	**0.93**	**88.75%**
Baseline	0.86	0.92	0.89	81.60%

Figure 33: Constructed lexicon results

I will use the proposed methods in this paper to generate a financial cryptocurrency and social media microblog specific lexicon known as StockTwitLexi (STL).

In their subsequent paper *Creating Domain-Specific Sentiment Lexicons via Text Mining* [33], the authors mentioned that several methods were experimented with to merge the 2 scores as an ensemble. I will explore these methods with experiments specific to our domain and therefore choose the most optimum method for our task.

Codebase

The code to build StockTwitLexi (STL) can be found under *buildlexicon_stocktwitlexi.ipynb*.

Probabilistic Score

Baye's Theorem

The probabilistic score $Score_{prob}(w)$ of a word w is calculated using the posterior probabilities of the mined text. This is based on Baye's theorem that calculates the posterior probability, the event A happening given that event B has happened.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For our purposes, event A the event of the text to be positive/negative event while event B is the event of observing the word w. The table below describes how the Baye's Theorem was applied for the equations used in the paper.

Bayes Theorem	$\boldsymbol{P(A\|B)}$ =	$\frac{\boldsymbol{P(A)} \times \boldsymbol{P(B\|A)}}{\boldsymbol{P(B)}}$
Probability of observing a positive-sentiment text, given that word w is observed	$P(pos\|w) =$	$\frac{P(pos) \times P(w\|pos)}{P(w)}$
Probability of observing a negative-sentiment text, given that word w is observed	$P(neg\|w) =$	$\frac{P(neg) \times P(w\|neg)}{P(w)}$

Figure 34: Baye's Theorem mapping

Formulas

$$Score_{prob}(w) = p(pos|w) - p(neg|w)$$

Where:

$$p(pos|w) = \frac{p(pos) \times p(w|pos)}{p(w)}$$

$$p(neg|w) = \frac{p(neg) \times p(w|neg)}{p(w)}$$

$$p(w|pos) = \frac{\frac{\sum_{r \in R_{pos}} n_{wr}}{\sum_{w'} \sum_{r \in R_{pos}} n_{w'r}} + 1}{k_{dic} + 1}$$

$$p(w|neg) = \frac{\frac{\sum_{r \in R_{neg}} n_{wr}}{\sum_{w'} \sum_{r \in R_{neg}} n_{w'r}} + 1}{k_{dic} + 1}$$

Variables mapping

Now, I will elaborate more on how the variables in the paper are mapped to the code and how the formulas are implemented.

	Variables in the paper	**Variables used in code**
1	$P(w)$	word_occurence
2	$P(pos)$	p_pos
3	$P(neg)$	p_neg
4	$P(pos\|w)$	p_pos_g_word
5	$P(neg\|w)$	p_neg_g_word
6	$P(w\|pos)$	p_word_g_pos
7	$P(w\|neg)$	p_word_g_neg
8	$\sum_{r \in R_{pos}} n_{wr}$	n_pos
9	$\sum_{r \in R_{neg}} n_{wr}$	n_neg
10	$\sum_{w\prime} \sum_{r \in R_{pos}} n_{w\prime r}$	n_every_pos
11	$\sum_{w\prime} \sum_{r \in R_{neg}} n_{w\prime r}$	n_every_neg
12	k_{dic}	k_dict

13	$Score_{prob}(w)$	probabilistic_score

Figure 35: Probabilistic score variables mapping

Implementation

1	**Data input**
	`stocktwit_df = data_reader.read_data4("lexicon", True)` To build a good lexicon, we will use Data 4 which is a large corpus of 680,933 rows that we have scraped previously on StockTwits. Rationale of using this data is explained under *Reason of choosing StockTwits* in Datasets.
2	**Generate base elements**
	We generated *word_occurence* using collections.Counter() and *k_dict* is calculated by finding the size of *word_occurence* dictionary. $$p(w) = \sum_{r \in R} n_{wr}$$

	Sentiment	User_id	Message	Date	Time	Symbol
0	Bearish	442859953	crypto is even worthless if there is no world	2022-03-10	08:39:18Z	SPY
1	Bearish	442859939	why is it crashing so fast anyone know	2022-03-10	08:39:09Z	SPY
2	Bearish	442859938	bulls got trapped again how can it be l m f a o	2022-03-10	08:39:08Z	SPY
3	Bearish	442859889	by friday cmon make me rich daddy	2022-03-10	08:38:22Z	SPY
4	Bearish	442859884	algo last night predicts	2022-03-10	08:38:19Z	SPY
...	...	...	...	...	...	...
680928	Bullish	235142330	bullish rsi	2020-08-10	18:46:14Z	SOL.X
680929	Bullish	234896323	time to back up the truck and load up	2020-08-09	22:37:22Z	SOL.X
680930	Bullish	231400354	this things going to pump harder than arnold s...	2020-07-28	01:25:14Z	SOL.X
680931	Bullish	231399519	daddy wants gains	2020-07-28	01:19:47Z	SOL.X
680932	Bullish	231397503	solana will jill eth and do x	2020-07-28	01:06:35Z	SOL.X

680933 rows × 6 columns

To refine our dictionary, we removed tokens that occur < 16 times in the whole corpus or if the token is < 3 characters in length. No other large filters or preprocessing is done to ensure that we manage to capture the unique terms present in the corpus.

```
# P(w) - total no. of occurence of w
corpus = stocktwit_df['Message'].str.cat(sep=' ')
corpus_list = corpus.split()
counter = collections.Counter(corpus_list)
word_occurence = dict(counter)
word_occurence
```

```
{'crypto': 26145,
 'is': 172039,
 'even': 12678,
 'worthless': 684,
 'if': 41969,
 'there': 15661,
 'no': 25487,
 'world': 8206,
 'why': 12106,
 'it': 94175,
 'crashing': 1117,
```

```
k_dict = len(word_occurence)
k_dict
```

```
13239
```

Next, we find *p_pos*, *p_neg*, *n_pos* and *n_neg* by first finding the rows where the token is present. Then, we split these rows into 2 DataFrames based on their sentiment – Bullish or Bearish. With this, we can formulate the *n_pos* and *n_neg* dictionaries by counting the number of times the token appears for each sentiment.

$$p(pos) = \sum_{w'} \sum_{r \in R_{pos}} n_{w'r}$$

$$p(neg) = \sum_{w'} \sum_{r \in R_{neg}} n_{w'r}$$

We then calculate corresponding proportion of words that belongs to the positive/negative class with regards to the number of times the token appears to get *p_pos*, *p_neg*. We iterate this process for all the tokens present in *word_occurence*.

```
# P(Pos) and P(Neg); N(Pos) and N(Neg)
p_pos = {}
p_neg = {}
n_pos = {}
n_neg = {}

for unique_word in tqdm(word_occurence.keys()):

    rowidx_containword = stocktwit_df["Message"].apply(lambda row: unique_word in row.split())
    df_containword = stocktwit_df[rowidx_containword]

    df_containword_pos = df_containword[df_containword["Sentiment"] == "Bullish"]
    count_in_posclass = df_containword_pos["Message"].apply(lambda row: row.split().count(unique_word)).sum()

    df_containword_neg = df_containword[df_containword["Sentiment"] == "Bearish"]
    count_in_negclass = df_containword_neg["Message"].apply(lambda row: row.split().count(unique_word)).sum()

    assert word_occurence[unique_word] == (count_in_posclass+count_in_negclass)

    n_pos[unique_word] = count_in_posclass
    n_neg[unique_word] = count_in_negclass
    p_pos[unique_word] = count_in_posclass / word_occurence[unique_word]
    p_neg[unique_word] = count_in_negclass / word_occurence[unique_word]

    assert (p_pos[unique_word] + p_neg[unique_word]) == 1
```

Thirdly, *n_every_pos* and *n_every_neg* are calculated by finding the total word count in all positive/negative sentiment rows. We check that the sum equals the total number of words in the corpus as a sanity check.

```
# N(Every Pos) and N(Every Neg)
stocktwit_df_pos = stocktwit_df[stocktwit_df["Sentiment"] == "Bullish"]
stocktwit_df_neg = stocktwit_df[stocktwit_df["Sentiment"] == "Bearish"]

n_every_pos = len(stocktwit_df_pos["Message"].str.cat(sep=' ').split())
print(n_every_pos)
n_every_neg = len(stocktwit_df_neg["Message"].str.cat(sep=' ').split())
print(n_every_neg)
n_corpus = len(stocktwit_df["Message"].str.cat(sep=' ').split())
print(n_corpus)
assert n_corpus == (n_every_pos+n_every_neg)
```

```
6784438
2799717
9584155
```

3	**Generate composite elements and probabilistic score**

p_word_g_pos and *p_word_g_neg* are calculated using the formulas:

$$p(w|pos) = \frac{\frac{\sum_{r \in R_{pos}} n_{wr}}{\sum_{w'} \sum_{r \in R_{pos}} n_{w'r}} + 1}{k_{dic} + 1} \qquad p(w|neg) = \frac{\frac{\sum_{r \in R_{neg}} n_{wr}}{\sum_{w'} \sum_{r \in R_{neg}} n_{w'r}} + 1}{k_{dic} + 1}$$

With these, *p_pos_g_word* and *p_neg_g_word* are calculated based on:

$$p(pos|w) = \frac{p(pos) \times p(w|pos)}{p(w)}$$

$$p(neg|w) = \frac{p(neg) \times p(w|neg)}{p(w)}$$

Lastly, we can find the probabilistic score:

$$Score_{prob}(w) = p(pos|w) - p(neg|w)$$

```
# Score_prob(W)
probabilistic_score = {}

for word, freq in word_occurence.items():
    p_word_g_pos = ((n_pos[word] / n_every_pos) + 1) / (k_dict + 1)
    p_word_g_neg = ((n_neg[word] / n_every_neg) + 1) / (k_dict + 1)
    p_pos_g_word = (p_pos[word] *  p_word_g_pos) / freq
    p_neg_g_word = (p_neg[word] *  p_word_g_neg) / freq
    probabilistic_score[word] = p_pos_g_word - p_neg_g_word
```

```
probabilistic_score
```

```
{'crypto': 2.148120032241357e-09,
 'even': 1.8219054513635102e-09,
 'worthless': 7.10042277614196e-09,
 'there': 1.8921375575695464e-09,
 'world': 3.6617392994043477e-09,
 'why': 2.7071306210153533e-09,
 'crashing': -4.4242694497590335e-09,
 'fast': 9.166396458185093e-09,
 'anyone': 6.458281600102852e-09,
 'know': 2.5364396587651704e-09,
 'bulls': -1.7763292364113776e-09,
 'got': 3.347708048342275e-09,
 'trapped': -3.536886448754294e-10,
```

The paper uses Amazon product reviews which are rated with scores from 1 star to 5 stars as its mined text. $\sum_{r \in R_{neg}} n_{wr}$ and $\sum_{r \in R_{pos}} n_{wr}$ which represents the number of occurrences of every word in the negative class and positive class respectively is therefore calculated by scaling the extreme ends (1 star and 5 star reviews) to the factor of γ.

$$\sum_{r \in R_{pos}} n_{wr} = \gamma\, n_{w5^*} + n_{w4^*}$$

$$\sum_{r \in R_{neg}} n_{wr} = \gamma\, n_{w1^*} + n_{w2^*}$$

In our case, we will omit this scaling since our sentiment scores from our mined text are only labelled as 0 or 1.

Information Theoretic Score

TF-IDF

The information theoretic score $Score_{it}(w)$ of a word w is based on the well known information theory based measure called Term Frequency – Inverse Document Frequency (TF-IDF) that is mainly used in information retrieval. This measure reflects the importance of a word to a document in a collection of corpus.

Term frequency refers to the number of times a term occurs in a document.

Document frequency, on the other hand, refers to the number of documents that contain the word. In general, words tend to appear less frequently in general should be given more weight since they are more meaningful while those that appear more frequently like stop words should be given lesser weight since they are less meaningful. Using the Inverse Document Frequency (IDF) will help us achieve this.

$$tf\ idf\ (t, d, D) = tf\ (t, d)\ .\ idf\ (t, D)$$

where:

$$tf\ (t, d) = log\ (1 + freq\ (t, d))$$

$$idf\ (t, D) = log\ \left(\frac{N}{count\ (d \in D : t \in d)} \right)$$

The difference for calculating the information theoretic score is the replacement of the Term Frequency term with the difference between $pos(w)$ and $neg(w)$. The 2 scores are calculated based on the term frequency of the token in each class.

Formulas

$$Score_{it}(w) = \big(pos(w) - neg(w)\big) \times IDF(w)$$

where:

$$pos(w) = \frac{rtf_{w\_pos}}{N_{pos}} \times N$$

$$neg(w) = \frac{rtf_{w\_neg}}{N_{neg}} \times N$$

$$IDF(w) = log\frac{N}{df_w}$$

Variables Mapping

Now, I will elaborate more on how the variables in the paper are mapped to the code and how the formulas are implemented.

	Variables in the paper	**Variables used in code**
1	rtf_{w_pos}	rtf_pos_dict
2	rtf_{w_neg}	rtf_neg_dict
3	$pos(w)$	pos_w
4	$neg(w)$	neg_w
5	df_w	dfw_dict
6	$IDF(w)$	idf_w
7	$Score_{it}(w)$	info_score

Figure 36: Information Theoretic score variable mapping

Implementation

1	**Data input**
	We will use the same Data 4 for this purpose too.
2	**Generate TF terms**

	The relative term frequency of word w in row r (rtf_w) is calculated by summing up the number of times each token appears in row r, divided by the number of terms present in the row, for every row.
	
	To generate *rtf_pos_dict* and *rtf_neg_dict*, we will use the previously split DataFrames based on their sentiment values (*stocktwit_df_pos, stocktwit_df_neg*) and perform the above mentioned operation.
	
	`# Relative Term Frequency of word w in review r`
	`rtf_pos_dict = {key: 0 for key in word_occurence.keys()}`
	`rtf_neg_dict = {key: 0 for key in word_occurence.keys()}`
	
	`def calculate_rtf(msg, rtf_dict):`
	`  for word in rtf_dict.keys():`
	`  word_list = msg.split(" ")`
	`  if word_list.count(word) != 0:`
	`  rtf_dict[word] += word_list.count(word)/len(word_list)`
	
	`stocktwit_df_pos["Message"].progress_apply(lambda row: calculate_rtf(row, rtf_pos_dict))`
	`stocktwit_df_neg["Message"].progress_apply(lambda row: calculate_rtf(row, rtf_neg_dict))`
	
	With the rtf_w values, we can easily calculate *pos(w) and neg(w)* using the formulas :
	
	$$pos(w) = \frac{rtf_{w\_pos}}{N_{pos}} \times N$$
	
	$$neg(w) = \frac{rtf_{w\_neg}}{N_{neg}} \times N$$
	
	N is the length of the main DataFrame while N_{pos} and N_{neg} are the lengths of split DataFrames.

```
# Pos(w) and Neg(w)
pos_w = {key: 0 for key in word_occurence.keys()}
neg_w = {key: 0 for key in word_occurence.keys()}

for word in word_occurence.keys():
    pos_w[word] = rtf_pos_dict[word] / len(stocktwit_df_pos) * len(stocktwit_df)
    neg_w[word] = rtf_neg_dict[word] / len(stocktwit_df_neg) * len(stocktwit_df)
```

3 **Generate DF term**

Here, we will generate the df_w term by simply counting the number of rows where a token appear in, repeating this for all tokens.

```
# Document Frequency of word w
dfw_dict = {key: 0 for key in word_occurence.keys()}

def calculate_dfw(msg):
    for word in word_occurence.keys():
        word_list = msg.split(" ")
        if word_list.count(word) != 0:
            dfw_dict[word] += 1

stocktwit_df["Message"].progress_apply(lambda row: calculate_dfw(row))
dfw_dict
```

```
100%|
```

```
{'crypto': 22797,
 'even': 12128,
 'worthless': 669,
 'there': 14587,
 'world': 7616,
 'why': 11572,
 'crashing': 1095,
 'fast': 2749,
 'anyone': 6302,
 'know': 14775,
 'bulls': 16564,
 'got': 10843,
 'trapped': 795,
```

4 **Generate IDF element and information theoretic score**

With all the base elements, we then calculate $IDF(w)$ and the information theoretic score $Score_{it}(w)$ based on these formulas:

$$IDF(w) = log\frac{N}{df_w}$$

$$Score_{it}(w) = \big(pos(w) - neg(w)\big) \times IDF(w)$$

```
# Inverse Document Frequency of word w
idf_w = {key: 0 for key in word_occurence.keys()}
info_score = {}

for word in word_occurence.keys():
    idf_w[word] = math.log(len(stocktwit_df) / dfw_dict[word])
    info_score[word] = (pos_w[word] - neg_w[word]) * idf_w[word]

info_score
```

Ensemble Combination Method

We will explore the various methods to combine the probabilistic score $Score_{prob}(w)$ and information theoretic score $Score_{it}(w)$ together and use empirical evidence to select the best method for our use. This comparison can be found under *buildlexicon_stocktwitlexi_comparison.ipynb*.

We will test 5 different methods using 3 datasets – Data 1, 2 and 3.

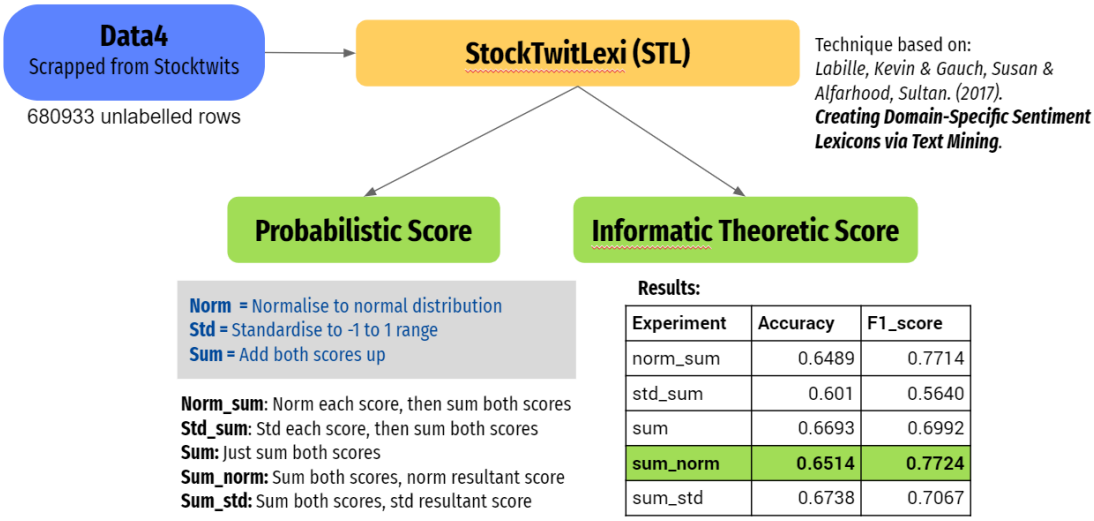

Figure 37: StockTwitLexi Ensemble Combination Method diagram

Combinations Methods

Name	Explanation
Norm_sum	Norm each score, then average both scores
Std_sum	Std each score, then average both scores
Sum	Just average both scores
Sum_norm	Average both scores, norm resultant score
Sum_std	Average both scores, std resultant score

Figure 38: Score combination methods

where:

Norm = Normalise to normal distribution
Std = Standardise to -1 to 1 range
Sum = Add both scores up

Test Results

Data 1				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
domain_lexicon_norm.csv	0.691547	0.804143	0.6832	0.977117
domain_lexicon_std.csv	0.575877	0.569277	0.832599	0.432494
domain_lexicon_sum.csv	0.641127	0.688412	0.787046	0.611747
domain_lexicon_sum_norm.csv	**0.695502**	**0.805923**	**0.686527**	**0.975591**
domain_lexicon_ sum_std.csv	0.644093	0.694656	0.782235	0.624714

Figure 39: STL Test Results for Data 1

Data 2				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
domain_lexicon_norm.csv	0.613714	0.752199	0.612903	0.973435
domain_lexicon_std.csv	0.533714	0.434903	0.805128	0.297913
domain_lexicon_sum.csv	0.626286	0.644178	0.755102	0.56167
domain_lexicon_sum_norm.csv	**0.620571**	**0.755882**	**0.617047**	**0.975332**
domain_lexicon_sum_std.csv	0.633143	0.655209	0.75495	0.578748

Figure 40: STL Test Results for Data 2

Data 3				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
domain_lexicon_norm.csv	0.641688	0.75814	0.62572	0.961652
domain_lexicon_std.csv	0.69509	0.687831	0.855263	0.575221
domain_lexicon_sum.csv	0.740741	0.765027	0.812604	0.722714
domain_lexicon_sum_norm.csv	0.638243	0.75553	0.624038	0.957227
domain_lexicon_sum_std.csv	**0.744186**	**0.770302**	**0.809756**	**0.734513**

Figure 41: STL Test Results for Data 3

Average of 3 Test Datasets				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
domain_lexicon_norm.csv	0.648983	0.771494	0.640608	0.970735
domain_lexicon_std.csv	0.60156	0.564004	0.830997	0.435209
domain_lexicon_sum.csv	0.669385	0.699206	0.784917	0.632044
domain_lexicon_sum_norm.csv	**0.651439**	**0.772445**	**0.642537**	**0.969383**
domain_lexicon_sum_std.csv	0.673807	0.706722	0.782314	0.645992

Figure 42: STL Test Results for Average of 3 Datasets

Discussion of results

We can see that performing the *sum_norm* combination has the best average results in terms of its F1 score and it is ranked 1st for 2 out of the 3 datasets. This differs from the paper's results where the authors concluded that *sum* performs best. This can be due to the difference in the Data used to create the lexicon, with ours being labelled as 0 or 1 while the paper's product reviews dataset was labelled from 1-5 stars.

