MIAMI

Department of Computer Science

Analyzing Stock Market Prices using Twitter

Jiajun Chen

Joseph Chakko

Date Mining Project 2 Report

Fall 2014

Table of Contents

I. Introduction and motivation

One issue with writing is that the original feelings of the author may not be captured. It can be difficult for humans and even more challenging for machines to empathize or determine sarcasm, especially over the Internet. Sentiment analysis is a technique employed by computers to estimate the emotion within a text. When paired with Twitter, a candid social media platform that generates millions of user written texts daily, many applications for sentiment analysis appear. The emotions within tweets are valuable because they reflect the thoughts of real people. Still, there is far too much data to be processed by humans, and computers do not understand the nuances of our language. Can sentiment data mined from Twitter be practically applied?

In an attempt to answer this question, we have developed a program that seeks correlations between a company's stock price and the emotion found in the tweets about that company. A corporation that is receiving praise online should have rising stock prices, and a company that is being spoken about negatively will see a decrease in their stock value. This would be most prominent if a company started a new advertising campaign online or was receiving backlash from a recent scandal over Twitter. To test this hypothesis, we needed to gather the financial and emotional data each day for a month.

II. Design and structure

The structure of our program involves two branches of data that are processed and compared with each other to see if a relationship exists. The first branch is the stock market data that is primarily used to see changes in stock prices from day to day. The second branch is the tweets information collected from Twitter. More processing is required here as the tweets must be cleaned, parsed, stemmed, and finally analyzed for emotional content. In the end, do data mining processes by applying different algorithms on the data set. Figure II-1 shows the flow diagram of the TwitterStock program.