We will follow our empirical evidence and use *sum_norm* for StockTwitLexi (STL). The formulas are as follows:

$$x_w = \frac{Score_{prob}(w) + Score_{it}(w)}{2}$$

$$sum\_norm_w = \frac{x_w - \mu}{\sigma}$$

Senti-DD

Inspiration

In the paper *Automatic Construction of Context-Aware Sentiment Lexicon in the Financial Domain Using Direction-Dependent Words* [34], the authors propose a method to incorporate context when building a sentiment lexicon from a given corpus.

The constructed lexicon, Senti-DD, will comprise of a pairing of a directional word and a direction-dependent word. This is to capture the true sentiment of a given word as its semantic orientation changes according to the context in which it is being used. For example, 'profit' conveys a positive sentiment while 'profit decrease' conveys a negative sentiment. This cause unigram lexicons to not perform well when the text to be processed is not straightforward.

The paper also highlighted the need for domain-specificity when solving sentiment analysis task which reaffirms our need to construct new lexicons discussed earlier.

Dataset	Measure	SWN [26]	TextBlob [27]	VADER [29]	LM [9]	LM+Senti-DD
DS50	Precision	0.4778	0.5155	0.6028	0.6147	**0.7090**
	Recall	0.3969	0.4852	0.5396	0.6232	**0.7055**
	F1-score	0.4107	0.4953	0.5452	0.5914	**0.7001**
DS66	Precision	0.4851	0.5275	0.6194	0.6337	**0.7389**
	Recall	0.4044	0.4968	0.5534	0.6363	**0.7315**
	F1-score	0.4194	0.5070	0.5599	0.6023	**0.7271**
DS75	Precision	0.4916	0.5426	0.6409	0.6507	**0.7796**
	Recall	0.4009	0.5039	0.5590	0.6556	**0.7702**
	F1-score	0.4199	0.5169	0.5702	0.6174	**0.7673**
DS100	Precision	0.4624	0.5476	0.6405	0.6377	**0.8238**
	Recall	0.3873	0.5228	0.5688	0.6476	**0.8128**
	F1-score	0.4058	0.5317	0.5770	0.5982	**0.8105**

Figure 43: Senti-DD Performance

Their results were shown to perform better in classification performance with Senti-DD than without it. Furthermore, this process can be automated through their proposed framework. As such, I will use the proposed methods in this paper to automatically construct a context-aware lexicon named Senti-DD using Data 1 and 4, before testing the results with Data 2 and 3.

Codebase

The code to build Senti-DD can be found under *buildlexicon_sentidd.ipynb*.

Pre-preparation of supplementary sources

First, to build Senti-DD we have to prepare the following 2 dictionaries. The code can be found under *sentidd_supplementarydata.ipynb*.

Directional words

We obtain the list of directional words chosen in the paper, before creating the stemmed column using PorterStemmer(). The file is saved as *directional_words.csv* under *custom_lexicons/sentidd*.

Table 3: List of directional words

Directionality type	Words
Up	accelerate, advance, award, better, climb, double, faster, gain, grow, higher, increase, jump, quicken, rebound, recover, rise, rose, step-up, surge, up
Down	constrain, decelerate, decline, decrease, down, drop, fall, fell, slower, weaken, weaker

Figure 44: List of directional words for Senti-DD

	token	label	stemmed				
0	accelerate	up	acceler	15	rise	up	rise
1	advance	up	advanc	16	rose	up	rose
2	award	up	award	17	step-up	up	step-up
3	better	up	better	18	surge	up	surg
4	climb	up	climb	19	up	up	up
5	double	up	doubl	20	constrain	down	constrain
6	faster	up	faster	21	decelerate	down	deceler
7	gain	up	gain	22	decline	down	declin
8	grow	up	grow	23	decrease	down	decreas
9	higher	up	higher	24	down	down	down
10	increase	up	increas	25	drop	down	drop
11	jump	up	jump	26	fall	down	fall
12	quicken	up	quicken	27	fell	down	fell
13	rebound	up	rebound	28	slower	down	slower
14	recover	up	recov	29	weaken	down	weaken
				30	weaker	down	weaker

Figure 45: Processed directional words

Loughran-McDonald Dictionary

We obtain the updated dictionary from University of Notre Dame – Software Repository for Accounting and Finance. Then, we reformatted the dictionary and only included tokens that are either negative or positive. The file is saved as *LM_Word_List.csv* under *custom_lexicons/sentidd*.

	Word	Seq_num	Word Count	Word Proportion	Average Proportion	Std Dev	Doc Count	Negative	Positive	Uncertainty	Litigious	Strong_Modal	Weak_Modal
0	AARDVARK	1	354	1.550080e-08	1.422600e-08	3.815486e-06	99	0	0	0	0	0	0
1	AARDVARKS	2	3	1.313627e-10	8.653817e-12	9.241714e-09	1	0	0	0	0	0	0
2	ABACI	3	9	3.940882e-10	1.169679e-10	5.290465e-08	7	0	0	0	0	0	0
3	ABACK	4	29	1.269840e-09	6.654735e-10	1.595100e-07	28	0	0	0	0	0	0
4	ABACUS	5	8570	3.752595e-07	3.809464e-07	3.529356e-05	1108	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
86526	ZYGOTE	86529	50	2.189379e-09	8.729336e-10	1.886011e-07	35	0	0	0	0	0	0
86527	ZYGOTES	86530	1	4.378757e-11	1.809516e-11	1.932446e-08	1	0	0	0	0	0	0
86528	ZYGOTIC	86531	0	0.000000e+00	0.000000e+00	0.000000e+00	0	0	0	0	0	0	0
86529	ZYMURGIES	86532	0	0.000000e+00	0.000000e+00	0.000000e+00	0	0	0	0	0	0	0
86530	ZYMURGY	86533	0	0.000000e+00	0.000000e+00	0.000000e+00	0	0	0	0	0	0	0

86531 rows × 16 columns

	word	label
9	abandon	negative
10	abandoned	negative
11	abandoning	negative
12	abandonment	negative
13	abandonments	negative
...	...	...
86080	wrongdoing	negative
86081	wrongdoings	negative
86085	wrongful	negative
86086	wrongfully	negative
86094	wrongly	negative

2709 rows × 2 columns

Figure 46: Processing LM Word List

PMI Score

Before we dive into the implementation details, let us first discuss the Pointwise Mutual Information (PMI) score. PMI is a measure of association between a pair of outcomes x and y. Hence, the algorithm computes the log probability of co-occurrence scaled by the product of the single probability of occurrence as follows:

$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)}$$

In our case, the 2 outcomes which the paper exploits are:

1. Event *w* where word *w* is observed and,
2. Event *t* where the row is of direction-dependency type *t*

Framework and Formulas

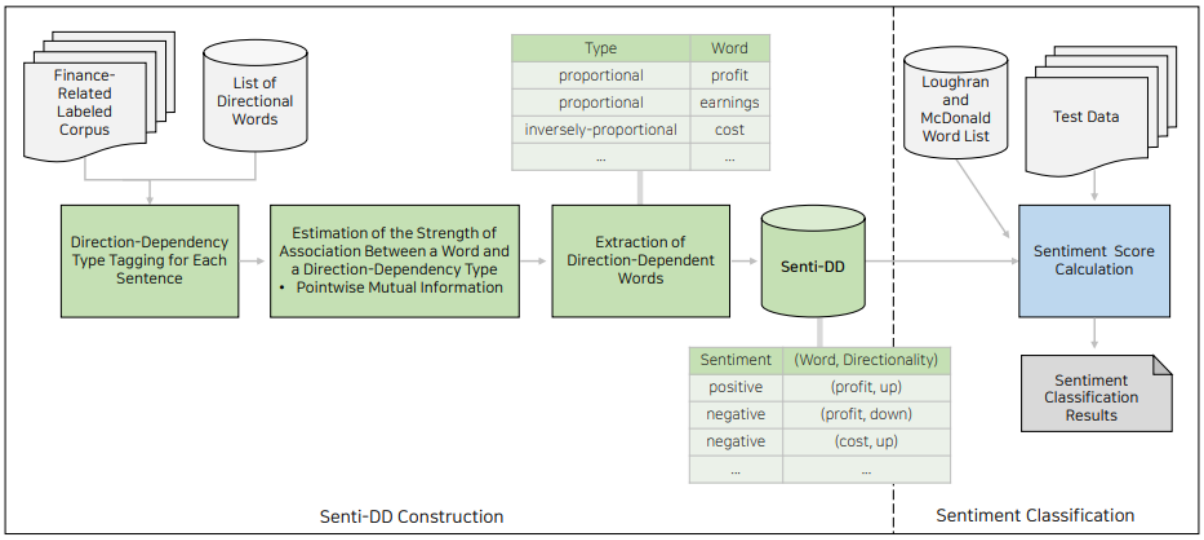

Figure 47: Senti-DD Procedure

Direction-Dependency Type Tagging

$$DirectionScore(s) = UpScore(s) - DownScore(s)$$

Strength of Association between word w and Directional-Dependency Type t

$$DirectionDependencyScore(w) = \begin{cases} |PMI(w,t_p)| & \text{if } PMI(w,t_p) > PMI(w,t_i) \\ 0 & \text{if } PMI(w,t_p) = PMI(w,t_i) \\ -|PMI(w,t_i)| & \text{if } PMI(w,t_p) < PMI(w,t_i) \end{cases}$$

$$\text{, where } PMI(w,t) = log_2 \frac{p(w,t)}{p(w)p(t)}$$

Implementation

From the framework, we can see that there are 2 inputs - (1) the directional list which we have created earlier and (2) a financial related corpus. We will vary (2) the financial related corpus between Data 1 and Data 4 and test the Senti-DDs built on them with Data 2 and 3 to eventually select the best lexicon for our task.

1	**Data input**

	I will use Data 1 for the following explanation.
	
	`data1 = data_reader.read_data1("dataframe", True)`
	`data1`
	
	text_cleaned | Label
	0 | downgrades stocks investing tradeideas | 0
	1 | looking sexy this morning break on volume and ... | 1
	2 | stock hasnt moved much since first few weeks a... | 1
	3 | whole foods may feel price competition but wil... | 1
	4 | apples iphone se could be doing better than ex... | 1
	... | ... | ...
	2018 | hold through all the media bs and you will be ... | 1
	2019 | i love blood like a vampirekeep bleeding for me | 0
	2020 | yea dude airplanes are still flying | 1
	2021 | ten year yield weekly year yield still very be... | 1
	2022 | the best scenario going forward is this stock ... | 0
	
	2023 rows × 2 columns

2	**Construct Senti-DD**
	construct_senti_dd() is the umbrella function that will perform the following steps:

- *Direction-dependency Type Tagging for Each Row*

  We count the number of "up" and "down" words present in each row to get the *UpScore(s)* and *DownScore(s)* respectively. The "up" and "down" labels are based on *directional_words.csv* that we created earlier. Then, we can calculate the *DirectionScore(s)* of the row based on the formula:

$$DirectionScore(s) = UpScore(s) - DownScore(s)$$

  Following that, we will assign the directional dependency tag based on the row's *DirectionScore* and its sentiment label.

DirectionScore	Sentiment Label	Directional Dependency Tag
Positive	Positive	Proportional
Positive	Negative	Inversely Proportional
Negative	Positive	Inversely Proportional
Negative	Negative	Proportional

```
def assign_direction_dependency_type(text, label):
    tokens = word_tokenize(text)
    up_cnt, down_cnt = 0, 0
    for token in tokens:
        if stemmer.stem(token) in directional_words_df[directional_words_df['label']=='up'].stemmed.values:
            up_cnt += 1
        if stemmer.stem(token) in directional_words_df[directional_words_df['label']=='down'].stemmed.values:
            down_cnt += 1
    score = up_cnt - down_cnt
    if (score > 0 and label == 'positive') or (score < 0 and label == 'negative'): return 'proportional'
    if (score > 0 and label == 'negative') or (score < 0 and label == 'positive'): return 'inversely_proportional'
```

- *Estimation of the Strength of Association Between a Word and a Direction-Dependency Type*

  Next, we reduced the corpus by removing words that appear less than 5 times and words with 1 character length. Then, we calculate the PMI score of event *w* where word *w* is observed and event *t* where the row is of direction-dependency type *t*, which we explained in the earlier section. We assign the *DirectionDependencyScore()* based on the formula:

$$DirectionDependencyScore(w) = \begin{cases} |PMI(w,t_p)| & \text{if } PMI(w,t_p) > PMI(w,t_i) \\ 0 & \text{if } PMI(w,t_p) = PMI(w,t_i) \\ -|PMI(w,t_i)| & \text{if } PMI(w,t_p) < PMI(w,t_i) \end{cases}$$

```
def pmi(df, word, t):
    n = len(df)
    a = count_sentences_containing(df[df.direction_dependency==t].nouns, word)
    b = count_sentences_containing(df[df.direction_dependency!=t].nouns, word)
    c = count_sentences_not_containing(df[df.direction_dependency==t].nouns, word)
    return (n*a)/((a+b)*(a+c))

def pmi_combined(df, word):
    pmi_prop = pmi(df, word, 'proportional')
    pmi_invprop = pmi(df, word, 'inversely_proportional')
    if pmi_prop > pmi_invprop: return abs(pmi_prop)
    elif pmi_prop < pmi_invprop: return -abs(pmi_invprop)
    else: return 0
```

- *Extraction of Direction-dependent Words*

  For each row, we extract a single representative word with these rules.

Directional Dependency Tag	**Representative Word**
Proportional	$argmax_{w \in pro\_cand}(DDScore_w)$
Inversely Proportional	$argmin_{w \in inv\_cand}(DDScore_w)$

Where:

$DDScore_w$ is Directional-Dependency score of word w

pro_cand is the list of candidate words with positive $DDScore_w$

inv_cand is the list of candidate words with negative $DDScore_w$

```
# Extraction of Direction-dependent Words
df['scores'] = df['nouns'].progress_apply(lambda x: np.array([lexicon_df[lexicon_df.token==token].iloc[0].pmi for token in x]))

df['token_score'] = df.progress_apply(lambda x: extract_token_score(x['nouns'], x['scores'], x['direction_dependency']), axis=1)
df['entity'] = df['token_score'].progress_apply(lambda x: x[0])
df['entity_score'] = df['token_score'].progress_apply(lambda x: x[1])
df.drop(columns=['token_score'], inplace=True)

dd = pd.DataFrame.from_records(
    list(zip(df.direction_dependency.values, df.entity.values, df.entity_score.values)),
                            columns=['direction_dependency', 'entity', 'score'])
dd.dropna(subset=['score'], inplace=True)
dd.drop_duplicates(subset=['direction_dependency', 'entity'], inplace=True)

proportional_words = dd[dd.direction_dependency=='proportional'].entity.values
inversely_proportional_words = dd[dd.direction_dependency=='inversely_proportional'].entity.values
print('\n\nProportional type entities ({})\n{}'.format(len(proportional_words), ', '.join(proportional_words)))
print('\nInversely proportional type entities ({})\n{}'.format(len(inversely_proportional_words),
                                                             ', '.join(inversely_proportional_words)))
```

- *Senti-DD Construction based on the List of Directional Words and the Direction-dependent Words*

  Lastly, we create pairs of words from the list of directional words from *directional_words.csv* and the list of direction-dependent words which we have created based on this table.

Directional Word	Directional-dependent Word	Label
Up	Proportional	Positive
Up	Inversely Proportional	Negative
Down	Proportional	Negative
Down	Inversely Proportional	Positive

```
# Senti-DD Construction based on the List of Directional Words and the Direction-dependent Words
up_words = directional_words_df[directional_words_df.label=='up'].stemmed.values
down_words = directional_words_df[directional_words_df.label=='down'].stemmed.values

records = []
records.extend([('positive', entity, direction) for entity in proportional_words for direction in up_words])
records.extend([('positive', entity, direction) for entity in inversely_proportional_words for direction in down_words])
records.extend([('negative', entity, direction) for entity in proportional_words for direction in down_words])
records.extend([('negative', entity, direction) for entity in inversely_proportional_words for direction in up_words])
senti_dd = pd.DataFrame.from_records(records, columns=['sentiment', 'entity', 'directional_word'])
```

3 **Usage of Senti-DD Lexicon**

To calculate the *Sentiment* score of a token w, we will first find the score based on the Loughran-McDonald Word List which we prepared earlier. We subtract the number of negative words in the row from the number of positive words in the row based on this dictionary.

$$SentimentScore(s) = PosScore(s) - NegScore(s)$$

$$PosScore(s) = |S_{pos}|$$

$$NegScore(s) = |S_{neg}|$$

$$S_{pos} = Set\ of\ positive\ words\ in\ the\ row$$

$$S_{neg} = Set\ of\ negative\ words\ in\ the\ row$$

Similarly, we calculate the *ContextSentimentScore* by counting the number of positive pairs based on the Senti-DD Lexicon, and subtracting by the number of negative pairs.

$$ContextSentimentScore(s) = ContextPosScore(s) - ContextNegScore(s)$$

$$ContextPosScore(s) = |S_{pos\_con}|$$

$$ContextNegScore(s) = |S_{neg\_con}|$$

$$S_{pos\_con} = Set\ of\ positive\ pairs\ in\ the\ row$$

$$S_{neg\_con} = Set\ of\ negative\ pairs\ in\ the\ row$$

	Finally, we can obtain our final *RefinedSentimentScore* by combining the 2 scores using the formula: $$RefinedSentimentScore(s) = \begin{cases} SentimentScore(s) + 1 & \text{if } ContextSentimentScore > 0 \\ SentimentScore(s) & \text{if } ContextSentimentScore = 0 \\ SentimentScore(s) - 1 & \text{if } ContextSentimentScore < 0 \end{cases}$$

Test Results

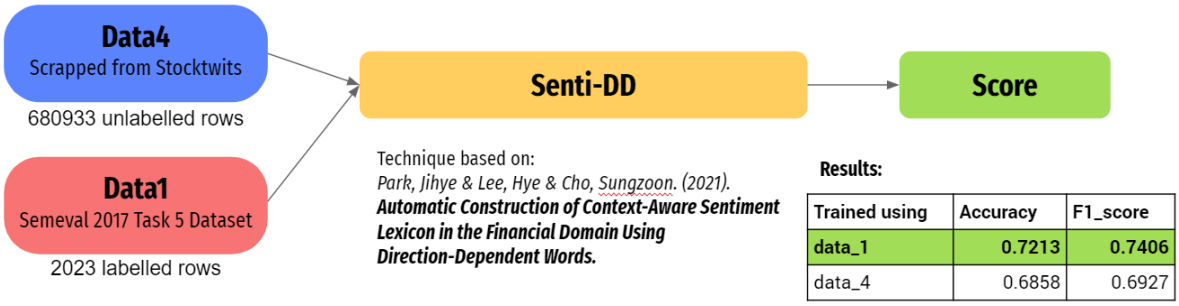

Figure 48: Senti-DD Overview

Created with Data 1				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
data2	0.851441	0.858947	0.906667	0.816
data3	0.59126	0.622328	0.696809	0.562232
Average	**0.721351**	**0.740638**	**0.801738**	**0.689116**

Figure 49: Test Results for Senti-DD created with Data 1

Created with Data 4				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
data2	0.769767	0.765957	0.89011	0.672199
data3	0.601852	0.619469	0.686275	0.564516
Average	0.68581	0.692713	0.788193	0.618358

Figure 50: Test Results for Senti-Dd created with Data 4

Discussion of results

We can see that performing the classification results using Senti-DD built on Data 1 is better than the one built on Data 4. It has a higher F1_score for both test Data 2 and Data 3 and thus a higher average F1_score. The average accuracy is also higher due it is great performance for test Data 2.

We will thus follow our empirical evidence and use Senti-DD built on Data 1.

5. Symbolic Approach

Lexicon Ensemble

We propose the use of Lexicon Ensemble for our Symbolic Approach. This is similar to the voting ensembles used in machine learning that combines the predictions from multiple models to improve the overall performance. It can be considered as a meta-model, a model of models.

In our case, we will use this concept to combine lexicon scores from multiple lexicons to try to improve the F1_scores presented by individual lexicons.

Lexicons used

The lexicons to be experimented with are SenticNet, NTUSD-Fin, SentiWordNet, Afinn, Vader and the 2 newly constructed lexicons, StockTwitLexi and Senti-DD.

Hypothesis and Benefits

Our hypothesis is that an ensemble will help to reduce the spread or dispersion of the scores and thus increase performance. Intuitively, our observations of the scores generated by individual lexicons indicate that some lexicons are not able to capture the degree of sentiment intensity as well as others, albeit capturing strong-sentiment tokens better. This implies that a combination should be complementary.

	senticnet	ntusd	sentiwordnet	afinn	vader	stocktwitlexi	sentidd	actual_class
0	0.302600	-0.084072	0.015625	0.038095	0.5859	0.062336	0.5	1
1	-0.126786	0.131360	0.017045	-0.005263	0.2023	0.088886	-0.5	0
2	0.137750	0.026968	-0.013158	-0.024242	-0.7351	0.088475	0.0	1
3	0.857500	0.286495	0.020833	0.018182	0.3818	0.089604	0.0	1
4	0.854500	0.309721	0.015625	0.040000	0.6249	0.116167	0.5	1
...	...	...	...	...	...	...	...	...
870	0.802500	0.152355	-0.050000	0.020000	0.2732	0.088337	0.5	1
871	0.661000	0.088727	0.022727	0.030769	0.5719	0.082826	0.0	1
872	0.850500	0.170936	-0.057692	0.000000	0.0000	0.040192	-0.5	1
873	0.833000	-0.152019	0.041667	0.080000	0.2023	0.049296	0.0	1
874	0.271000	0.026862	-0.009615	0.008696	0.2960	-0.018969	-0.5	0

875 rows × 8 columns

Figure 51: Inability for some lexicons to capture degree of sentiment intensity

Normalisation and Standardisation

Standardisation

To ensure that each row if of the same scale, we will standardise the scores based on the number of words processed in each row. This is to prevent rows that contain more words present in the lexicon to have a larger magnitude of score compared to those with fewer words.

$$StandardisedScore(s) = \frac{SentimentScore(s)}{|P|}$$

$$where\ P\ is\ the\ set\ of\ processed\ words\ in\ the\ row$$

Normalisation

To ensure that each lexicon score if of the same scale, we will normalise each score based on its max and min score in the lexicon. We will limit the range of each score to be 2.

$$NormalisedScore(s, L) = \frac{SentimentScore(s)}{\frac{\max(score_{score \in L}) - \min(score_{score \in L})}{2}}$$

$$where\ L\ is\ the\ lexicon\ of\ choice$$

```
def standardise_scores(score, count):
    if (count>0):
        return score/count
    else:
        return score

def normalise_scores(pred_raw, lexicon):
    if(lexicon=="senticnet"): #-1 to 1
        return pred_raw
    elif(lexicon=="ntusd"): #-3.81 to 1.22, range is 5 so /2.5
        return [pred/2.5 for pred in pred_raw]
    elif(lexicon=="sentiwordnet"): #-1 to 1 since pos-neg
        return pred_raw
    elif(lexicon=="stocktwitlexi"): #-1 to 1
        return pred_raw
    elif(lexicon=="afinn"): #-5 to 5
        return [pred/5 for pred in pred_raw]
    elif(lexicon=="vader"): #-1 to 1
        return pred_raw
    elif(lexicon=="sentidd"): #-2 to 2
        return [pred/2 for pred in pred_raw]
```

Figure 52: Functions to standardise and normalise scores

Voting Methods

We will experiment with 3 types of voting methods based on the as shown in this table.

Method	**Explanation**
Soft Voting	Predict the class with the largest summed probability from models
Hard Voting	Predict the class with the largest sum of votes from models
Soft Voting & Leave 2 out	Exclude highest and lowest lexicon score before performing soft voting

Figure 53: Voting methods

```
def combine_voting_leave2soft(row):
    lowest_col = row.sort_values().idxmin()
    highest_col = row.sort_values().idxmax()
    lowest_dict[lowest_col] += 1
    highest_dict[highest_col] += 1
    return row.sort_values().iloc[1:4].mean()

def combine_voting_soft(row):
    return row.sort_values().mean()

def combine_voting_hard(row):
    pos_vote = (row > 0).sum()
    neg_vote = (row <= 0).sum()
    return 1 if pos_vote>neg_vote else 0
```

Figure 54: Functions implementing voting methods

Generating Combinations

There is no training of models, but instead, we will iterate through all combinations of lexicons to find the best ensemble. We use *itertools.combinations()* to generate all combinations of length ≤ number of lexicons. We aim to find the best combination of lexicons in classifying the sentiment of financial cryptocurrency text.

```
combination_list = []
lexicons = ["senticnet", "ntusd", "sentiwordnet", "stocktwitlexi", "afinn", "vader", "sentidd"]

for i in range(1, len(lexicons)+1):
    combination_tuples = list(itertools.combinations(lexicons, i))
    combination_list.extend([list(elem) for elem in combination_tuples])
```

Figure 55: Code to generate combinations

Overall Framework

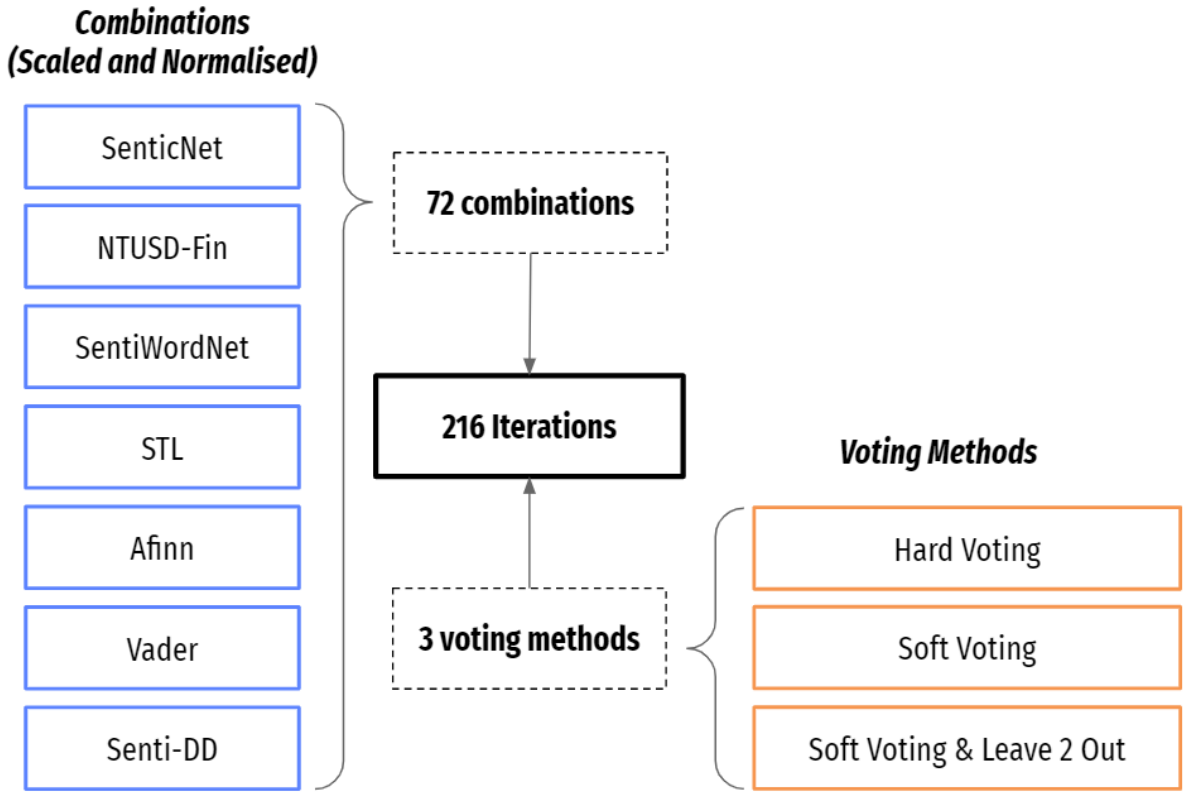

Figure 56: Lexicon Ensemble Overall Framework diagram

We will iterate 216 different combinations of lexicons and voting methods for each test dataset and see which gives us the best results. We will test on Data 2 and 3 since Data 1 was used to generate Senti-DD while Data 4 was used to generate StockTwitLexi (STL). The code for this section is saved as *symbolic_iterations.ipynb*.

Test Results

Individual Lexicons

	Data 2			
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
ntusd	0.662857	0.739629	0.691419	0.795066
stocktwitlexi	0.620571	0.755882	0.617047	0.975332
senticnet	0.668571	0.760726	0.672993	0.874763
vader	**0.749714**	**0.785504**	**0.811741**	**0.760911**
sentiwordnet	0.635429	0.691787	0.704724	0.679317
afinn	0.717714	0.740818	0.828638	0.669829
sentidd	0.606857	0.542553	0.906667	0.387097

Figure 57: Individual Lexicon Test Results for Data 2

	Data 3			
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
ntusd	**0.761413**	**0.790943**	**0.809892**	**0.772861**
stocktwitlexi	0.638243	0.75553	0.624038	0.957227
senticnet	0.584841	0.668956	0.625964	0.718289
vader	0.546942	0.528674	0.673516	0.435103
sentiwordnet	0.55814	0.585956	0.647059	0.535398
afinn	0.538329	0.508257	0.67233	0.408555
sentidd	0.479759	0.30254	0.696809	0.193215

Figure 58: Individual Lexicon Test Results for Data 3

	Average of Data 2 and 3			
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
ntusd	**0.712135**	**0.765286**	**0.750655**	**0.783964**
stocktwitlexi	0.629407	0.755706	0.620543	0.96628
senticnet	0.626706	0.714841	0.649478	0.796526
vader	0.648328	0.657089	0.742628	0.598007
sentiwordnet	0.596784	0.638872	0.675892	0.607358
afinn	0.628022	0.624538	0.750484	0.539192
sentidd	0.543308	0.422547	0.801738	0.290156

Figure 59: Individual Lexicon Test Results for Average of Data 2 and 3

Lexicon Ensemble (Top 10 Results)

Data 2

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.786286	0.84166	0.759939	0.943074
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.779429	0.837405	0.75303	0.943074
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.776	0.834179	0.752672	0.935484
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.817143	0.855335	0.816926	0.897533
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.781714	0.837447	0.759259	0.933586
ntusd_stocktwitlexi_afinn_vader_soft	0.780571	0.836457	0.758887	0.931689
ntusd_stocktwitlexi_vader_soft	0.774857	0.832909	0.753067	0.931689
ntusd_sentiwordnet_vader_soft	0.796571	0.844406	0.78282	0.916509
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.817143	0.855072	0.818024	0.895636
ntusd_sentiwordnet_afinn_vader_soft	0.796571	0.844406	0.78282	0.916509

Figure 60: Lexicon Ensemble Test Results for Data 2, Top 10

Full results can be found in the *Appendix*.

Data 3

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.726098	0.773504	0.747934	0.800885
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.72093	0.774059	0.734127	0.818584
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.726098	0.774468	0.745902	0.80531
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.70801	0.751284	0.747445	0.755162
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.718346	0.767921	0.740082	0.797935
ntusd_stocktwitlexi_afinn_vader_soft	0.718346	0.768249	0.739427	0.79941
ntusd_stocktwitlexi_vader_soft	0.721792	0.771408	0.741497	0.803835
ntusd_sentiwordnet_vader_soft	0.718346	0.758315	0.76	0.756637
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.703704	0.747059	0.744868	0.749263
ntusd_sentiwordnet_afinn_vader_soft	0.715762	0.756637	0.756637	0.756637

Figure 61: Lexicon Ensemble Test Results for Data 3, Top 10

Full results can be found in the *Appendix*.

Average of Data 2 and 3

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	**0.756192**	**0.807582**	**0.753936**	**0.871979**
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.750179	0.805732	0.743579	0.880829
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.751049	0.804324	0.749287	0.870397
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.762577	0.803309	0.782185	0.826348
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.75003	0.802684	0.749671	0.865761
ntusd_stocktwitlexi_afinn_vader_soft	0.749459	0.802353	0.749157	0.865549
ntusd_stocktwitlexi_vader_soft	0.748324	0.802159	0.747282	0.867762
ntusd_sentiwordnet_vader_soft	0.757459	0.80136	0.77141	0.836573
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.760423	0.801066	0.781446	0.822449
ntusd_sentiwordnet_afinn_vader_soft	0.756167	0.800521	0.769729	0.836573

Figure 62: Lexicon Ensemble Test Results for Average of Data 2 and 3, Top 10

Full results can be found in the *Appendix*.

Discussion of results

Individual Lexicons

As we can see, when comparing amongst individual lexicons, NTUSD-Fin had the highest average F1_Score of 0.765 with StockTwitLexi coming in close as the 2nd with 0.755. This shows that this constructed lexicon performs well individually, beating other well-known lexicons such as SenticNet and Vader for the purpose of classifying financial sentiment of cryptocurrency microblog texts.

On the other hand, Senti-DD does not perform well individually and came in last amongst all 7 lexicons used. This maybe due to the lack of *directional word and directional-dependent word* pairings present in such corpus, such that the scores calculated are not fine-grained enough.

Lexicon Ensemble

The combination of NTUSD-Fin, StockTwitLexi, Afinn, Vader and Senti-DD lexicons using Leave 2 Out method has the highest F1_Score of 0.808. This beats the highest individual F1_Score by a huge margin.

This proves our hypothesis right that an ensemble of lexicons will help to boost performance. In fact, 51.7% (103/199) of the ensemble variants has an F1_Score higher than the highest individual F1_Score.

We can also see that Senti-DD, a lexicon that did not perform well individually, is actually present in the best Lexicon Ensemble. It is also present in 3 of the top 5 best performing variants. This implies the strong complementary value of the lexicon when combining with other lexicons.

To confirm Senti-DD's value in the ensemble variants, I counted the number of times each lexicon score was left out in our *Leave 2 Out* method for being the highest and lowest score among the rest. Since there is no huge skew towards Senti-DD, we can reject the hypothesis that ensemble variants with Senti-DD perform well because Senti-DD is being left out.

{'senticnet': 1753, 'ntusd': 2551, 'sentiwordnet': 3328, 'afinn': 3449, 'vader': 2736, 'stocktwitlexi': 490, 'sentidd': 4943, 'actual_class': 0}
{'senticnet': 7454, 'ntusd': 1836, 'sentiwordnet': 561, 'afinn': 223, 'vader': 4644, 'stocktwitlexi': 2725, 'sentidd': 1807, 'actual_class': 0}

Figure 63: Count of lexicons being left out in Data 2 (Highest and Lowest Score)

{'senticnet': 3888, 'ntusd': 4979, 'sentiwordnet': 4024, 'afinn': 3621, 'vader': 3848, 'stocktwitlexi': 813, 'sentidd': 4369, 'actual_class': 0}
{'senticnet': 9109, 'ntusd': 2946, 'sentiwordnet': 1530, 'afinn': 567, 'vader': 4201, 'stocktwitlexi': 5205, 'sentidd': 1984, 'actual_class': 0}

Figure 64: Count of lexicons being left out in Data 3 (Highest and Lowest Score)

Post-Analysis of SenticNet Polarity Classfication API

Now, we will do a deeper analysis on the SenticNet Polarity Classfication API by looking at the instances where the API predicted wrongly and was out-voted by all the other lexicons 6 to 1. This post-analysis can be found in *baseline_senticnet_analysis.ipynb*.

We combined both Data 2 and 3 and get the individual scores from all the lexicons. We converted the fine-grained scores to class labels and we counted how many of the other lexicons voted with or against SenticNet (column labels *vote_w_sn* and *vote_a_sn* respectively). Then, we split the DataFrame into when SenticNet predicted correctly or wrongly. Below shows the distribution of the number of lexicons that voted with SenticNet in the resultant 2 DataFrames.

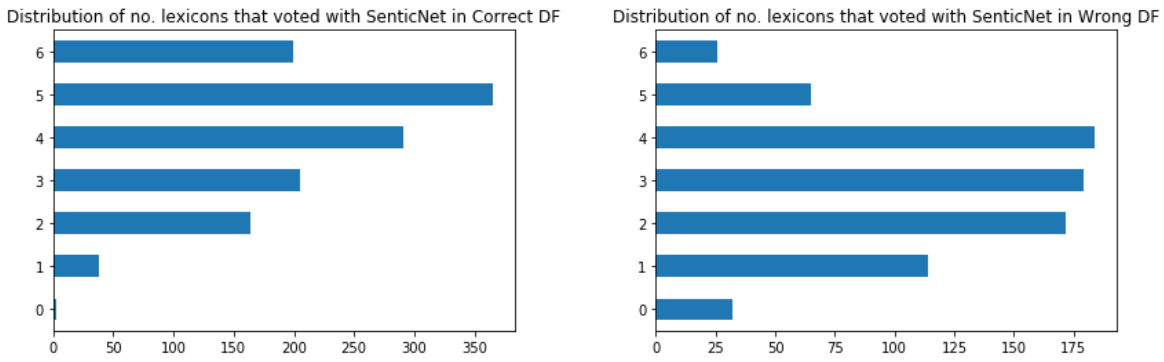

Figure 65: Distribution of the number of lexicons that voted with SenticNet

We can see the SenticNet generally voted with the majority (around 4-5 lexicons) when it is correct while the voting was more contentious (around 3) when it was wrong. This highlights

the fact that SenticNet perform generally well as an individual lexicon, which is backed by our results where it came in 3rd behind NTUSD-Fin and StockTwitLexi.

We continued our analysis on SenticNet's shortcomings by picking out instances where SenticNet predicted wrong & all lexicons voted against it. The full table can be found in the *Appendix* but I will highlight some examples here.

	senticnet	actual_class	text	processed_words
0	0	1	rt happy wednesday here are my topthingstoknowtoday in financial markets earnings in the morning	happy earnings
1	0	1	impressive	impressive
2	0	1	rt added to chart setup still looks good for continuation	chart setup good
3	1	0	guaranteed to dump every time	dump time

Figure 66: Select instances where SenticNet predicted wrongly and all lexicons voted against it

	processed_words	**SenticNet Score**	**SenticNet Polarity**
0	happy, earnings	0.268 + (-0.789) = -0.521	Negative
1	impressive	-0.918	Negative
2	chart, setup, good	0.203 + (-0.821) + 0.356 = -0.262	Negative
3	dump, time	0.582 + (-0.344) = 0.238	Positive

Figure 67: SenticNet score calculation for select instances

In row 1, 'happy earnings' is clearly a positive sentiment but the negative score from 'earnings' outweighed the score from 'happy'. In row 2, 'impressive' has a negative score surprisingly. In row 3, the negative score of 'setup' implies that the context of the word is taken wrongly - it refers to the trade setup instead of a scheme or trick. In row 4, the word 'dump', which was explained in *Need to construct new Lexicons* to have a negative sentiment, has a positive score in SenticNet instead.

From this analysis, we can see that although SenticNet perform generally well, it has shortcomings in the financial cryptocurrency domain due to the different contextual meanings of the tokens.

Post-Analysis of Vader

Since Vader is one of the more popular lexicons used in the literature, I will analyse the value of the lexicon in terms of the added benefit it brings to our best performing Lexicon Ensemble. To do so, I will take a look of instances in both Data 2 and 3 in which the addition of Vader helped our Lexicon Ensemble to predict correctly instead of wrongly without it.

We simply perform the *Soft Voting & Leave 2 out* voting mechanism on the variants shown in the figure below and pick out instances where our best ensemble predicted wrongly and the ensemble without Vader predicted correctly. There are 33 instances and full table can be found in the *Appendix*. I will highlight a select few below.