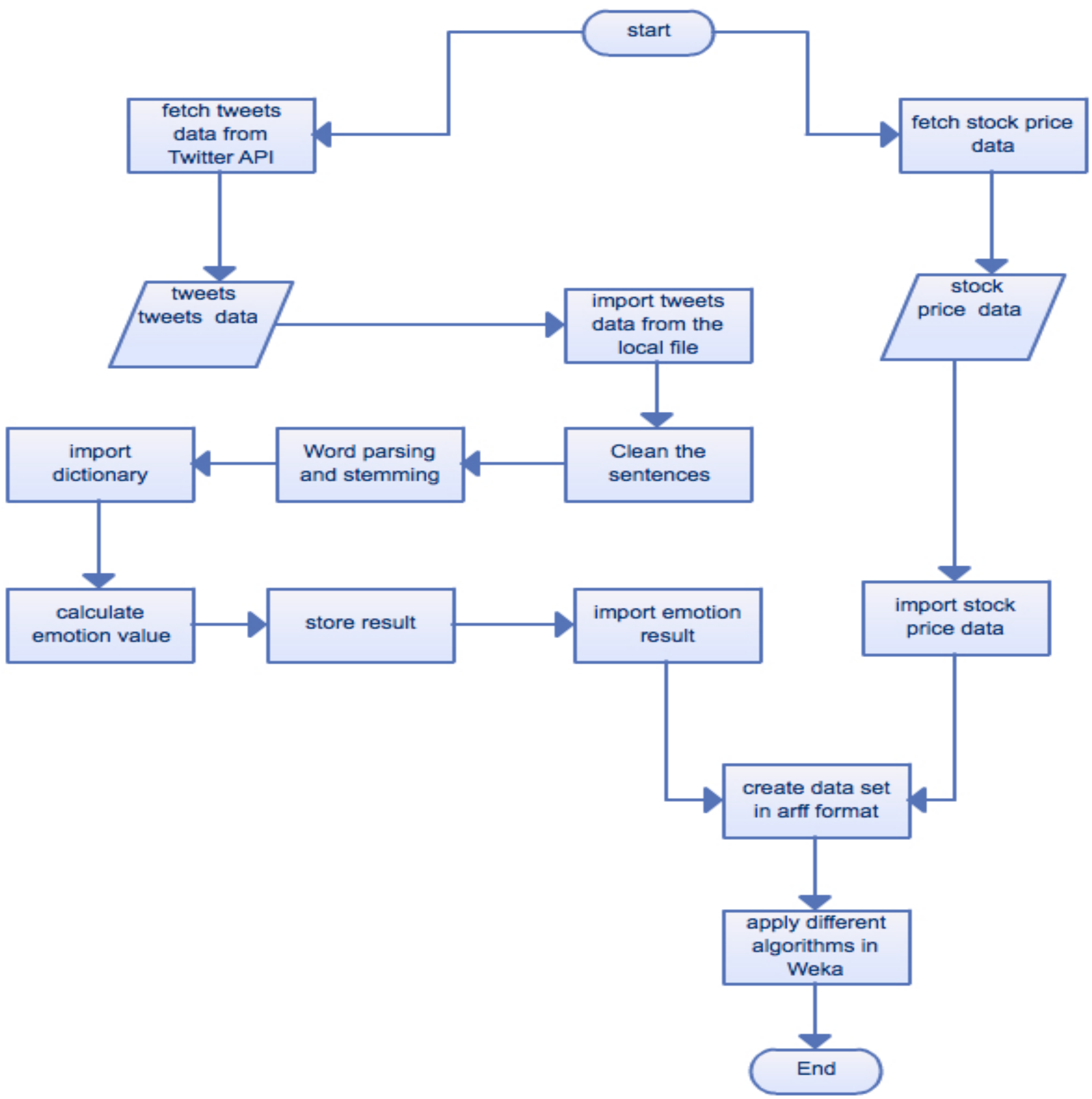


Figure II-1: Flow diagram of the TwitterStock program

Three different kinds of functionality have been implemented as three use cases of the program, which includes the whole KDD (Knowledge Discovery in Database) process. The first functionality is fetching stock market and tweets data from different sources. With respect to those collected data, the second functionality has been implemented to clean and analyze those raw data and convert them into a "data-mineable" format. The last functionality calls Weka functions and applies different algorithms on those converted data. Figure II-2 shows the use case diagram of the program that analyzes stock market prices using Twitter.