Variant	**Lexicons Involved**
Best performing Lexicon Ensemble	NTUSD-Fin, StockTwitLexi, Afinn, Vader, Senti-DD
Best performing Lexicon Ensemble without Vader	NTUSD-Fin, StockTwitLexi, Afinn, Senti-DD

Figure 68: Lexicons involved in analysis of Vader

	actual_class	leave2soft_w_vader	leave2soft_wo_vader	text	processed_words	vader
0	0	-0.000514	0.020148	rt saturday country descends into chaos sunday chaos deepens monday rallies for no reason trump declares	[chaos, chaos, no]	-0.8625
1	0	-0.018432	0.009658	hope not i added some during the panic selloff	[hope, panic]	-0.1027
2	0	-0.041898	0.016542	making a comeback and dumping up points in that now	[dumping]	-0.3182
3	0	-0.005511	0.006697	once nothing is fixed by the third hike its gonna be a bigger basis point for the next and market gonna dump already planned in advance i bet	[dump]	-0.3818
4	0	-0.016744	0.011008	china screwed us all	[screwed]	-0.4939

Figure 69: Select instances where Vader helped our Lexicon Ensemble to predict correctly instead of wrongly

In row 2 and 3, Vader considered the words “dump” and “dumping” as moderately negative which helped to push the ensemble sentiment score down such that it became negative. This tokens’ negative scores using Vader is in contrast to the SenticNet’s positive score for a similar term “dump” explained in the previous sub-section.

The ensemble scores without Vader tend to be on the borderline of 0 and as such, Vader’s input helps to capture the relevant terms and skew the score in the right direction, improving the overall ensemble’s classification ability. This can be seen in rows 0, 1 and 4.

Approach Conclusion

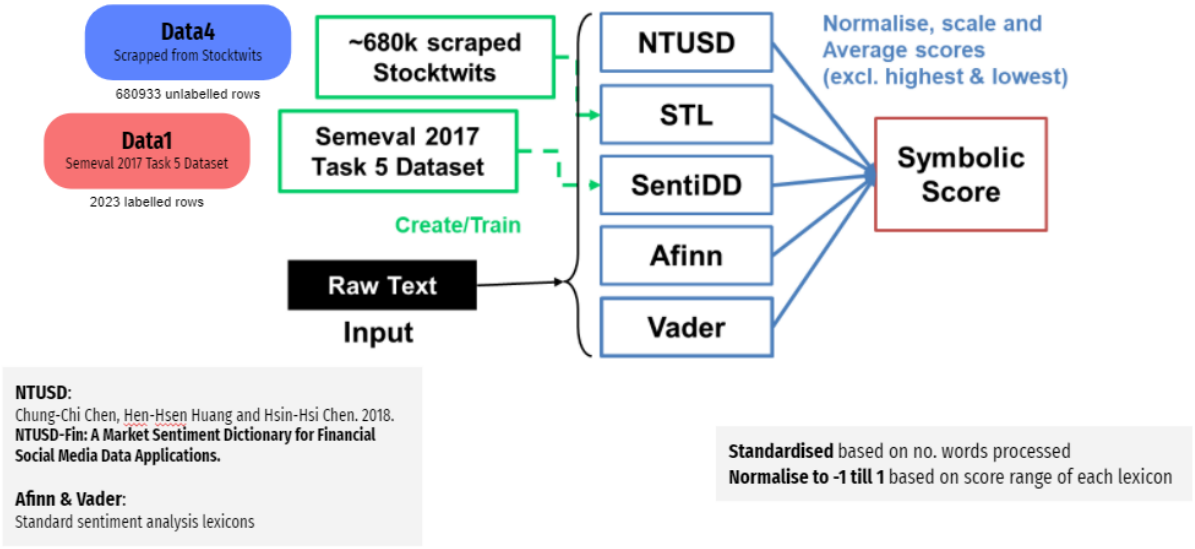

Figure 70: Symbolic Approach Proposed Architecture diagram

Backed by our experimental results, we propose the use of Ensemble Lexicon (NTUSD-Fin, StockTwitLexi, Afinn, Vader and Senti-DD lexicons using Leave 2 Out method) to calculate the Symbolic Score. The architecture above highlights the whole process of generating this score.

6. Sub Symbolic Approach

Language Models - Deep Neural Networks

We proposed the use of Language Models for our Sub Symbolic Approach. This differentiates from traditional rule based models or simple vectorisation techniques such as Bag-Of-Words, TF-IDF and word2vec with a key advantage of overcoming the biggest limitation of previous method: polysemy disambiguation (a word with different meanings). Language models being able to sieve out the different meaning of a word in different context.

Models

We will explore the use of 2 popular networks used in Natural Language Processing (NLP) for our task:

1. BERT — saved as *bert.ipynb*
2. RNN — saved as *rnn.ipynb*
    a. Unidirectional LSTM
    b. Bidirectional LSTM
    c. Unidirectional GRU
    d. Bidirectional GRU

Training and Validation Datasets

We will train and validate our models using a 80/20 split with the following 3 datasets:

1. Data 1
2. Data 4 (A sample of 4000 of since its original size ~600k is too huge)
3. Combined Data from Data 1 and 4 (Equal proportion of 2023 rows each)

They can be found under *data/subsymbolic* with these filenames.

Data	**Purpose**	**Filename**
Data 1	Train	*data1_train.csv*
	Validation	*data1_val.csv*
Data 4	Train	*data4_train.csv*
	Validation	*data4_val.csv*
Combined	Train	*combined_data_train.csv*
	Validation	*combined_data_val.csv*

Figure 71: Sub Symbolic train and validation data filenames

Models

Models can be found under *model* with these file naming convention.

Model	**Naming Convention**
BERT	bert_model_<*data1/data4/combined_data*>.pt
RNN	< *data1/data4/combined*>_<*uni/bi*><*lstm/gru*>_*model.pt*

Figure 72: Sub Symbolic model file naming convention

Testing Datasets

The various models will be tested on Data 2 and 3 and the best model will be selected to be our Sub Symbolic model. They can be found under *data/subsymbolic* with these filenames.

Data	**Purpose**	**Filename**
Data 2	Test	*data2_test.csv*
Data 3	Test	*data3_test.csv*

Figure 73: Sub Symbolic test data filenames

Hyperparameters

Various hyperparameters were experimented and these are the ones that produces the best performance for each language model.

RNNs	
Hyperparameter	**Value**
Number of Epochs	15
Learning Rate	5e-04
Hidden Dimension	512
Number of RNN layers	4
Dropout	0.5
Batch Size	32
Optimiser	Adam

Figure 74: RNN Hyperparameters

BERT	
Hyperparameter	**Value**
Number of Epochs	5
Learning Rate	5e-05
Batch Size	16
Optimiser	AdamW

Figure 75: BERT Hyperparameters

Implementation

We used PyTorch to develop both the RNN and BERT models and ran our code on Google Colaboratory-hosted GPU Runtime.

1	**Data input**
	We use the created datasets for training and validation purposes.

```
data1_train_filepath = "data1_train.csv"
data1_val_filepath = "data1_val.csv"
data4_train_filepath = "data4_train.csv"
data4_val_filepath = "data4_val.csv"
combined_data_train_filepath = "combined_data_train.csv"
combined_data_val_filepath = "combined_data_val.csv"
```

2	**Data Loaders**
	Custom data loaders were created to load the data to be used by our models. Detailed implementation can be found under *bert.ipynb* and *rnn.ipynb*.

```
class CustomDataset(data.Dataset):
    def __init__(self, df, fields, is_test=False, **kwargs):
        examples = []
        for i, row in df.iterrows():
            label = row.Label #if not is_test else None
            text = row.text_cleaned
            examples.append(data.Example.fromlist([text, label], fields))

        super().__init__(examples, fields, **kwargs)

    @staticmethod
    def sort_key(ex):
        return len(ex.text)

    @classmethod
    def splits(cls, fields, train_df, val_df=None, test_df=None, **kwargs):
        train_data, val_data, test_data = (None, None, None)
        data_field = fields

        if train_df is not None:
            train_data = cls(train_df.copy(), data_field, **kwargs)
        if val_df is not None:
            val_data = cls(val_df.copy(), data_field, **kwargs)
        if test_df is not None:
            test_data = cls(test_df.copy(), data_field, False, **kwargs)

        return tuple(d for d in (train_data, val_data, test_data) if d is not None)
```

RNN Data Loader

```python
def create_data_loader(root,train_filename,dev_filename,batch_size=16):
    X_train, y_train, X_val, y_val, X_ = data_handler(root, train_filename, dev_filename)

    # Concatenate train data and test data
    all_texts = np.concatenate([X_train, X_val])

    # Encode our concatenated data
    encoded_texts = [tokenizer.encode(sent, add_special_tokens=True) for sent in all_texts]

    # Find the maximum length
    max_len = max([len(sent) for sent in encoded_texts])
    #print('Max length: ', max_len)

    # Print sentence 0 and its encoded token ids
    token_ids = list(preprocessing_for_bert([X_[0]])[0].squeeze().numpy())
    #print('Original: ', X_[0])
    #print('Token IDs: ', token_ids)

    # Run function `preprocessing_for_bert` on the train set and the validation set
    print('Tokenizing data...')
    train_inputs, train_masks = preprocessing_for_bert(X_train)
    val_inputs, val_masks = preprocessing_for_bert(X_val)

    # Convert other data types to torch.Tensor
    train_labels = torch.tensor(y_train)
    val_labels = torch.tensor(y_val)

    # Create the DataLoader for our training set
    train_data = TensorDataset(train_inputs, train_masks, train_labels)
    train_sampler = RandomSampler(train_data)
    train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)

    # Create the DataLoader for our validation set
    val_data = TensorDataset(val_inputs, val_masks, val_labels)
    val_sampler = SequentialSampler(val_data)
    val_dataloader = DataLoader(val_data, sampler=val_sampler, batch_size=batch_size)

    return train_dataloader, val_dataloader, y_val
```

BERT Data Loader

3	**Building Models**
	Below shows the implementation classes for RNN and BERT. The hyperparameters and model type mentioned in *Models* and *Hyperparameters* are passed into the RNN model.

```python
# RNN Models
class RNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, output_dim, n_layers,
                 bidirectional, rnn_type, dropout, pad_idx):
        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx = pad_idx)
        self.rnn_type = rnn_type

        # LSTM
        if self.rnn_type == 'lstm':
            self.rnn = nn.LSTM(embedding_dim,
                               hidden_dim,
                               num_layers=n_layers,
                               bidirectional=bidirectional,
                               dropout=dropout,
                               batch_first=True)
        # GRU
        else:
            self.rnn = nn.GRU(embedding_dim,
                              hidden_dim,
                              num_layers=n_layers,
                              bidirectional=bidirectional,
                              dropout=dropout,
                              batch_first=True)

        self.fc1 = nn.Linear(hidden_dim * 2, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, 1)

        self.dropout = nn.Dropout(dropout)

    def forward(self, text, text_lengths):
        embedded = self.embedding(text)
        # pack sequence
        packed_embedded = nn.utils.rnn.pack_padded_sequence(embedded,
                                                            text_lengths.to('cpu'),
                                                            batch_first=True)
        # lstm
        if self.rnn_type == 'lstm':
            packed_output, (hidden, cell) = self.rnn(packed_embedded)
        # gru
        else:
            packed_output, hidden = self.rnn(packed_embedded)

        hidden = self.dropout(torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim = 1))
        output = self.fc1(hidden)
        output = self.dropout(self.fc2(output))

        return output
```

RNN Model

```
# Create the BertClassfier class
class BertClassifier(nn.Module):
    def __init__(self, freeze_bert=False):
        super(BertClassifier, self).__init__()
        # Specify hidden size of BERT, hidden size of our classifier, and number of labels
        D_in, H, D_out = 768, 50, 2

        # Instantiate BERT model
        self.bert = BertModel.from_pretrained('bert-base-uncased')

        # Instantiate an one-layer feed-forward classifier
        self.classifier = nn.Sequential(
            nn.Linear(D_in, H),
            nn.ReLU(),
            nn.Linear(H, D_out)
        )

        # Freeze the BERT model
        if freeze_bert:
            for param in self.bert.parameters():
                param.requires_grad = False

    def forward(self, input_ids, attention_mask):
        # Feed input to BERT
        outputs = self.bert(input_ids=input_ids,
                            attention_mask=attention_mask)

        # Extract the last hidden state of the token `[CLS]` for classification task
        last_hidden_state_cls = outputs[0][:, 0, :]

        # Feed input to classifier to compute logits
        logits = self.classifier(last_hidden_state_cls)
```

BERT Model

4	**Training ,Validation and Testing**
	The models are trained and validated based on the datasets mentioned in *Training and Validation Datasets* before being tested on the datasets mentioned in *Testing Datasets*.

Train and Validation Results

Models trained with Data 1

Model	Train Loss	Validation Loss	Validation Accuracy	AUC	F1 Score (Macro Average)	F1 Score (Weighted Average)
Uni-LSTM	0.48066	0.46291	0.79052	0.84455	0.7770	0.7989
Bi-LSTM	0.43244	0.41731	0.82166	0.86922	0.7865	0.8094
Uni-GRU	0.43386	0.45743	0.80735	0.84153	0.7802	0.8028
Bi-GRU	0.39468	0.40731	0.82521	0.87977	0.8097	0.8308
BERT	**0.303580**	**0.372742**	**85.29**	**0.9154**	**0.8505**	**0.8664**

Figure 76: Sub Symbolic Train and Validation Results for models trained with Data 1

ROC Curve for RNNs

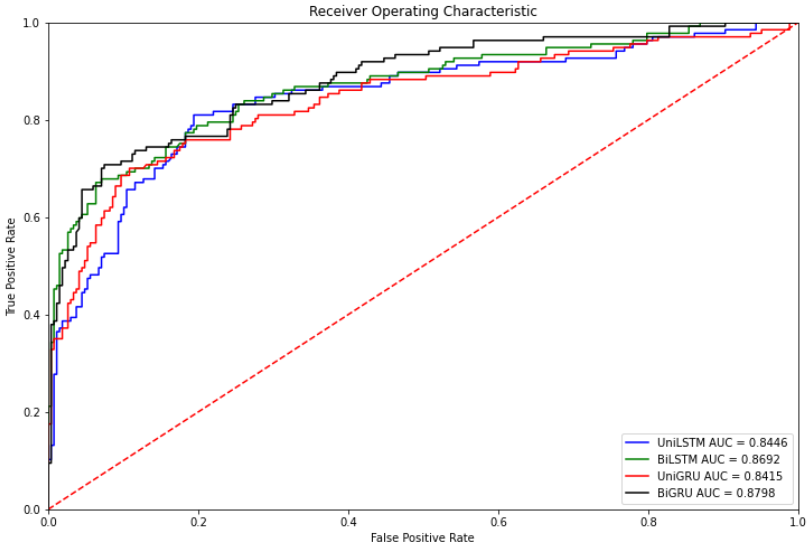

Figure 77: ROC Curves for RNN models trained with Data 1

ROC Curve for BERT

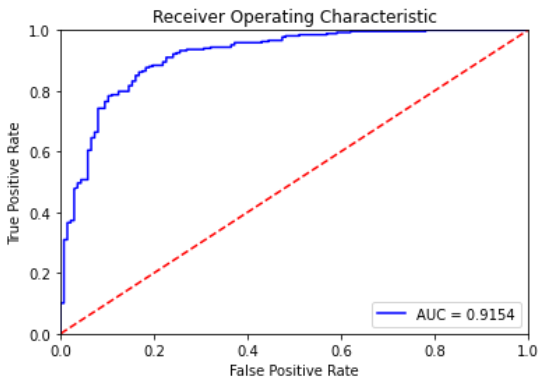

Figure 78: ROC Curve for BERT model trained with Data 1

Models trained with Data 4

Model	Train Loss	Validation Loss	Validation Accuracy	AUC	F1 Score (Macro Average)	F1 Score (Weighted Average)
Uni-LSTM	0.49662	0.51600	0.76000	0.72088	0.6466	0.7365
Bi-LSTM	0.46139	0.50551	0.77125	0.71928	0.6449	0.7221
Uni-GRU	0.55977	0.51993	0.76750	0.70794	0.6308	0.7318
Bi-GRU	0.56447	0.52622	0.76500	0.70462	0.6281	0.7169
BERT	**0.449436**	**0.519721**	**76.88**	**0.7656**	**0.6891**	**0.7629**

Figure 79: Sub Symbolic Train and Validation Results for models trained with Data 4

ROC Curve for RNNs

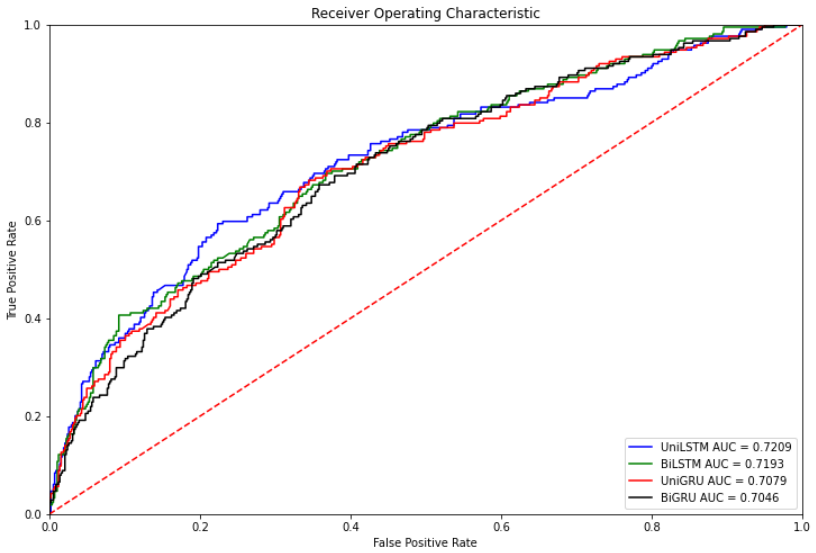

Figure 80: ROC Curves for RNN models trained with Data 4

ROC Curve for BERT

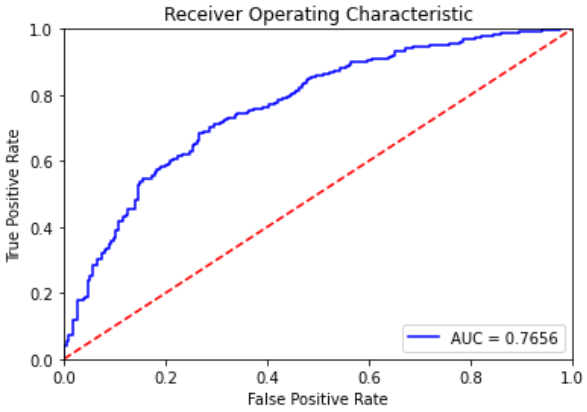

Figure 81: ROC Curve for BERT model trained with Data 4

Models trained with Combined Data

Model	Train Loss	Validation Loss	Validation Accuracy	AUC	F1 Score (Macro Average)	F1 Score (Weighted Average)
Uni-LSTM	0.50449	0.52301	0.74399	0.74670	0.6667	0.7259
Bi-LSTM	0.53975	0.49742	0.75481	0.78793	0.6839	0.7422
Uni-GRU	0.49132	0.53118	0.75361	0.73452	0.6711	0.7302
Bi-GRU	0.48699	0.49455	0.74399	0.76909	0.6623	0.7235
BERT	**0.393493**	**0.437550**	**80.51**	**0.8480**	**0.7568**	**0.8023**

Figure 82: Sub Symbolic Train and Validation Results for models trained with Combined Data

ROC Curve for RNNs

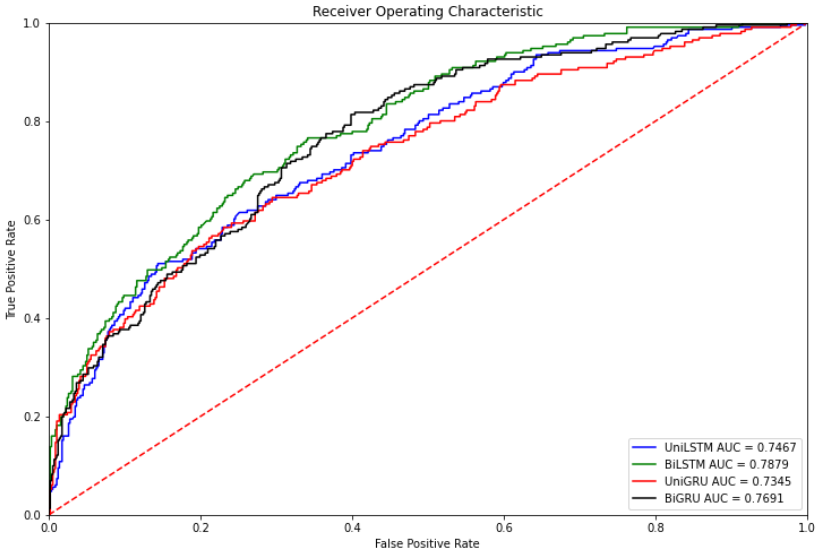

Figure 83: ROC Curve for RNN models trained with Combined Data

ROC Curve for BERT

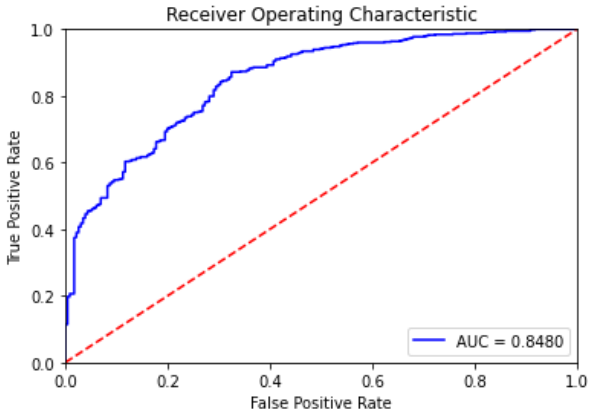

Figure 84: ROC Curve for BERT model trained with Combined Data

Test Results

Data 2

Experiment	Accuracy	Precision	Recall	F1_score
data1_bigru_model	0.698286	0.73357	0.783681	0.757798
data1_bilstm_model	0.674286	0.695161	0.817837	0.751526
data1_unigru_model	0.656	0.657382	0.895636	0.758233
data1_unilstm_model	0.674286	0.696429	0.814042	0.750656
data1_bert_model	**0.774857**	**0.761905**	**0.910816**	**0.829732**
data4_bigru_model	0.624	0.621622	0.960152	0.754661
data4_bilstm_model	0.657143	0.65742	0.899431	0.759615
data4_unigru_model	0.633143	0.630711	0.943074	0.755894
data4_unilstm_model	0.646857	0.650552	0.893738	0.752998
data4_bert_model	0.635429	0.633676	0.935484	0.755556
combined_bigru_model	0.684571	0.682678	0.889943	0.772652
combined_bilstm_model	0.682286	0.675599	0.908918	0.775081
combined_unigru_model	0.658286	0.649606	0.939279	0.768037
combined_unilstm_model	0.656	0.657821	0.893738	0.757844
combined_bert_model	0.748571	0.740219	0.897533	0.811321

Figure 85: Sub Symbolic Test Results for Data 2

Data 3

Experiment	Accuracy	Precision	Recall	F1_score
data1_bigru_model	0.624462	0.676385	0.684366	0.680352
data1_bilstm_model	0.621878	0.672439	0.687316	0.679796
data1_unigru_model	0.624462	0.640046	0.815634	0.71725
data1_unilstm_model	0.612403	0.659664	0.69469	0.676724
data1_bert_model	0.726098	0.748619	0.79941	0.773181
data4_bigru_model	0.655469	0.638446	0.945428	0.762188
data4_bilstm_model	0.677003	0.662728	0.910029	0.766936
data4_unigru_model	0.645134	0.62963	0.952802	0.758216
data4_unilstm_model	0.677864	0.658664	0.930678	0.771394
data4_bert_model	0.72093	0.703917	0.90118	0.790427
combined_bigru_model	0.678725	0.669633	0.887906	0.763475
combined_bilstm_model	0.705426	0.692661	0.890855	0.779355
combined_unigru_model	0.678725	0.657704	0.938053	0.773252
combined_unilstm_model	0.668389	0.658037	0.899705	0.760125
combined_bert_model	**0.73385**	**0.727497**	**0.870206**	**0.792478**

Figure 86: Sub Symbolic Test Results for Data 3

Average of Data 2 and 3

Experiment	Accuracy	Precision	Recall	F1_score
data1_bigru_model	0.661374	0.704977	0.734023	0.719075
data1_bilstm_model	0.648082	0.6838	0.752576	0.715661
data1_unigru_model	0.640231	0.648714	0.855635	0.737742
data1_unilstm_model	0.643344	0.678046	0.754366	0.71369
data1_bert_model	0.750478	0.755262	0.855113	0.801457
data4_bigru_model	0.639735	0.630034	0.95279	0.758424
data4_bilstm_model	0.667073	0.660074	0.90473	0.763276
data4_unigru_model	0.639138	0.63017	0.947938	0.757055
data4_unilstm_model	0.662361	0.654608	0.912208	0.762196
data4_bert_model	0.67818	0.668797	0.918332	0.772992
combined_bigru_model	0.681648	0.676156	0.888924	0.768064
combined_bilstm_model	0.693856	0.68413	0.899887	0.777218
combined_unigru_model	0.668505	0.653655	0.938666	0.770645
combined_unilstm_model	0.662195	0.657929	0.896722	0.758984
combined_bert_model	**0.741211**	**0.733858**	**0.88387**	**0.8019**

Figure 87: Sub Symbolic Test Results for Average of Data 2 and 3

Discussion of results

From our results, we can see the BERT model generally outperforms all the other RNNs variants – Uni-LSTM, Bi-LSTM, Uni-GRU and Bi-GRU for both test datasets. This can be due to the fact that the transformer architecture of BERT allows for the context of words to be better learning since the model can learn from both directions simultaneously. This is in contrast to Uni-directional and Bi-directional RNNs, which learns in one direction or both directions before concatenation respectively, and may have caused them to lose the true context of words.

In all of our experiments, the BERT model trained on the combined dataset performs best, with an average F1_score of 0.8019. The use of a combined dataset contributed by a 50/50 split between Data 1 and 4 could have given a better generalisation of the model, allowing it to outperform those that are only trained on either dataset.

Approach Conclusion

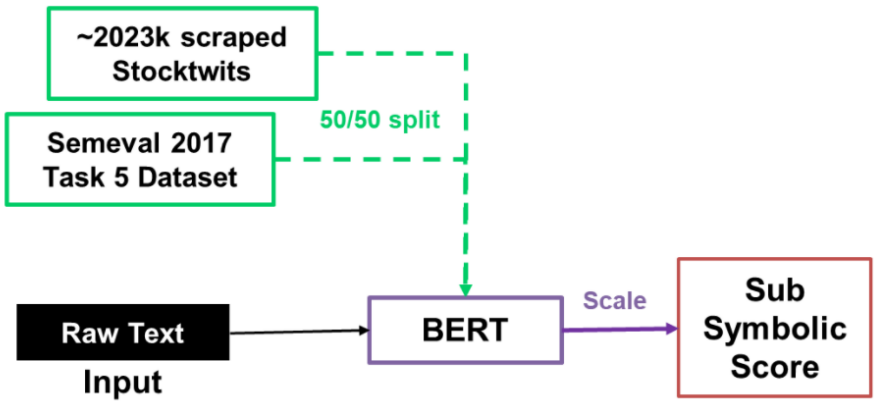

Figure 88: Sub Symbolic Approach Proposed Architecture diagram

Backed by our experimental results, we propose the use of the BERT Model trained on the combined dataset to calculate the Sub Symbolic Score. The SoftMax probability is scaled to the range of -1 to 1 for this purpose. The architecture above highlights the whole process of generating this score.

7. Hybrid Approach

Combining Symbolic and Sub Symbolic Approach

After generating both Symbolic and Sub Symbolic scores via the methods selected under *Approach Conclusion* of both subsections, we propose the combination of scores by giving each score a percentage weightage.

We run a simple iteration of 20/80%, 40/60%, 60/40% and 80/20% weightage for the Symbolic/Sub Symbolic scores and generate the test results for Data 2 and 3.

The code for this section is saved as *symbolic_subsymbolic.csv*.

Test Results

Data 2				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
data2_symbolic0.2	0.752	0.813734	0.742947	0.899431
data2_symbolic0.4	0.761143	0.820908	0.748437	0.908918
data2_symbolic0.6	0.773714	0.831058	0.755039	0.924099
data2_symbolic0.8	**0.794286**	**0.845626**	**0.771518**	**0.935484**

Figure 89: Hybrid Approach Test Results for Data 2

Data 3				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
data3_symbolic0.2	0.734711	0.792453	0.729529	0.867257
data3_symbolic0.4	0.737295	0.794058	0.732254	0.867257
data3_symbolic0.6	0.744186	0.799189	0.737828	0.871681
data3_symbolic0.8	**0.756245**	**0.807089**	**0.750317**	**0.873156**

Figure 90: Hybrid Approach Test Results for Data 3

Average				
Experiment	**Accuracy**	**F1_score**	**Precision**	**Recall**
symbolic0.2	0.743356	0.803094	0.736238	0.883344
symbolic0.4	0.749219	0.807483	0.740346	0.888088
symbolic0.6	0.75895	0.815124	0.746434	0.89789
symbolic0.8	**0.775266**	**0.826358**	**0.760918**	**0.90432**

Figure 91: Hybrid Approach Test Results for Average of Data 2 and 3

Discussion of Results

Our results show that giving a 80% and 20% weightage on the Symbolic score and Sub Symbolic score respectively produced the best results.

Approach Conclusion

The overall results of the Hybrid Approach beat the individual Symbolic or Sub Symbolic implying the possible synergies that arise from this combination of scores

Test F1_score of chosen ensemble/model		
Symbolic	**Sub Symbolic**	**Hybrid**
0.8076	0.8019	**0.8264**

Figure 92: Test F1 Score of chosen ensemble/model

Test Accuracy of chosen ensemble/model		
Symbolic	**Sub Symbolic**	**Hybrid**
0.7562	0.7412	**0.7753**

Figure 93: Test Accuracy of chosen ensemble/model

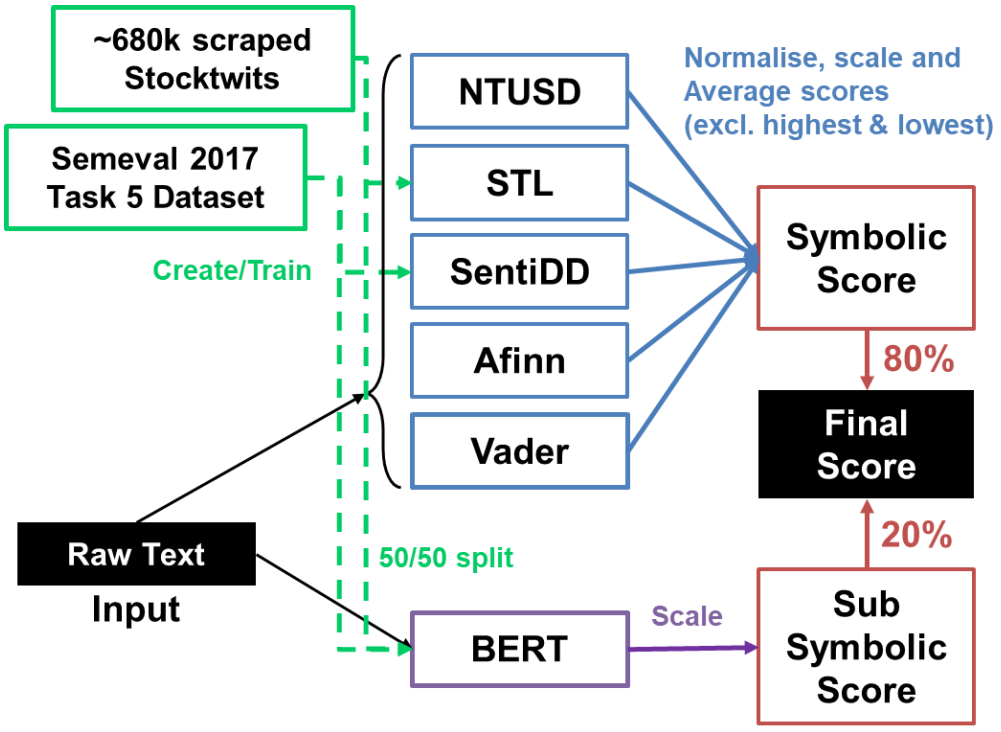

Figure 94: Hybrid Approach Proposed Architecture diagram

Backed by our experimental results, we propose the use of the combination of Symbolic and Sub Symbolic Architectures with a 80% to 20% split between the generated scores. This diagram above depicts the full architecture.

8. Overall Conclusion and Future Works

Overall Conclusion

In this project, we discussed the methods which can be employed to predict financial sentiment of social media cryptocurrency content.

We propose the creation of 2 new lexicons StockTwitLexi and Senti-DD and used empirical evidence to back our selection of data used to create these lexicons.

Then, we explored the Lexicon Ensemble for our Symbolic Approach and found that the combination of NTUSD-Fin, StockTwitLexi, Afinn, Vader and Senti-DD lexicons using Leave 2 Out method performs the best. This combination outperforms the use of individual lexicons. Furthermore, both the created lexicons proved useful in the Ensemble methods, validating their value in our task.

To supplement our Symbolic Approach, we further explored the use of Language Models as our Sub Symbolic Approach. Our results have shown that the BERT model trained on the combined dataset of Data 1 and 4 performs best. This model outperforms the various variants of RNNs (Uni-LSTM, Bi-LSTM, Uni-GRU and Bi-GRU).

Lastly, we propose a combined architecture under our Hybrid Approach where we take the scores generated from both Symbolic and Sub Symbolic Approaches. Our experiments show that the combination of both approaches via a 80/20 split performs best. The Hybrid Approach generated results that beat individual approaches in terms of Test F1_score and Test Accuracy.

Future Works

With the comprehensive research and experiments done in this project, we are now able to better predict the financial sentiments of social media cryptocurrency content. This is a gateway to perform many upstream financial prediction tasks such as volatility forecasting, asset allocation, and market trend predictions. Future work can also focus on the fusion of the

processed sentiment data using our architecture with other structured data coming from the financial market to build better financial market leading indicators and prediction models.

9. References

[1] *Cryptocurrency prices, charts and market capitalizations*. CoinMarketCap. (n.d.). Retrieved March 24, 2022, from https://coinmarketcap.com/

[2] SoFi. (2022, March 7). *Bitcoin price history: Price of bitcoin 2009 - 2022*. SoFi. Retrieved March 24, 2022, from https://www.sofi.com/learn/content/bitcoin-price-history/

[3] Farell, R. (n.d.). *An analysis of the cryptocurrency industry*. Retrieved March 24, 2022, from https://www.researchgate.net/publication/304089371_An_Analysis_of_the_Cryptocurrency_Industry

[4] Yahoo! (2022, March 24). *Bitcoin USD (BTC-USD) price, value, news & history*. Yahoo! Finance. Retrieved March 24, 2022, from https://finance.yahoo.com/quote/BTC-USD/

[5] 13, L. V. M. (2020, May 10). *How rogue traders make a fortune on volatile markets*. Literary Hub. Retrieved March 24, 2022, from https://lithub.com/how-rogue-traders-make-a-fortune-on-volatile-markets/

[6] Dolan, R. J. (2002) Science 298, 1191–1194

[7] Damasio, A. R. (1994) Descartes' error : emotion, reason, and the human brain. (Putnam), pp. xix, 312 p.+

[8] Kahneman, D & Tversky, Amos (1979) Prospect Theory: An Analysis of Decision under Risk. (Econometrica), pp. 263–291.

[9] Nofsinger, J. (2005) Journal of Behaviour Finance. 6, 144–160

[10] *Refinitiv Alt Data*. (n.d.). Retrieved March 24, 2022, from https://www.refinitiv.cn/content/dam/marketing/en_us/documents/fact-sheets/alternative-data-fact-sheet.pdf

[11] Hansen, K. B., & Borch, C. (n.d.). *Alternative data and sentiment analysis: Prospecting non-standard data in machine learning-driven finance*. Retrieved March 24, 2022, from https://journals.sagepub.com/doi/abs/10.1177/20539517211070701

[12] Cloudflare. (n.d.). *What is data scraping?* Retrieved March 24, 2022, from https://www.cloudflare.com/learning/bots/what-is-data-scraping/

[13] Opeyemi, S. (2021, November 4). *What is social media scraping? (and why you should do it)*. Scraping Robot. Retrieved March 24, 2022, from https://scrapingrobot.com/blog/social-media-scraping/#:~:text=of%20contents%20above.-,Social%20Media%20Scraping%3A%20What%20Is%20It%3F,automatic%20process%20performed%20by%20bots.

[14] 6 real-world examples of Natural Language Processing. Expert.ai. (2022, February 10). Retrieved March 24, 2022, from https://www.expert.ai/blog/natural-language-processing-examples/

[15] Kotelnikova, A., Paschenko, D., Bochenina, K., & Kotelnikov, E. (2021, November 19). *Lexicon-based methods vs. Bert for text sentiment analysis*. arXiv.org. Retrieved March 24, 2022, from https://arxiv.org/abs/2111.10097

[16] Tong RM (2001) An operational system for detecting and tracking opinions in on-line discussions. In: Working Notes of the SIGIR Workshop on Operational Text Classification, pp 1–6

[17] Turney P, Littman M (2003) Measuring praise and criticism: inference of semantic orientation from association. ACM Transact Inform Syst J 21(4):315–346

[18] Sadia, A., Khan, F., & Bashir, F. (1970, January 1). *[PDF] an overview of lexicon-based approach for sentiment analysis: Semantic scholar*. [PDF] An Overview of Lexicon-Based Approach For Sentiment Analysis | Semantic Scholar. Retrieved March 24, 2022, from https://www.semanticscholar.org/paper/An-Overview-of-Lexicon-Based-Approach-For-Sentiment-Sadia-Khan/e53033c31e6ee88bad1cb3da4b122be60a53d4d5

[19] *SenticNet*. SenticNet API. (n.d.). Retrieved March 24, 2022, from https://sentic.net/api/

[20] Chung-Chi Chen, Hen-Hsen Huang and Hsin-Hsi Chen. 2018. NTUSD-Fin: A Market Sentiment Dictionary for Financial Social Media Data Applications. *In Proceedings of the 1st Financial Narrative Processing Workshop, 7 May 2018, Miyazaki, Japan.*

[21] Esuli, A., & Sebastiani, F. (n.d.). *SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining*. Retrieved March 24, 2022, from https://github.com/aesuli/SentiWordNet/blob/master/papers/LREC06.pdf

[22] Finn Årup Nielsen, "A new ANEW: evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages. Volume 718 in CEUR Workshop Proceedings: 93-98. 2011 May. Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, Mariann Hardey (editors)

[23] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[24] *Backpropagation*. Brilliant Math & Science Wiki. (n.d.). Retrieved March 24, 2022, from https://brilliant.org/wiki/backpropagation/#:~:text=Backpropagation%2C%20short%20for%20%22backward%20propagation,to%20the%20neural%20network's%20weights.

[25] By: IBM Cloud Education. (n.d.). *What are recurrent neural networks?* IBM. Retrieved March 24, 2022, from https://www.ibm.com/cloud/learn/recurrent-neural-networks

[26] Phi, M. (2020, June 28). *Illustrated guide to LSTM's and GRU's: A step by step explanation*. Medium. Retrieved March 24, 2022, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[27] *Attention is all you need - arxiv.org*. (n.d.). Retrieved March 24, 2022, from https://arxiv.org/pdf/1706.03762v5.pdf

[28] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (n.d.). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. ACL Anthology. Retrieved March 24, 2022, from https://aclanthology.org/N19-1423/

[29] *Fine-tuning pre-trained Bert Models*¶. Fine-tuning Pre-trained BERT Models - gluonnlp 0.10.0 documentation. (n.d.). Retrieved March 24, 2022, from https://nlp.gluon.ai/examples/sentence_embedding/bert.html

[30] Agarwal, R. (2022, March 12). *The 5 classification evaluation metrics every data scientist must know*. Medium. Retrieved March 24, 2022, from https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

[31] Stephanie. (2020, September 16). *Cohen's Kappa Statistic*. Statistics How To. Retrieved March 24, 2022, from https://www.statisticshowto.com/cohens-kappa-statistic/

[32] Labille, Kevin & Alfarhood, Sultan & Gauch, Susan. (2016). Estimating Sentiment via Probability and Information Theory. 121-129. 10.5220/0006072101210129.

[33] Labille, Kevin & Gauch, Susan & Alfarhood, Sultan. (2017). Creating Domain-Specific Sentiment Lexicons via Text Mining.

[34] Park, Jihye & Lee, Hye & Cho, Sungzoon. (2021). Automatic Construction of Context-Aware Sentiment Lexicon in the Financial Domain Using Direction-Dependent Words.