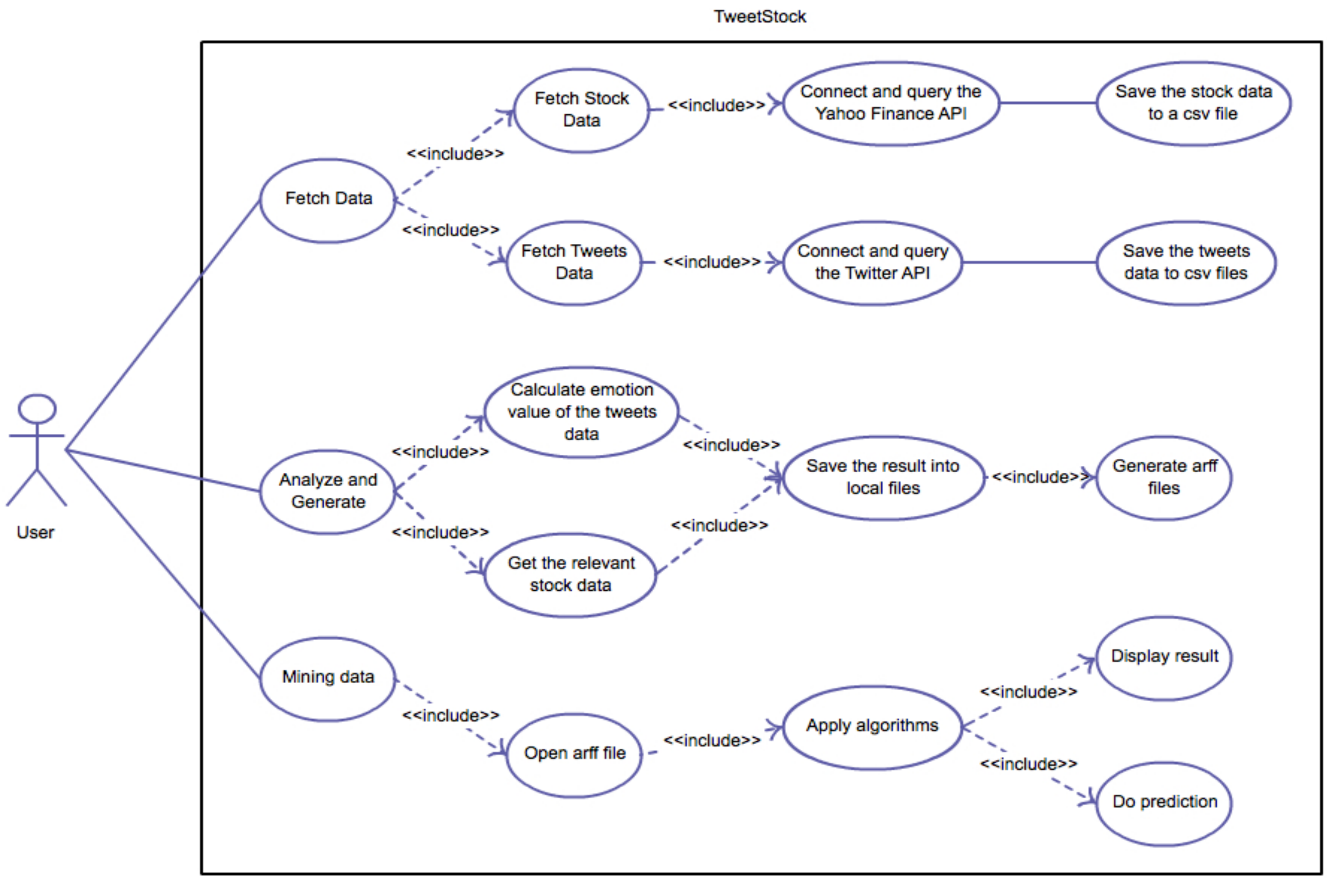


Figure II-2: Use case diagram of the TwitterStock program

The class structure of our program includes two interfaces and several classes. The interface Menu connects to the classes of data fetching, data analyzing and it also connects to the Output interface. Tweets are cleaned, parsed, stemmed, and finally analyzed for emotional content by the classes of data analyzing. The Output interface shows the result when applying data mining algorithms onto the stock market and tweets data. Basically the class structure follows from the three different kinds of functionality that have been mentioned above in the use cases diagram. Figure II-3 indicates the class diagram of the TwitterStock program.