10.Appendix

Instances where SenticNet predicted wrongly and all lexicons voted against it

	senticnet	actual_class	text	processed_words
0	1	0	abg sundal collier tf bank ab tf bank loan loss level down bp qoq pbll in q slightly lower than abgsc and consensus loan loss level came in at bp down bp vs q share to outperform on loan loss relief equity stocks	loan loss lower consensus loan loss share outperform loan loss relief equity stocks
1	0	1	rt happy wednesday here are my topthingstoknowtoday in financial markets earnings in the morning	happy earnings
2	0	1	dow futures up points on easing of coronavirus lockdowns as earnings season hits stride us stockindex futures rose tuesday with analysts citing progress toward easing coronavirus lockdowns a	coronavirus earnings stride rose progress coronavirus
3	0	1	growing vegetables spine gourd farming is profitable agriculture busin via depression usa community can try thanks trump	growing spine profitable agriculture depression community try thanks
4	0	1	rt apple has bought back billion in stock over the past years which is greater than the market cap of companie	
5	0	1	rt apple has bought back billion in stock over the past years which is greater than the market cap of companie	
6	0	1	impressive	impressive
7	0	1	rt meet your staff these are some of the best traders in the hypify community and their job is to teach you	best community job teach
8	0	1	rt advac looks like best nearterm prospect no theoretical risk of contamination by active sarscov h	best prospect theoretical risk contamination
9	0	1	rt added to chart setup still looks good for continuation	chart setup good
10	1	0	rt extreme sentiment dislocation only bulls on stocks aaii survey below march level when mkt collapsed everyone must	extreme sentiment dislocation stocks survey
11	1	0	time for the dump	time dump
12	0	1	havent had a flat day in awhile good sign its stabilizing imo	flat awhile good sign
13	1	0	load up those puts its over	load
14	1	0	short all rips this will not see anymore	short see
15	1	0	another bull head fake	bull head fake
16	1	0	real inflation close to	real inflation close
17	1	0	poor bulls hike on the way t minus minutes and counting	poor hike counting
18	1	0	came down to market open open level in minutes lmaoooo	open open
19	1	0	rate hikes this year oil up today no end in sight on war market will close down today waaay over bought dead cat bounce	rate oil end sight war close dead cat bounce
20	1	0	red close	close
21	0	1	impressive volume big boys taking notice	impressive volume big
22	0	1	this market is going parabolic as the good news is aboi it t o start flowing in omg	parabolic good flowing omg
23	1	0	bull trap	bull trap
24	1	0	we should be shorting the sec cause they going down	cause
25	1	0	guaranteed to dump every time	dump time
26	0	1	ripple proposes regulatory clarity in south koreas crypto distinction donates in xrp to ukraine	ripple regulatory clarity distinction
27	1	0	lol some motherfucker trying to dump it	lol motherfucker dump
28	1	0	bull trap	bull trap
29	1	0	bear market sucks for bulls	bear
30	1	0	bull trap	bull trap
31	1	0	bull trap	bull trap

Figure 95: Instances where SenticNet predicted wrongly and all lexicons voted against it

Instances where Vader helped our Lexicon Ensemble to predict correctly instead of wrongly

	actual_class	leave2soft_w_vader	leave2soft_wo_vader	text	processed_words	vader
0	0	-0.032115	0.008214	rt several large corporations are stopping advertising on facebook either till july or for the whole of this year if these	[stopping]	-0.1531
1	0	-0.046601	0.005298	rt biggest mistakes of the century from before apple is only a pc maker amazon is only a bookstore goo	[mistakes, amazon]	-0.2023
2	0	-0.032115	0.008214	rt several large corporations are stopping advertising on facebook either till july or for the whole of this year if these	[stopping]	-0.1531
3	0	-0.028465	0.006491	crazy opening gaps	[crazy]	-0.3400
4	0	-0.004048	0.001849	unless you are an arrogant letter writeradvice seller not all are but some are it is time to realize that the obvious problems in this economy and the financial mkts provide an uncertain road w both risk and opportunity even if u are a pious shark	[arrogant, problems, uncertain, risk, opportunity]	-0.6486
5	1	0.005451	-0.013879	rt the on whats driving the shorting of stocks that has seen bets against the spdr sp trust the biggest exchange	[trust]	0.5106
6	0	-0.000514	0.020148	rt saturday country descends into chaos sunday chaos deepens monday rallies for no reason trump declares	[chaos, chaos, no]	-0.8625
7	0	-0.018432	0.009658	hope not i added some during the panic selloff	[hope, panic]	-0.1027
8	0	-0.005531	0.030720	rt be greedy when others are fearful and be fearful when others are greedy	[greedy, fearful, fearful, greedy]	-0.8750
9	1	0.002611	-0.046279	rt in made new all time highs on declining earnings growth in made new all time highs on declining	[growth]	0.3818
10	0	-0.004486	0.022590	corona blew a lead over racism esf	[racism]	-0.6249
11	0	-0.009385	0.038521	rt if you miss a stocks initial breakout from a cup with handle you should keep your eye on it in time it may form a	[miss]	-0.1531
12	0	-0.033655	0.002191	ya ya spx spx i can call numbers too crash this crash that truth is feds were able to get sp to without spending more than of their intended use just think what they can do if they really wanted to stop fighting follow fed price	[crash, crash, truth, stop, fighting]	-0.8171
13	0	-0.034298	0.006969	did you know that required minimum distributions from an ira or k account have been suspended for retirement savers wont be forced to make withdrawals from their plans in the midst of a bear market stocks markets stegent equity advisors inc	[suspended, forced]	-0.1581
14	0	-0.041898	0.016542	making a comeback and dumping up points in that now	[dumping]	-0.3182
15	0	-0.008210	0.000164	no one is going to upgrade	[no]	-0.2960
16	0	-0.000269	0.016841	rt oecd projects the european regions growth to be worst hit by the pandemic in while korea and china the original	[growth, worst, original]	-0.0516
17	0	-0.047838	0.011650	rt for most of us technical investment terms can be confusing investment investmentopportunities capitalinvestmen	[confusing]	-0.2263
18	1	0.014838	-0.000685	to be honest that unfilled gap is actually a double unfilled gap was a island bottom gap	[honest]	0.5106
19	0	-0.029082	0.012258	rt provided clarity today with the close below the day ma after failing to break above on tuesday its now likely	[clarity, failing]	-0.1531
20	0	-0.071596	0.018272	yields rising not good for inflated bubble assests	[good]	-0.3412
21	0	-0.017067	0.002644	chief pump strategist couldnt be reached for comment	[reached]	-0.0762
22	0	-0.005511	0.006697	once nothing is fixed by the third hike its gonna be a bigger basis point for the next and market gonna dump already planned in advance i bet	[dump]	-0.3818
23	1	0.007051	-0.001196	short squeeze candidate for vxx a heavily shorted instrument with about of its shares sold short the halt of share creation makes it a prime candidate for a potential short squeeze ihor dusaniwsky managing director of predictive analytics at s partners said vxx short sellers are down about million for the yeartodate according to s data vxx shorts are already teetering the lack of supply might really put them over the edge dusaniwsky said	[shares, share, creation, lack]	0.4939
24	0	-0.025252	0.032997	doom and gloom	[doom, gloom]	-0.7430
25	1	0.006181	-0.000359	only dumped cause of spy lol	[dumped, lol]	0.0258
26	1	0.014016	-0.015457	accumulate and be rich close to million shib prepared for burning today tokens destroyed within hours	[rich, prepared, destroyed]	0.3182
27	1	0.012164	-0.021748	is going to fly to dont miss out on x	[miss]	0.1139
28	1	0.015463	-0.019934	covered in video cryptos ready to fly you dont want to miss this pump	[ready, want, miss]	0.4063
29	1	0.018257	-0.009891	i got no money and im unemployed i liquidated all my credit cards and put everything into crypto lfg	[no, credit]	0.1027
30	1	0.015673	-0.047250	no fear	[no, fear]	0.3875
31	1	0.007321	-0.021113	that outage is reminiscent of the outages in this year its ok guys cardano has never had an outage since its inception we will carry the blockchain torch into the mainstream economy you can count on it	[ok]	0.2960
32	0	-0.060500	0.004737	the speculative honeymoon across all asset classes crypto trash otc scams and shitty nonprofitable companies on a bigleague exchange is over everyone can agree that everything good comes to an end unless youre religious and into that paradoxical voodoo place for the dead that the invisible man in the sky oversees	[speculative, asset, scams, shitty, agree, good, dead]	-0.6597
33	0	-0.016744	0.011008	china screwed us all	[screwed]	-0.4939

Figure 96: Instances where Vader helped our Lexicon Ensemble to predict correctly instead of wrongly

Lexicon Ensemble (Full Results)

Data 2

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.786286	0.84166	0.759939	0.943074
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.779429	0.837405	0.75303	0.943074
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.776	0.834179	0.752672	0.935484
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.817143	0.855335	0.816926	0.897533
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.781714	0.837447	0.759259	0.933586
ntusd_stocktwitlexi_afinn_vader_soft	0.780571	0.836457	0.758887	0.931689
ntusd_stocktwitlexi_vader_soft	0.774857	0.832909	0.753067	0.931689
ntusd_sentiwordnet_vader_soft	0.796571	0.844406	0.78282	0.916509
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.817143	0.855072	0.818024	0.895636
ntusd_sentiwordnet_afinn_vader_soft	0.796571	0.844406	0.78282	0.916509
senticnet_ntusd_stocktwitlexi_vader_sentidd_leave2soft	0.773714	0.836634	0.740146	0.962049
ntusd_stocktwitlexi_vader_sentidd_soft	0.819429	0.856624	0.82087	0.895636
ntusd_afinn_vader_soft	0.796571	0.844406	0.78282	0.916509
ntusd_sentiwordnet_vader_sentidd_soft	0.835429	0.865672	0.851376	0.880455
ntusd_vader_soft	0.793143	0.841921	0.779935	0.914611
ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.704	0.791968	0.68663	0.935484
sentiwordnet_stocktwitlexi_afinn_vader_soft	0.792	0.844974	0.766615	0.941176
ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.820571	0.857402	0.8223	0.895636
stocktwitlexi_vader_sentidd_soft	0.826286	0.862568	0.823834	0.905123

sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.828571	0.863884	0.827826	0.903226
stocktwitlexi_afinn_vader_soft	0.784	0.839966	0.75841	0.941176
stocktwitlexi_vader_soft	0.781714	0.838546	0.756098	0.941176
sentiwordnet_stocktwitlexi_vader_soft	0.788571	0.842821	0.763077	0.941176
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.828571	0.86413	0.82669	0.905123
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.747429	0.818107	0.722384	0.943074
ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.833143	0.86406	0.848263	0.880455
stocktwitlexi_afinn_vader_sentidd_soft	0.827429	0.863595	0.824138	0.907021
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_leave2soft	0.731429	0.809408	0.706799	0.946869
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.730286	0.808442	0.706383	0.944972
ntusd_afinn_vader_sentidd_soft	0.829714	0.860617	0.848708	0.872865
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_leave2soft	0.733714	0.813749	0.703039	0.965844
ntusd_vader_sentidd_soft	0.827429	0.858746	0.846863	0.870968
ntusd_stocktwitlexi_vader_hard	0.723429	0.804523	0.700422	0.944972
ntusd_sentiwordnet_stocktwitlexi_soft	0.688	0.78037	0.677374	0.920304
senticnet_ntusd_stocktwitlexi_afinn_sentidd_leave2soft	0.732571	0.810065	0.707801	0.946869
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.779429	0.827525	0.782095	0.878558
senticnet_ntusd_stocktwitlexi_afinn_vader_leave2soft	0.728	0.811111	0.697135	0.969639
ntusd_stocktwitlexi_afinn_soft	0.688	0.780016	0.677871	0.918406
senticnet_ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.798857	0.848537	0.776378	0.935484
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.797714	0.847807	0.775157	0.935484
senticnet_ntusd_stocktwitlexi_vader_sentidd_soft	0.796571	0.846816	0.774803	0.933586
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.8	0.849268	0.777603	0.935484

ntusd_stocktwitlexi_afinn_hard	0.710857	0.792793	0.697406	0.918406
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_leave2soft	0.696	0.792512	0.672848	0.963947
senticnet_ntusd_sentiwordnet_vader_sentidd_soft	0.808	0.852632	0.792822	0.922201
ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.773714	0.822581	0.779287	0.870968
senticnet_ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.808	0.852373	0.793781	0.920304
ntusd_stocktwitlexi_afinn_sentidd_soft	0.772571	0.821525	0.778912	0.86907
senticnet_ntusd_vader_sentidd_soft	0.802286	0.847845	0.790164	0.914611
senticnet_ntusd_afinn_vader_sentidd_soft	0.802286	0.847845	0.790164	0.914611
ntusd_stocktwitlexi_sentidd_soft	0.766857	0.816547	0.776068	0.86148
ntusd_sentiwordnet_stocktwitlexi_hard	0.693714	0.785256	0.679612	0.929791
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.755429	0.823432	0.728467	0.946869
ntusd_sentiwordnet_afinn_soft	0.707429	0.776614	0.718901	0.844402
senticnet_stocktwitlexi_afinn_vader_sentidd_soft	0.794286	0.84375	0.7776	0.922201
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.750857	0.820724	0.724238	0.946869
senticnet_ntusd_stocktwitlexi_vader_soft	0.747429	0.818704	0.721098	0.946869
senticnet_ntusd_sentiwordnet_vader_soft	0.761143	0.824517	0.739458	0.931689
ntusd_stocktwitlexi_soft	0.659429	0.760835	0.659249	0.899431
senticnet_stocktwitlexi_vader_sentidd_soft	0.794286	0.844021	0.776715	0.924099
ntusd_stocktwitlexi_sentidd_hard	0.699429	0.774636	0.70625	0.857685
senticnet_ntusd_sentiwordnet_afinn_vader_soft	0.765714	0.827296	0.743939	0.931689
senticnet_ntusd_stocktwitlexi_afinn_vader_soft	0.748571	0.819376	0.722142	0.946869
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.798857	0.847487	0.779904	0.927894
senticnet_ntusd_stocktwitlexi_hard	0.670857	0.777434	0.655802	0.954459

senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.763429	0.824129	0.746154	0.920304
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.796571	0.846021	0.777424	0.927894
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.766857	0.826235	0.749614	0.920304
senticnet_ntusd_stocktwitlexi_sentidd_soft	0.76	0.820513	0.746501	0.910816
sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.771429	0.823944	0.768473	0.888046
senticnet_ntusd_vader_soft	0.757714	0.821849	0.737557	0.927894
senticnet_ntusd_afinn_vader_soft	0.76	0.823529	0.739065	0.929791
stocktwitlexi_afinn_sentidd_soft	0.771429	0.826389	0.7616	0.903226
senticnet_ntusd_stocktwitlexi_afinn_sentidd_soft	0.762286	0.822526	0.747287	0.914611
ntusd_sentiwordnet_afinn_sentidd_soft	0.771429	0.812383	0.80334	0.821632
ntusd_sentiwordnet_soft	0.677714	0.752632	0.699837	0.814042
sentiwordnet_stocktwitlexi_afinn_soft	0.673143	0.77898	0.657106	0.956357
sentiwordnet_stocktwitlexi_sentidd_soft	0.766857	0.820738	0.764321	0.886148
senticnet_stocktwitlexi_afinn_vader_soft	0.746286	0.816225	0.723935	0.935484
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.752	0.820215	0.727941	0.939279
senticnet_sentiwordnet_stocktwitlexi_vader_soft	0.750857	0.819536	0.726872	0.939279
senticnet_stocktwitlexi_vader_soft	0.745143	0.81555	0.722874	0.935484
senticnet_ntusd_sentiwordnet_afinn_sentidd_soft	0.769143	0.824042	0.761675	0.897533
senticnet_ntusd_afinn_sentidd_soft	0.771429	0.824561	0.766721	0.891841
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.696	0.789223	0.677551	0.944972
senticnet_ntusd_sentiwordnet_sentidd_soft	0.766857	0.821678	0.76175	0.891841
senticnet_ntusd_sentiwordnet_stocktwitlexi_soft	0.689143	0.78515	0.67253	0.943074
ntusd_afinn_soft	0.685714	0.757282	0.707921	0.814042

ntusd_sentiwordnet_sentidd_soft	0.754286	0.796978	0.793233	0.800759
stocktwitlexi_sentidd_soft	0.758857	0.817316	0.751592	0.895636
stocktwitlexi_afinn_soft	0.649143	0.771407	0.634804	0.982922
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.774857	0.835696	0.745536	0.950664
senticnet_ntusd_stocktwitlexi_afinn_soft	0.686857	0.783912	0.670715	0.943074
senticnet_ntusd_sentidd_soft	0.764571	0.818981	0.762684	0.88425
senticnet_ntusd_stocktwitlexi_soft	0.681143	0.779447	0.668022	0.935484
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.763429	0.822622	0.75	0.910816
senticnet_sentiwordnet_stocktwitlexi_sentidd_soft	0.76	0.820513	0.746501	0.910816
senticnet_ntusd_sentiwordnet_afinn_soft	0.700571	0.78871	0.685835	0.927894
ntusd_afinn_sentidd_soft	0.76	0.801512	0.798493	0.804554
sentiwordnet_stocktwitlexi_soft	0.648	0.765601	0.639136	0.954459
senticnet_stocktwitlexi_afinn_sentidd_soft	0.756571	0.818104	0.743789	0.908918
senticnet_ntusd_sentiwordnet_vader_sentidd_leave2soft	0.769143	0.827055	0.75351	0.916509
ntusd_sentidd_soft	0.753143	0.795455	0.793951	0.796964
senticnet_ntusd_sentiwordnet_soft	0.690286	0.780922	0.680282	0.916509
senticnet_stocktwitlexi_sentidd_soft	0.749714	0.812339	0.740625	0.899431
senticnet_ntusd_afinn_soft	0.693714	0.78282	0.683168	0.916509
senticnet_ntusd_sentiwordnet_afinn_vader_leave2soft	0.745143	0.812447	0.729607	0.916509
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_hard	0.725714	0.801653	0.710102	0.920304
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_hard	0.733714	0.804037	0.722054	0.907021
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.730286	0.8112	0.701245	0.962049
senticnet_ntusd_sentiwordnet_afinn_sentidd_leave2soft	0.756571	0.816221	0.748418	0.897533

senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_hard	0.718857	0.800325	0.699291	0.935484
senticnet_sentiwordnet_stocktwitlexi_afinn_soft	0.685714	0.782264	0.671196	0.937381
senticnet_ntusd_soft	0.685714	0.777328	0.677966	0.910816
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.731429	0.809717	0.706215	0.948767
senticnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.778286	0.836975	0.751131	0.944972
senticnet_stocktwitlexi_afinn_soft	0.674286	0.775414	0.663073	0.933586
senticnet_ntusd_stocktwitlexi_vader_sentidd_hard	0.752	0.813734	0.742947	0.899431
senticnet_sentiwordnet_stocktwitlexi_soft	0.676571	0.77593	0.665761	0.929791
senticnet_ntusd_afinn_vader_sentidd_leave2soft	0.777143	0.830876	0.765176	0.908918
senticnet_sentiwordnet_vader_sentidd_soft	0.793143	0.839397	0.788333	0.897533
senticnet_sentiwordnet_afinn_vader_sentidd_soft	0.792	0.838652	0.787022	0.897533
senticnet_ntusd_sentiwordnet_hard	0.715429	0.79058	0.70997	0.891841
senticnet_stocktwitlexi_soft	0.664	0.768504	0.656797	0.925996
senticnet_ntusd_stocktwitlexi_afinn_vader_hard	0.742857	0.808836	0.732308	0.903226
senticnet_ntusd_stocktwitlexi_afinn_sentidd_hard	0.749714	0.807726	0.751634	0.872865
senticnet_sentiwordnet_afinn_vader_soft	0.756571	0.818723	0.742284	0.912713
senticnet_sentiwordnet_vader_soft	0.753143	0.816327	0.739599	0.910816
senticnet_vader_sentidd_soft	0.792	0.83779	0.789916	0.891841
senticnet_afinn_vader_sentidd_soft	0.790857	0.836753	0.789562	0.889943
senticnet_sentiwordnet_afinn_sentidd_soft	0.770286	0.822595	0.768977	0.88425
senticnet_ntusd_vader_hard	0.730286	0.798291	0.726283	0.886148
senticnet_sentiwordnet_stocktwitlexi_hard	0.672	0.775956	0.659151	0.943074
senticnet_sentiwordnet_sentidd_soft	0.761143	0.815534	0.762376	0.87666

senticnet_afinn_vader_soft	0.752	0.815633	0.738462	0.910816
senticnet_vader_soft	0.748571	0.813243	0.735791	0.908918
senticnet_ntusd_afinn_hard	0.728	0.790861	0.736498	0.85389
ntusd_sentiwordnet_afinn_vader_sentidd_leave2soft	0.789714	0.831193	0.804618	0.859583
senticnet_stocktwitlexi_afinn_hard	0.706286	0.791227	0.691761	0.924099
senticnet_stocktwitlexi_vader_hard	0.698286	0.789809	0.680384	0.941176
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.765714	0.821273	0.759677	0.893738
senticnet_stocktwitlexi_sentidd_hard	0.698286	0.781457	0.693098	0.895636
senticnet_sentiwordnet_afinn_soft	0.694857	0.780608	0.688406	0.901328
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.768	0.816621	0.77931	0.857685
senticnet_sentiwordnet_soft	0.683429	0.773137	0.680115	0.895636
senticnet_afinn_sentidd_soft	0.756571	0.811337	0.760797	0.86907
senticnet_ntusd_sentidd_hard	0.724571	0.776645	0.759058	0.795066
senticnet_sentidd_soft	0.749714	0.804987	0.758389	0.857685
sentiwordnet_vader_sentidd_soft	0.811429	0.841499	0.85214	0.83112
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.746286	0.802139	0.756303	0.85389
sentiwordnet_afinn_vader_sentidd_soft	0.809143	0.840191	0.84749	0.833017
sentiwordnet_afinn_vader_soft	0.772571	0.818926	0.786713	0.85389
sentiwordnet_vader_soft	0.766857	0.814545	0.78185	0.850095
ntusd_sentiwordnet_vader_hard	0.745143	0.796347	0.767606	0.827324
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.752	0.798888	0.780797	0.817837
senticnet_afinn_soft	0.681143	0.769992	0.680758	0.886148
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.766857	0.813528	0.784832	0.844402

senticnet_sentiwordnet_afinn_vader_sentidd_leave2soft	0.769143	0.81932	0.774958	0.86907
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.785143	0.818182	0.83432	0.802657
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.746286	0.803887	0.752066	0.863378
senticnet_ntusd_sentiwordnet_vader_sentidd_hard	0.769143	0.811567	0.798165	0.825427
sentiwordnet_stocktwitlexi_vader_hard	0.699429	0.775406	0.704969	0.86148
ntusd_sentiwordnet_afinn_hard	0.726857	0.773888	0.771698	0.776091
sentiwordnet_stocktwitlexi_afinn_hard	0.707429	0.775044	0.721768	0.836812
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.733714	0.794351	0.742574	0.85389
senticnet_ntusd_sentiwordnet_afinn_vader_hard	0.745143	0.795225	0.770463	0.821632
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.745143	0.796718	0.766667	0.829222
senticnet_ntusd_sentiwordnet_afinn_sentidd_hard	0.761143	0.8	0.80695	0.793169
senticnet_sentiwordnet_vader_hard	0.723429	0.783929	0.740304	0.833017
sentiwordnet_afinn_sentidd_soft	0.758857	0.797699	0.806202	0.789374
sentiwordnet_afinn_soft	0.699429	0.759817	0.732394	0.789374
senticnet_sentiwordnet_afinn_hard	0.716571	0.772477	0.74778	0.798861
ntusd_stocktwitlexi_afinn_vader_sentidd_hard	0.784	0.81561	0.839357	0.793169
senticnet_stocktwitlexi_afinn_vader_sentidd_hard	0.782857	0.817308	0.82846	0.806452
stocktwitlexi_vader_sentidd_hard	0.782857	0.818008	0.825919	0.810247
sentiwordnet_stocktwitlexi_sentidd_hard	0.694857	0.752089	0.736364	0.768501
stocktwitlexi_afinn_vader_hard	0.757714	0.796545	0.805825	0.787476
afinn_vader_sentidd_soft	0.795429	0.820821	0.868644	0.777989
senticnet_ntusd_afinn_vader_sentidd_hard	0.776	0.806324	0.841237	0.774194
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.771429	0.800797	0.842767	0.762808

afinn_vader_soft	0.756571	0.793804	0.810277	0.777989
vader_sentidd_soft	0.793143	0.81809	0.869658	0.772296
sentiwordnet_sentidd_soft	0.732571	0.766932	0.807128	0.73055
senticnet_afinn_vader_hard	0.756571	0.791381	0.817814	0.766603
ntusd_afinn_vader_hard	0.754286	0.786918	0.823651	0.753321
senticnet_vader_sentidd_hard	0.769143	0.801961	0.829615	0.776091
ntusd_sentiwordnet_sentidd_hard	0.693714	0.723711	0.792325	0.666034
ntusd_sentiwordnet_afinn_vader_sentidd_hard	0.769143	0.792181	0.865169	0.73055
ntusd_vader_sentidd_hard	0.758857	0.780437	0.864055	0.711575
senticnet_sentiwordnet_afinn_vader_sentidd_hard	0.763429	0.792793	0.838983	0.751423
senticnet_sentiwordnet_sentidd_hard	0.707429	0.74902	0.774848	0.724858
sentiwordnet_afinn_vader_hard	0.752	0.783217	0.827004	0.743833
stocktwitlexi_afinn_sentidd_hard	0.749714	0.775385	0.84375	0.717268
afinn_sentidd_soft	0.749714	0.768743	0.866667	0.690702
senticnet_afinn_sentidd_hard	0.734857	0.760825	0.832957	0.70019
ntusd_afinn_sentidd_hard	0.731429	0.739756	0.888298	0.633776
afinn_vader_sentidd_hard	0.729143	0.749206	0.84689	0.671727
sentiwordnet_vader_sentidd_hard	0.746286	0.762313	0.874693	0.675522
sentiwordnet_afinn_sentidd_hard	0.714286	0.719101	0.881543	0.607211

Figure 97: Lexicon Ensemble Test Results for Data 2

Data 3

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.726098	0.773504	0.747934	0.800885
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.72093	0.774059	0.734127	0.818584
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.726098	0.774468	0.745902	0.80531
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.70801	0.751284	0.747445	0.755162
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.718346	0.767921	0.740082	0.797935
ntusd_stocktwitlexi_afinn_vader_soft	0.718346	0.768249	0.739427	0.79941
ntusd_stocktwitlexi_vader_soft	0.721792	0.771408	0.741497	0.803835
ntusd_sentiwordnet_vader_soft	0.718346	0.758315	0.76	0.756637
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.703704	0.747059	0.744868	0.749263
ntusd_sentiwordnet_afinn_vader_soft	0.715762	0.756637	0.756637	0.756637
senticnet_ntusd_stocktwitlexi_vader_sentidd_leave2soft	0.701981	0.763661	0.711196	0.824484
ntusd_stocktwitlexi_vader_sentidd_soft	0.698536	0.742647	0.740469	0.744838
ntusd_afinn_vader_soft	0.71404	0.754438	0.756677	0.752212
ntusd_sentiwordnet_vader_sentidd_soft	0.698536	0.732824	0.759494	0.707965
ntusd_vader_soft	0.716624	0.756477	0.759287	0.753687
ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.755383	0.806276	0.75	0.871681
sentiwordnet_stocktwitlexi_afinn_vader_soft	0.686477	0.752717	0.697733	0.817109
ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.69509	0.739323	0.738235	0.740413
stocktwitlexi_vader_sentidd_soft	0.670973	0.733612	0.695767	0.775811
sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.670973	0.731364	0.698925	0.766962

stocktwitlexi_afinn_vader_soft	0.686477	0.755047	0.694307	0.827434
stocktwitlexi_vader_soft	0.687339	0.756212	0.694205	0.830383
sentiwordnet_stocktwitlexi_vader_soft	0.685616	0.751869	0.697352	0.815634
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.670112	0.730471	0.69852	0.765487
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.716624	0.775733	0.721166	0.839233
ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.694229	0.729627	0.754331	0.70649
stocktwitlexi_afinn_vader_sentidd_soft	0.666667	0.72956	0.693227	0.769912
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_leave2soft	0.716624	0.78341	0.707491	0.877581
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.724376	0.782313	0.72601	0.848083
ntusd_afinn_vader_sentidd_soft	0.693368	0.727412	0.756369	0.70059
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_leave2soft	0.705426	0.774108	0.700957	0.864307
ntusd_vader_sentidd_soft	0.69509	0.728943	0.757962	0.702065
ntusd_stocktwitlexi_vader_hard	0.728682	0.782308	0.736021	0.834808
ntusd_sentiwordnet_stocktwitlexi_soft	0.753661	0.805442	0.747475	0.873156
senticnet_ntusd_stocktwitlexi_afinn_sentidd_leave2soft	0.706288	0.77462	0.701796	0.864307
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.710594	0.756168	0.744286	0.768437
senticnet_ntusd_stocktwitlexi_afinn_vader_leave2soft	0.704565	0.771486	0.703524	0.853982
ntusd_stocktwitlexi_afinn_soft	0.749354	0.801906	0.744627	0.868732
senticnet_ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.671835	0.732256	0.699329	0.768437
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.669251	0.732218	0.694444	0.774336
senticnet_ntusd_stocktwitlexi_vader_sentidd_soft	0.670973	0.732493	0.697333	0.771386
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.666667	0.729937	0.692715	0.771386
ntusd_stocktwitlexi_afinn_hard	0.735573	0.78636	0.744401	0.833333

senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_leave2soft	0.71404	0.78553	0.698851	0.896755
senticnet_ntusd_sentiwordnet_vader_sentidd_soft	0.668389	0.725196	0.702628	0.749263
ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.708872	0.754717	0.742857	0.766962
senticnet_ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.666667	0.722978	0.702364	0.744838
ntusd_stocktwitlexi_afinn_sentidd_soft	0.704565	0.750908	0.739628	0.762537
senticnet_ntusd_vader_sentidd_soft	0.669251	0.724138	0.705882	0.743363
senticnet_ntusd_afinn_vader_sentidd_soft	0.669251	0.724138	0.705882	0.743363
ntusd_stocktwitlexi_sentidd_soft	0.70112	0.748368	0.736091	0.761062
ntusd_sentiwordnet_stocktwitlexi_hard	0.713178	0.779616	0.707083	0.868732
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.672696	0.740791	0.689086	0.800885
ntusd_sentiwordnet_afinn_soft	0.751938	0.786982	0.789318	0.784661
senticnet_stocktwitlexi_afinn_vader_sentidd_soft	0.647717	0.719286	0.672657	0.772861
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.672696	0.741497	0.688131	0.803835
senticnet_ntusd_stocktwitlexi_vader_soft	0.676141	0.743169	0.692112	0.80236
senticnet_ntusd_sentiwordnet_vader_soft	0.67528	0.737282	0.698811	0.780236
ntusd_stocktwitlexi_soft	0.74677	0.800813	0.740602	0.871681
senticnet_stocktwitlexi_vader_sentidd_soft	0.644272	0.717317	0.669221	0.772861
ntusd_stocktwitlexi_sentidd_hard	0.749354	0.7865	0.782482	0.79056
senticnet_ntusd_sentiwordnet_afinn_vader_soft	0.670973	0.733612	0.695767	0.775811
senticnet_ntusd_stocktwitlexi_afinn_vader_soft	0.67528	0.741249	0.693196	0.79646
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.640827	0.713008	0.668387	0.764012
senticnet_ntusd_stocktwitlexi_hard	0.704565	0.782498	0.686318	0.910029
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.670112	0.73568	0.69131	0.786136

senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.639966	0.713699	0.66624	0.768437
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.667528	0.733425	0.68961	0.783186
senticnet_ntusd_stocktwitlexi_sentidd_soft	0.674419	0.738227	0.695822	0.786136
sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.660637	0.734501	0.676179	0.803835
senticnet_ntusd_vader_soft	0.674419	0.736034	0.698939	0.777286
senticnet_ntusd_afinn_vader_soft	0.672696	0.734266	0.698138	0.774336
stocktwitlexi_afinn_sentidd_soft	0.64944	0.730998	0.662275	0.815634
senticnet_ntusd_stocktwitlexi_afinn_sentidd_soft	0.670112	0.734581	0.69281	0.781711
ntusd_sentiwordnet_afinn_sentidd_soft	0.707149	0.739264	0.769968	0.710914
ntusd_sentiwordnet_soft	0.766581	0.798812	0.804185	0.79351
sentiwordnet_stocktwitlexi_afinn_soft	0.679587	0.771218	0.661392	0.924779
sentiwordnet_stocktwitlexi_sentidd_soft	0.64944	0.728123	0.665446	0.803835
senticnet_stocktwitlexi_afinn_vader_soft	0.655469	0.73262	0.669927	0.80826
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.64944	0.727759	0.665851	0.80236
senticnet_sentiwordnet_stocktwitlexi_vader_soft	0.646856	0.727394	0.662228	0.806785
senticnet_stocktwitlexi_vader_soft	0.652024	0.730667	0.666667	0.80826
senticnet_ntusd_sentiwordnet_afinn_sentidd_soft	0.661499	0.720285	0.696011	0.746313
senticnet_ntusd_afinn_sentidd_soft	0.661499	0.719486	0.697095	0.743363
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.676141	0.752957	0.67891	0.845133
senticnet_ntusd_sentiwordnet_sentidd_soft	0.659776	0.720453	0.692517	0.750737
senticnet_ntusd_sentiwordnet_stocktwitlexi_soft	0.680448	0.756402	0.681657	0.849558
ntusd_afinn_soft	0.751077	0.784167	0.794251	0.774336
ntusd_sentiwordnet_sentidd_soft	0.71404	0.743827	0.779935	0.710914

stocktwitlexi_sentidd_soft	0.632214	0.722907	0.645423	0.821534
stocktwitlexi_afinn_soft	0.664944	0.768314	0.644356	0.951327
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.631352	0.702778	0.664042	0.746313
senticnet_ntusd_stocktwitlexi_afinn_soft	0.677003	0.754098	0.678867	0.848083
senticnet_ntusd_sentidd_soft	0.658053	0.718639	0.691678	0.747788
senticnet_ntusd_stocktwitlexi_soft	0.680448	0.756721	0.681228	0.851032
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.634798	0.713514	0.658354	0.778761
senticnet_sentiwordnet_stocktwitlexi_sentidd_soft	0.633936	0.71419	0.656366	0.783186
senticnet_ntusd_sentiwordnet_afinn_soft	0.676141	0.745602	0.68875	0.812684
ntusd_afinn_sentidd_soft	0.702842	0.732765	0.771615	0.69764
sentiwordnet_stocktwitlexi_soft	0.669251	0.76699	0.651546	0.932153
senticnet_stocktwitlexi_afinn_sentidd_soft	0.632214	0.713615	0.654367	0.784661
senticnet_ntusd_sentiwordnet_vader_sentidd_leave2soft	0.658053	0.704393	0.711278	0.69764
ntusd_sentidd_soft	0.708872	0.735524	0.783333	0.693215
senticnet_ntusd_sentiwordnet_soft	0.672696	0.744624	0.683951	0.817109
senticnet_stocktwitlexi_sentidd_soft	0.630491	0.713043	0.652387	0.786136
senticnet_ntusd_afinn_soft	0.672696	0.742547	0.686717	0.80826
senticnet_ntusd_sentiwordnet_afinn_vader_leave2soft	0.660637	0.711144	0.706997	0.715339
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_hard	0.664083	0.721826	0.698895	0.746313
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_hard	0.666667	0.718954	0.708155	0.730088
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.632214	0.711291	0.656679	0.775811
senticnet_ntusd_sentiwordnet_afinn_sentidd_leave2soft	0.653747	0.703976	0.702941	0.705015
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_hard	0.657192	0.718927	0.689702	0.750737

senticnet_sentiwordnet_stocktwitlexi_afinn_soft	0.64255	0.736508	0.6466	0.855457
senticnet_ntusd_soft	0.668389	0.741089	0.681088	0.812684
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.625323	0.707465	0.650185	0.775811
senticnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.611542	0.679915	0.655267	0.70649
senticnet_stocktwitlexi_afinn_soft	0.644272	0.739103	0.646409	0.862832
senticnet_ntusd_stocktwitlexi_vader_sentidd_hard	0.66236	0.699387	0.728435	0.672566
senticnet_sentiwordnet_stocktwitlexi_soft	0.639966	0.735777	0.643805	0.858407
senticnet_ntusd_afinn_vader_sentidd_leave2soft	0.644272	0.680588	0.715447	0.648968
senticnet_sentiwordnet_vader_sentidd_soft	0.613264	0.672023	0.665702	0.678466
senticnet_sentiwordnet_afinn_vader_sentidd_soft	0.611542	0.67008	0.664731	0.675516
senticnet_ntusd_sentiwordnet_hard	0.666667	0.717724	0.709957	0.725664
senticnet_stocktwitlexi_soft	0.638243	0.735516	0.641758	0.861357
senticnet_ntusd_stocktwitlexi_afinn_vader_hard	0.650301	0.694737	0.708589	0.681416
senticnet_ntusd_stocktwitlexi_afinn_sentidd_hard	0.659776	0.694981	0.729335	0.663717
senticnet_sentiwordnet_afinn_vader_soft	0.614126	0.680912	0.658402	0.705015
senticnet_sentiwordnet_vader_soft	0.614987	0.681851	0.658872	0.70649
senticnet_vader_sentidd_soft	0.604651	0.659243	0.663677	0.654867
senticnet_afinn_vader_sentidd_soft	0.604651	0.658228	0.664662	0.651917
senticnet_sentiwordnet_afinn_sentidd_soft	0.602929	0.666667	0.653901	0.679941
senticnet_ntusd_vader_hard	0.652024	0.690184	0.71885	0.663717
senticnet_sentiwordnet_stocktwitlexi_hard	0.618432	0.710647	0.637749	0.80236
senticnet_sentiwordnet_sentidd_soft	0.604651	0.669546	0.654008	0.685841
senticnet_afinn_vader_soft	0.604651	0.66715	0.656205	0.678466

senticnet_vader_soft	0.606374	0.66908	0.657183	0.681416
senticnet_ntusd_afinn_hard	0.652885	0.687839	0.724307	0.654867
ntusd_sentiwordnet_afinn_vader_sentidd_leave2soft	0.628768	0.645267	0.729981	0.578171
senticnet_stocktwitlexi_afinn_hard	0.601206	0.682659	0.637644	0.734513
senticnet_stocktwitlexi_vader_hard	0.598622	0.683424	0.633501	0.741888
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.605512	0.651446	0.672956	0.631268
senticnet_stocktwitlexi_sentidd_hard	0.614987	0.688502	0.652576	0.728614
senticnet_sentiwordnet_afinn_soft	0.608096	0.688569	0.642401	0.741888
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.633936	0.651354	0.733826	0.585546
senticnet_sentiwordnet_soft	0.607235	0.689373	0.640506	0.746313
senticnet_afinn_sentidd_soft	0.586563	0.649123	0.643478	0.654867
senticnet_ntusd_sentidd_hard	0.670112	0.683209	0.777778	0.609145
senticnet_sentidd_soft	0.589147	0.652586	0.644604	0.660767
sentiwordnet_vader_sentidd_soft	0.593454	0.613748	0.689338	0.553097
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.624462	0.652313	0.710069	0.603245
sentiwordnet_afinn_vader_sentidd_soft	0.591731	0.613377	0.686131	0.554572
sentiwordnet_afinn_vader_soft	0.601206	0.633413	0.683761	0.589971
sentiwordnet_vader_soft	0.604651	0.635425	0.688468	0.589971
ntusd_sentiwordnet_vader_hard	0.637382	0.648874	0.746641	0.573746
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.62963	0.645215	0.73221	0.576696
senticnet_afinn_soft	0.586563	0.66759	0.629243	0.710914
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.608958	0.622924	0.712928	0.553097
senticnet_sentiwordnet_afinn_vader_sentidd_leave2soft	0.579673	0.612083	0.663793	0.567847

senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.622739	0.613074	0.764317	0.511799
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.587425	0.623723	0.667227	0.585546
senticnet_ntusd_sentiwordnet_vader_sentidd_hard	0.614126	0.613793	0.738589	0.525074
sentiwordnet_stocktwitlexi_vader_hard	0.589147	0.646405	0.649776	0.643068
ntusd_sentiwordnet_afinn_hard	0.633936	0.644351	0.744681	0.567847
sentiwordnet_stocktwitlexi_afinn_hard	0.589147	0.642697	0.652968	0.632743
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.580534	0.621601	0.656814	0.589971
senticnet_ntusd_sentiwordnet_afinn_vader_hard	0.60379	0.61794	0.707224	0.548673
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.583118	0.6128	0.66958	0.564897
senticnet_ntusd_sentiwordnet_afinn_sentidd_hard	0.605512	0.603806	0.730126	0.514749
senticnet_sentiwordnet_vader_hard	0.583118	0.618898	0.663851	0.579646
sentiwordnet_afinn_sentidd_soft	0.576227	0.600649	0.66787	0.545723
sentiwordnet_afinn_soft	0.585702	0.624512	0.66335	0.589971
senticnet_sentiwordnet_afinn_hard	0.575366	0.604651	0.662566	0.556047
ntusd_stocktwitlexi_afinn_vader_sentidd_hard	0.576227	0.561497	0.709459	0.464602
senticnet_stocktwitlexi_afinn_vader_sentidd_hard	0.561585	0.556234	0.680171	0.470501
stocktwitlexi_vader_sentidd_hard	0.563307	0.55409	0.686275	0.464602
sentiwordnet_stocktwitlexi_sentidd_hard	0.569337	0.606299	0.650338	0.567847
stocktwitlexi_afinn_vader_hard	0.560724	0.560345	0.674274	0.479351
afinn_vader_sentidd_soft	0.551249	0.531896	0.68046	0.436578
senticnet_ntusd_afinn_vader_sentidd_hard	0.569337	0.544627	0.711905	0.441003
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.559862	0.542525	0.690205	0.446903
afinn_vader_soft	0.554694	0.544493	0.676149	0.455752

vader_sentidd_soft	0.54522	0.519126	0.678571	0.420354
sentiwordnet_sentidd_soft	0.552972	0.570008	0.650284	0.507375
senticnet_afinn_vader_hard	0.551249	0.544978	0.668094	0.460177
ntusd_afinn_vader_hard	0.567614	0.547748	0.703704	0.448378
senticnet_vader_sentidd_hard	0.553833	0.529946	0.688679	0.430678
ntusd_sentiwordnet_sentidd_hard	0.626184	0.604736	0.790476	0.489676
ntusd_sentiwordnet_afinn_vader_sentidd_hard	0.570198	0.533209	0.7289	0.420354
ntusd_vader_sentidd_hard	0.594315	0.544046	0.791549	0.414454
senticnet_sentiwordnet_afinn_vader_sentidd_hard	0.553833	0.529946	0.688679	0.430678
senticnet_sentiwordnet_sentidd_hard	0.560724	0.567797	0.667331	0.4941
sentiwordnet_afinn_vader_hard	0.548665	0.528777	0.677419	0.433628
stocktwitlexi_afinn_sentidd_hard	0.552972	0.530317	0.686183	0.432153
afinn_sentidd_soft	0.540913	0.50966	0.677262	0.408555
senticnet_afinn_sentidd_hard	0.547804	0.513438	0.690773	0.408555
ntusd_afinn_sentidd_hard	0.571921	0.509378	0.770149	0.380531
afinn_vader_sentidd_hard	0.538329	0.491461	0.68883	0.382006
sentiwordnet_vader_sentidd_hard	0.534884	0.476744	0.694915	0.362832
sentiwordnet_afinn_sentidd_hard	0.531438	0.465619	0.697059	0.349558