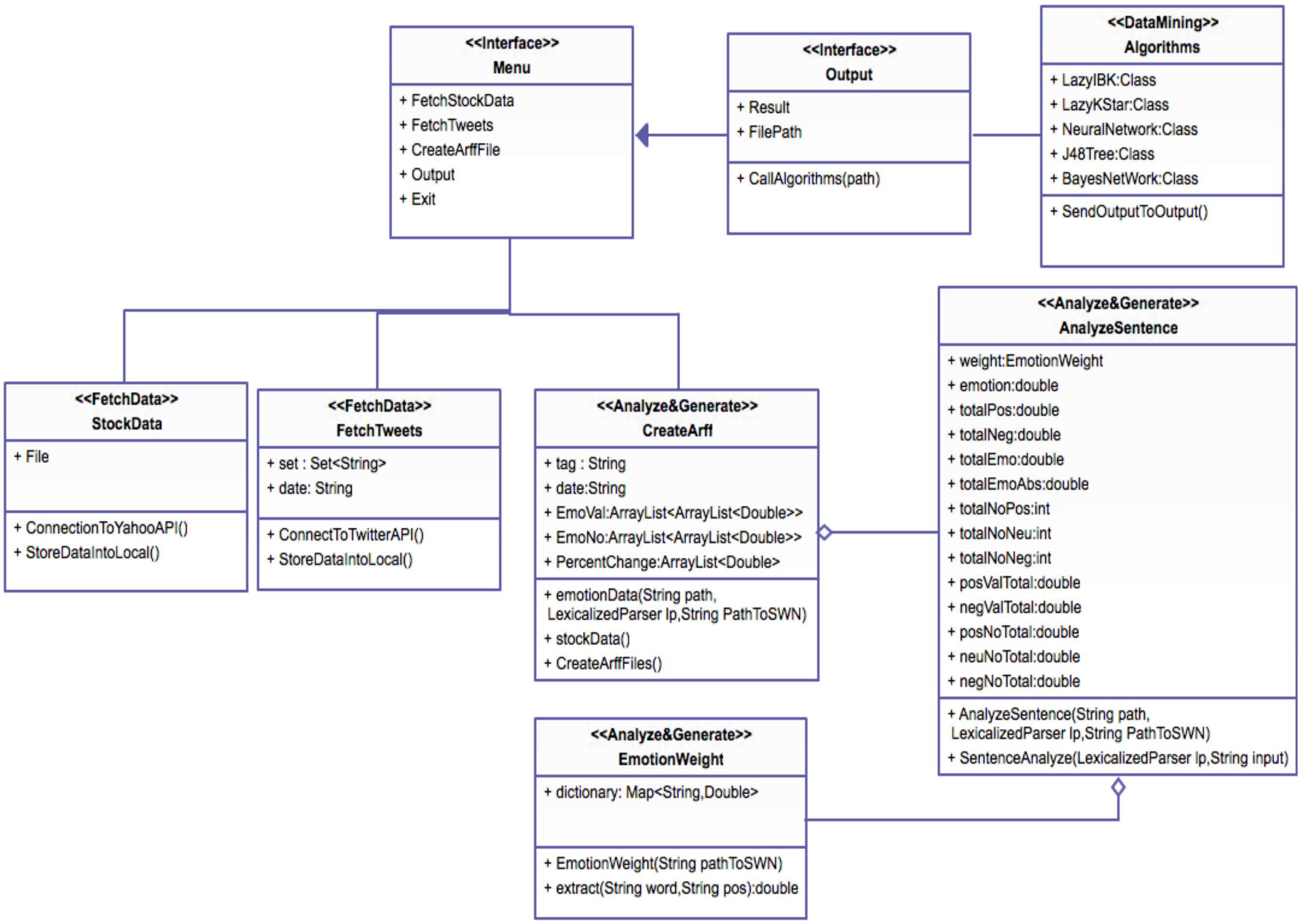


Figure II-3: Class diagram of the TwitterStock program

III. Implementation

I. Data Fetching and Implementation Abstract

The stock prices were gathered using the Yahoo! Finance API. The application contains the daily market information like the opening and closing price as well as the highest and lowest price from that day. The database is queried by writing commands as a URL that determines which companies to query and what information to retrieve. The API returns a comma separated values (CSV) file that is saved to your computer. The java class StockData helps handles the process. StockData opens a connection to the API using a URL containing the stock symbols of the 65 companies we followed for this project and saves the .csv file to the public folder of the computer. Figure III-1 shows a screenshot of the comma separated values file of the stock market data.