Figure 98: Lexicon Ensemble Test Results for Data 3

Average of Data 2 and 3

Experiment	Accuracy	F1_score	Precision	Recall
ntusd_stocktwitlexi_afinn_vader_sentidd_leave2soft	**0.756192**	**0.807582**	**0.753936**	**0.871979**
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.750179	0.805732	0.743579	0.880829
ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.751049	0.804324	0.749287	0.870397
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.762577	0.803309	0.782185	0.826348
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.75003	0.802684	0.749671	0.865761
ntusd_stocktwitlexi_afinn_vader_soft	0.749459	0.802353	0.749157	0.865549
ntusd_stocktwitlexi_vader_soft	0.748324	0.802159	0.747282	0.867762
ntusd_sentiwordnet_vader_soft	0.757459	0.80136	0.77141	0.836573
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.760423	0.801066	0.781446	0.822449
ntusd_sentiwordnet_afinn_vader_soft	0.756167	0.800521	0.769729	0.836573
senticnet_ntusd_stocktwitlexi_vader_sentidd_leave2soft	0.737848	0.800147	0.725671	0.893267
ntusd_stocktwitlexi_vader_sentidd_soft	0.758982	0.799636	0.780669	0.820237
ntusd_afinn_vader_soft	0.755306	0.799422	0.769748	0.83436
ntusd_sentiwordnet_vader_sentidd_soft	0.766982	0.799248	0.805435	0.79421
ntusd_vader_soft	0.754883	0.799199	0.769611	0.834149
ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.729692	0.799122	0.718315	0.903583
sentiwordnet_stocktwitlexi_afinn_vader_soft	0.739239	0.798846	0.732174	0.879143
ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.757831	0.798362	0.780267	0.818024
stocktwitlexi_vader_sentidd_soft	0.74863	0.79809	0.759801	0.840467
sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.749772	0.797624	0.763375	0.835094

stocktwitlexi_afinn_vader_soft	0.735239	0.797507	0.726358	0.884305
stocktwitlexi_vader_soft	0.734526	0.797379	0.725151	0.88578
sentiwordnet_stocktwitlexi_vader_soft	0.737094	0.797345	0.730214	0.878405
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.749342	0.797301	0.762605	0.835305
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.732026	0.79692	0.721775	0.891154
ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.763686	0.796843	0.801297	0.793473
stocktwitlexi_afinn_vader_sentidd_soft	0.747048	0.796578	0.758683	0.838466
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_leave2soft	0.724026	0.796409	0.707145	0.912225
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.727331	0.795377	0.716197	0.896527
ntusd_afinn_vader_sentidd_soft	0.761541	0.794015	0.802539	0.786728
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_leave2soft	0.71957	0.793929	0.701998	0.915076
ntusd_vader_sentidd_soft	0.76126	0.793845	0.802413	0.786516
ntusd_stocktwitlexi_vader_hard	0.726055	0.793416	0.718221	0.88989
ntusd_sentiwordnet_stocktwitlexi_soft	0.72083	0.792906	0.712425	0.89673
senticnet_ntusd_stocktwitlexi_afinn_sentidd_leave2soft	0.71943	0.792342	0.704799	0.905588
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.745011	0.791846	0.76319	0.823497
senticnet_ntusd_stocktwitlexi_afinn_vader_leave2soft	0.716283	0.791298	0.700329	0.911811
ntusd_stocktwitlexi_afinn_soft	0.718677	0.790961	0.711249	0.893569
senticnet_ntusd_stocktwitlexi_afinn_vader_sentidd_soft	0.735346	0.790396	0.737853	0.85196
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.733482	0.790012	0.734801	0.85491
senticnet_ntusd_stocktwitlexi_vader_sentidd_soft	0.733772	0.789654	0.736068	0.852486
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.733333	0.789603	0.735159	0.853435
ntusd_stocktwitlexi_afinn_hard	0.723215	0.789577	0.720903	0.87587

senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_leave2soft	0.70502	0.789021	0.685849	0.930351
senticnet_ntusd_sentiwordnet_vader_sentidd_soft	0.738195	0.788914	0.747725	0.835732
ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.741293	0.788649	0.761072	0.818965
senticnet_ntusd_sentiwordnet_afinn_vader_sentidd_soft	0.737333	0.787675	0.748073	0.832571
ntusd_stocktwitlexi_afinn_sentidd_soft	0.738568	0.786216	0.75927	0.815804
senticnet_ntusd_vader_sentidd_soft	0.735768	0.785992	0.748023	0.828987
senticnet_ntusd_afinn_vader_sentidd_soft	0.735768	0.785992	0.748023	0.828987
ntusd_stocktwitlexi_sentidd_soft	0.733988	0.782458	0.75608	0.811271
ntusd_sentiwordnet_stocktwitlexi_hard	0.703446	0.782436	0.693347	0.899261
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.714062	0.782112	0.708777	0.873877
ntusd_sentiwordnet_afinn_soft	0.729683	0.781798	0.754109	0.814532
senticnet_stocktwitlexi_afinn_vader_sentidd_soft	0.721002	0.781518	0.725129	0.847531
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_soft	0.711777	0.78111	0.706185	0.875352
senticnet_ntusd_stocktwitlexi_vader_soft	0.711785	0.780937	0.706605	0.874614
senticnet_ntusd_sentiwordnet_vader_soft	0.718211	0.7809	0.719134	0.855962
ntusd_stocktwitlexi_soft	0.703099	0.780824	0.699925	0.885556
senticnet_stocktwitlexi_vader_sentidd_soft	0.719279	0.780669	0.722968	0.84848
ntusd_stocktwitlexi_sentidd_hard	0.724391	0.780568	0.744366	0.824123
senticnet_ntusd_sentiwordnet_afinn_vader_soft	0.718344	0.780454	0.719853	0.85375
senticnet_ntusd_stocktwitlexi_afinn_vader_soft	0.711926	0.780313	0.707669	0.871665
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_soft	0.719842	0.780247	0.724146	0.845953
senticnet_ntusd_stocktwitlexi_hard	0.687711	0.779966	0.67106	0.932244
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_soft	0.71677	0.779904	0.718732	0.85322

senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_soft	0.718268	0.77986	0.721832	0.848165
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.717193	0.77983	0.719612	0.851745
senticnet_ntusd_stocktwitlexi_sentidd_soft	0.717209	0.77937	0.721162	0.848476
sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.716033	0.779223	0.722326	0.84594
senticnet_ntusd_vader_soft	0.716066	0.778941	0.718248	0.85259
senticnet_ntusd_afinn_vader_soft	0.716348	0.778898	0.718602	0.852064
stocktwitlexi_afinn_sentidd_soft	0.710434	0.778693	0.711938	0.85943
senticnet_ntusd_stocktwitlexi_afinn_sentidd_soft	0.716199	0.778553	0.720049	0.848161
ntusd_sentiwordnet_afinn_sentidd_soft	0.739289	0.775823	0.786654	0.766273
ntusd_sentiwordnet_soft	0.722147	0.775722	0.752011	0.803776
sentiwordnet_stocktwitlexi_afinn_soft	0.676365	0.775099	0.659249	0.940568
sentiwordnet_stocktwitlexi_sentidd_soft	0.708149	0.774431	0.714883	0.844991
senticnet_stocktwitlexi_afinn_vader_soft	0.700878	0.774423	0.696931	0.871872
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_soft	0.70072	0.773987	0.696896	0.870819
senticnet_sentiwordnet_stocktwitlexi_vader_soft	0.698857	0.773465	0.69455	0.873032
senticnet_stocktwitlexi_vader_soft	0.698583	0.773108	0.69477	0.871872
senticnet_ntusd_sentiwordnet_afinn_sentidd_soft	0.715321	0.772163	0.728843	0.821923
senticnet_ntusd_afinn_sentidd_soft	0.716464	0.772024	0.731908	0.817602
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_soft	0.686071	0.77109	0.67823	0.895052
senticnet_ntusd_sentiwordnet_sentidd_soft	0.713317	0.771066	0.727134	0.821289
senticnet_ntusd_sentiwordnet_stocktwitlexi_soft	0.684795	0.770776	0.677094	0.896316
ntusd_afinn_soft	0.718395	0.770724	0.751086	0.794189
ntusd_sentiwordnet_sentidd_soft	0.734163	0.770403	0.786584	0.755837

stocktwitlexi_sentidd_soft	0.695535	0.770112	0.698508	0.858585
stocktwitlexi_afinn_soft	0.657043	0.769861	0.63958	0.967125
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_leave2soft	0.703105	0.769237	0.704789	0.848488
senticnet_ntusd_stocktwitlexi_afinn_soft	0.68193	0.769005	0.674791	0.895578
senticnet_ntusd_sentidd_soft	0.711312	0.76881	0.727181	0.816019
senticnet_ntusd_stocktwitlexi_soft	0.680795	0.768084	0.674625	0.893258
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_soft	0.699113	0.768068	0.704177	0.844789
senticnet_sentiwordnet_stocktwitlexi_sentidd_soft	0.696968	0.767351	0.701433	0.847001
senticnet_ntusd_sentiwordnet_afinn_soft	0.688356	0.767156	0.687292	0.870289
ntusd_afinn_sentidd_soft	0.731421	0.767139	0.785054	0.751097
sentiwordnet_stocktwitlexi_soft	0.658625	0.766296	0.645341	0.943306
senticnet_stocktwitlexi_afinn_sentidd_soft	0.694393	0.76586	0.699078	0.84679
senticnet_ntusd_sentiwordnet_vader_sentidd_leave2soft	0.713598	0.765724	0.732394	0.807074
ntusd_sentidd_soft	0.731007	0.765489	0.788642	0.74509
senticnet_ntusd_sentiwordnet_soft	0.681491	0.762773	0.682116	0.866809
senticnet_stocktwitlexi_sentidd_soft	0.690103	0.762691	0.696506	0.842783
senticnet_ntusd_afinn_soft	0.683205	0.762684	0.684943	0.862384
senticnet_ntusd_sentiwordnet_afinn_vader_leave2soft	0.70289	0.761796	0.718302	0.815924
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_hard	0.694898	0.761739	0.704499	0.833308
senticnet_ntusd_sentiwordnet_stocktwitlexi_sentidd_hard	0.70019	0.761496	0.715104	0.818555
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_leave2soft	0.68125	0.761246	0.678962	0.86893
senticnet_ntusd_sentiwordnet_afinn_sentidd_leave2soft	0.705159	0.760099	0.725679	0.801274
senticnet_ntusd_sentiwordnet_stocktwitlexi_vader_hard	0.688025	0.759626	0.694496	0.843111

senticnet_sentiwordnet_stocktwitlexi_afinn_soft	0.664132	0.759386	0.658898	0.896419
senticnet_ntusd_soft	0.677052	0.759209	0.679527	0.86175
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_leave2soft	0.678376	0.758591	0.6782	0.862289
senticnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.694914	0.758445	0.703199	0.825731
senticnet_stocktwitlexi_afinn_soft	0.659279	0.757258	0.654741	0.898209
senticnet_ntusd_stocktwitlexi_vader_sentidd_hard	0.70718	0.75656	0.735691	0.785999
senticnet_sentiwordnet_stocktwitlexi_soft	0.658268	0.755854	0.654783	0.894099
senticnet_ntusd_afinn_vader_sentidd_leave2soft	0.710708	0.755732	0.740311	0.778943
senticnet_sentiwordnet_vader_sentidd_soft	0.703204	0.75571	0.727018	0.788
senticnet_sentiwordnet_afinn_vader_sentidd_soft	0.701771	0.754366	0.725877	0.786525
senticnet_ntusd_sentiwordnet_hard	0.691048	0.754152	0.709963	0.808752
senticnet_stocktwitlexi_soft	0.651121	0.75201	0.649278	0.893677
senticnet_ntusd_stocktwitlexi_afinn_vader_hard	0.696579	0.751786	0.720448	0.792321
senticnet_ntusd_stocktwitlexi_afinn_sentidd_hard	0.704745	0.751353	0.740485	0.768291
senticnet_sentiwordnet_afinn_vader_soft	0.685349	0.749818	0.700343	0.808864
senticnet_sentiwordnet_vader_soft	0.684065	0.749089	0.699236	0.808653
senticnet_vader_sentidd_soft	0.698326	0.748516	0.726797	0.773354
senticnet_afinn_vader_sentidd_soft	0.697754	0.74749	0.727112	0.77093
senticnet_sentiwordnet_afinn_sentidd_soft	0.686607	0.744631	0.711439	0.782096
senticnet_ntusd_vader_hard	0.691155	0.744237	0.722566	0.774932
senticnet_sentiwordnet_stocktwitlexi_hard	0.645216	0.743301	0.64845	0.872717
senticnet_sentiwordnet_sentidd_soft	0.682897	0.74254	0.708192	0.781251
senticnet_afinn_vader_soft	0.678326	0.741392	0.697333	0.794641

senticnet_vader_soft	0.677473	0.741162	0.696487	0.795167
senticnet_ntusd_afinn_hard	0.690443	0.73935	0.730402	0.754379
ntusd_sentiwordnet_afinn_vader_sentidd_leave2soft	0.709241	0.73823	0.7673	0.718877
senticnet_stocktwitlexi_afinn_hard	0.653746	0.736943	0.664703	0.829306
senticnet_stocktwitlexi_vader_hard	0.648454	0.736616	0.656943	0.841532
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.685613	0.736359	0.716317	0.762503
senticnet_stocktwitlexi_sentidd_hard	0.656636	0.734979	0.672837	0.812125
senticnet_sentiwordnet_afinn_soft	0.651477	0.734589	0.665403	0.821608
ntusd_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.700968	0.733988	0.756568	0.721615
senticnet_sentiwordnet_soft	0.645332	0.731255	0.660311	0.820974
senticnet_afinn_sentidd_soft	0.671567	0.73023	0.702138	0.761969
senticnet_ntusd_sentidd_hard	0.697342	0.729927	0.768418	0.702105
senticnet_sentidd_soft	0.669431	0.728786	0.701497	0.759226
sentiwordnet_vader_sentidd_soft	0.702441	0.727623	0.770739	0.692108
ntusd_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.685374	0.727226	0.733186	0.728567
sentiwordnet_afinn_vader_sentidd_soft	0.700437	0.726784	0.766811	0.693795
sentiwordnet_afinn_vader_soft	0.686889	0.726169	0.735237	0.72193
sentiwordnet_vader_soft	0.685754	0.724985	0.735159	0.720033
ntusd_sentiwordnet_vader_hard	0.691262	0.722611	0.757123	0.700535
ntusd_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.690815	0.722051	0.756503	0.697266
senticnet_afinn_soft	0.633853	0.718791	0.655	0.798531
senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.687907	0.718226	0.74888	0.69875
senticnet_sentiwordnet_afinn_vader_sentidd_leave2soft	0.674408	0.715701	0.719375	0.718458

senticnet_ntusd_sentiwordnet_stocktwitlexi_afinn_vader_sentidd_leave2soft	0.703941	0.715628	0.799318	0.657228
senticnet_sentiwordnet_stocktwitlexi_vader_sentidd_hard	0.666855	0.713805	0.709647	0.724462
senticnet_ntusd_sentiwordnet_vader_sentidd_hard	0.691634	0.71268	0.768377	0.67525
sentiwordnet_stocktwitlexi_vader_hard	0.644288	0.710905	0.677373	0.752274
ntusd_sentiwordnet_afinn_hard	0.680397	0.70912	0.758189	0.671969
sentiwordnet_stocktwitlexi_afinn_hard	0.648288	0.70887	0.687368	0.734778
senticnet_sentiwordnet_stocktwitlexi_afinn_vader_hard	0.657124	0.707976	0.699694	0.72193
senticnet_ntusd_sentiwordnet_afinn_vader_hard	0.674466	0.706583	0.738843	0.685152
senticnet_sentiwordnet_stocktwitlexi_afinn_sentidd_hard	0.66413	0.704759	0.718124	0.697059
senticnet_ntusd_sentiwordnet_afinn_sentidd_hard	0.683328	0.701903	0.768538	0.653959
senticnet_sentiwordnet_vader_hard	0.653273	0.701413	0.702077	0.706332
sentiwordnet_afinn_sentidd_soft	0.667542	0.699174	0.737036	0.667548
sentiwordnet_afinn_soft	0.642565	0.692165	0.697872	0.689672
senticnet_sentiwordnet_afinn_hard	0.645969	0.688564	0.705173	0.677454
ntusd_stocktwitlexi_afinn_vader_sentidd_hard	0.680114	0.688554	0.774408	0.628885
senticnet_stocktwitlexi_afinn_vader_sentidd_hard	0.672221	0.686771	0.754315	0.638477
stocktwitlexi_vader_sentidd_hard	0.673082	0.686049	0.756097	0.637424
sentiwordnet_stocktwitlexi_sentidd_hard	0.632097	0.679194	0.693351	0.668174
stocktwitlexi_afinn_vader_hard	0.659219	0.678445	0.74005	0.633414
afinn_vader_sentidd_soft	0.673339	0.676358	0.774552	0.607283
senticnet_ntusd_afinn_vader_sentidd_hard	0.672668	0.675475	0.776571	0.607598
sentiwordnet_stocktwitlexi_afinn_vader_sentidd_hard	0.665645	0.671661	0.766486	0.604856
afinn_vader_soft	0.655633	0.669149	0.743213	0.61687

vader_sentidd_soft	0.669181	0.668608	0.774115	0.596325
sentiwordnet_sentidd_soft	0.642772	0.66847	0.728706	0.618962
senticnet_afinn_vader_hard	0.65391	0.66818	0.742954	0.61339
ntusd_afinn_vader_hard	0.66095	0.667333	0.763678	0.600849
senticnet_vader_sentidd_hard	0.661488	0.665953	0.759147	0.603385
ntusd_sentiwordnet_sentidd_hard	0.659949	0.664224	0.791401	0.577855
ntusd_sentiwordnet_afinn_vader_sentidd_hard	0.66967	0.662695	0.797034	0.575452
ntusd_vader_sentidd_hard	0.676586	0.662242	0.827802	0.563015
senticnet_sentiwordnet_afinn_vader_sentidd_hard	0.658631	0.661369	0.763831	0.591051
senticnet_sentiwordnet_sentidd_hard	0.634076	0.658408	0.721089	0.609479
sentiwordnet_afinn_vader_hard	0.650332	0.655997	0.752212	0.588731
stocktwitlexi_afinn_sentidd_hard	0.651343	0.652851	0.764966	0.57471
afinn_sentidd_soft	0.645314	0.639202	0.771964	0.549628
senticnet_afinn_sentidd_hard	0.64133	0.637132	0.761865	0.554372
ntusd_afinn_sentidd_hard	0.651675	0.624567	0.829224	0.507154
afinn_vader_sentidd_hard	0.633736	0.620334	0.76786	0.526866
sentiwordnet_vader_sentidd_hard	0.640585	0.619528	0.784804	0.519177
sentiwordnet_afinn_sentidd_hard	0.622862	0.59236	0.789301	0.478384

Figure 99: Lexicon Ensemble Test Results for Average of Data 2 and 3