Tag	Name	Ask	Bid	Previous Clo:	Open	Change & Pe	Day's Value (	Trade Date	Days High	Days Low
VZ	Verizon Com	N/A	N/A	51.2	51.26	0.2941	0.0059	15-11-2014	51.73	51.18
CMCSA	Comcast Cor	N/A	51.56	54.3	54.48	-0.1963	-0.0037	15-11-2014	54.48	53.77
F	Ford Motor (	N/A	N/A	14.93	15.01	0.1959	0.0141	15-11-2014	15.265	14.985
GT	The Goodyea	26	25.03	25.425	25.18	0.1873	0.0077	15-11-2014	25.64	25.14
MPAA	Motorcar Pa	35	N/A	34.38	34.3	-0.3398	-0.0102	15-11-2014	34.43	33.75
SMP	Standard Mc	N/A	N/A	38.96	38.99	-0.8282	-0.0218	15-11-2014	39.02	38.03
GPRO	GoPro, Inc.	N/A	77.06	76.47	77.99	2.645	0.035	15-11-2014	79.49	76.75
COKE	Coca-Cola Bc	91.97	91.66	90.08	90.16	1.5822	0.0178	15-11-2014	91.98	90.01
DPS	Dr Pepper Sr	N/A	N/A	70.71	70.73	-0.6014	-0.0086	15-11-2014	70.77	69.81
MNST	Monster Bev	N/A	3	108.16	108.33	-0.2378	-0.0022	15-11-2014	108.56	107.15
PEP	Pepsico, Inc.	N/A	N/A	98.54	98.55	-0.8117	-0.0083	15-11-2014	98.55	97.36
CPB	Campbell Sou	N/A	N/A	43.85	43.85	-0.0977	-0.0023	15-11-2014	44.11	43.65
GIS	General Mills	N/A	N/A	51	50.99	-0.1765	-0.0035	15-11-2014	51.04	50.64
K	Kellogg Com	N/A	N/A	63.8	63.8	0.0098	0.0002	15-11-2014	64.02	63.64
KRFT	Kraft Foods (	58.1	56.75	57.63	57.81	-0.2064	-0.0036	15-11-2014	57.82	57.21
TSN	Tyson Foods,	N/A	N/A	41.18	41.18	-0.5074	-0.0126	15-11-2014	41.24	40.32
SJM	J.M. Smucke	N/A	N/A	99.87	99.76	0.3564	0.0036	15-11-2014	100.61	99.295
CL	Colgate-Palm	N/A	N/A	68.29	68.2	-0.7193	-0.0107	15-11-2014	68.34	67.51
PG	Procter & Ga	N/A	N/A	88.6	88.7	-0.4845	-0.0055	15-11-2014	88.89	87.93
REV	Revlon, Inc. I	N/A	N/A	33.56	33.46	-0.3687	-0.0113	15-11-2014	33.77	33.05
MORN	Morningstar,	71.23	N/A	69.49	69.68	0.0887	0.0013	15-11-2014	69.8552	68.91
LOGI	Logitech Inte	15.88	10.92	13.975	14.17	0.2739	0.0211	15-11-2014	14.27	14.11
HPQ	Hewlett-Pack	N/A	N/A	36.36	36.44	0.5446	0.0154	15-11-2014	37.06	36.43
AMND	Advanced M	N/A	N/A	2.66	2.675	-0.0312	-0.0188	15-11-2014	2.7	2.61
AAPL	Apple Inc.	114.45	114.38	112.82	113.16	1.3479	0.0121	15-11-2014	114.19	113.05
BBRY	BlackBerry Li	11.23	11.16	12.06	11.89	-0.7887	-0.0713	15-11-2014	12.2	11.11
GRMN	Garmin Ltd.	56.25	48.52	55.93	56.11	0.0196	0.0004	15-11-2014	56.53	55.865
MSI	Motorola Sol	N/A	N/A	64.43	64.38	0.8073	0.0127	15-11-2014	65.27	64.11
COF	Capital One I	N/A	N/A	81.6	81.41	0.158	0.002	15-11-2014	81.96	81.405
CIT	CIT Group In	N/A	N/A	49.27	49.26	-0.2841	-0.0059	15-11-2014	49.54	48.9
BAC	Bank of Ame	N/A	N/A	17.22	17.165	-0.0754	-0.0046	15-11-2014	17.25	17.1
WFC	Wells Fargo 8	N/A	N/A	53.39	53.39	-0.0393	-0.0007	15-11-2014	53.685	53.23
ANF	Abercrombie	N/A	N/A	28.9	28.36	0.00 - 0.00%	0	15-11-2014	29.05	28.22
ARO	Aeropostale	N/A	N/A	2.87	2.88	0.0586	0.0314	15-11-2014	3.03	2.86
AEO	American Ea	N/A	N/A	13.58	13.62	-0.0741	-0.0059	15-11-2014	13.7699	13.48
BURL	Burlington St	N/A	N/A	42.26	42.17	0.0391	0.0009	15-11-2014	42.48	41.34
DSW	DSW Inc. Cor	N/A	N/A	31.94	31.94	-0.0581	-0.0019	15-11-2014	32.17	31.4699
EXPR	Express, Inc.	N/A	N/A	14.63	14.62	0.1025	0.0075	15-11-2014	14.8	14.59
FOSL	Fossil Group,	110	107.3	109.805	108.2	-1.4715	-0.0135	15-11-2014	108.72	105.91
URBN	Urban Outfit	31.79	N/A	31.645	30.92	-0.7699	-0.0251	15-11-2014	31.41	30.77
JCP	J.C. Penney (	N/A	N/A	7.1	7.16	0.2406	0.0394	15-11-2014	7.565	7.07
KSS	Kohl's Corpo	N/A	N/A	56.07	56.07	1.0313	0.0187	15-11-2014	57.18	55.812

Figure III-1: A screenshot of the collected stock market data

The Twitter data was gathered using the Twitter4j library. The library accesses the Twitter API, which allows authorized program to query and fetch tweets. The Twitter API is queried by sending queries that include keywords and other setting parameters via the Twitter4j library. The API returns the tweets data and the program saves the data to your computer. The java class FetchTweets takes care of the process. FetchTweets queries the Twitter API with queries containing the stock tags of the 65 companies we followed for this project and saves the data into a local folder. To organize this information, a directory is created for each day the data is fetched. Data collected before 17:00, it will be saved into the directory with the current date. Otherwise it will be placed into a folder with tomorrow's date.

Figure III-2 gives a screenshot of the collected tweets files and the tweets inside those files.

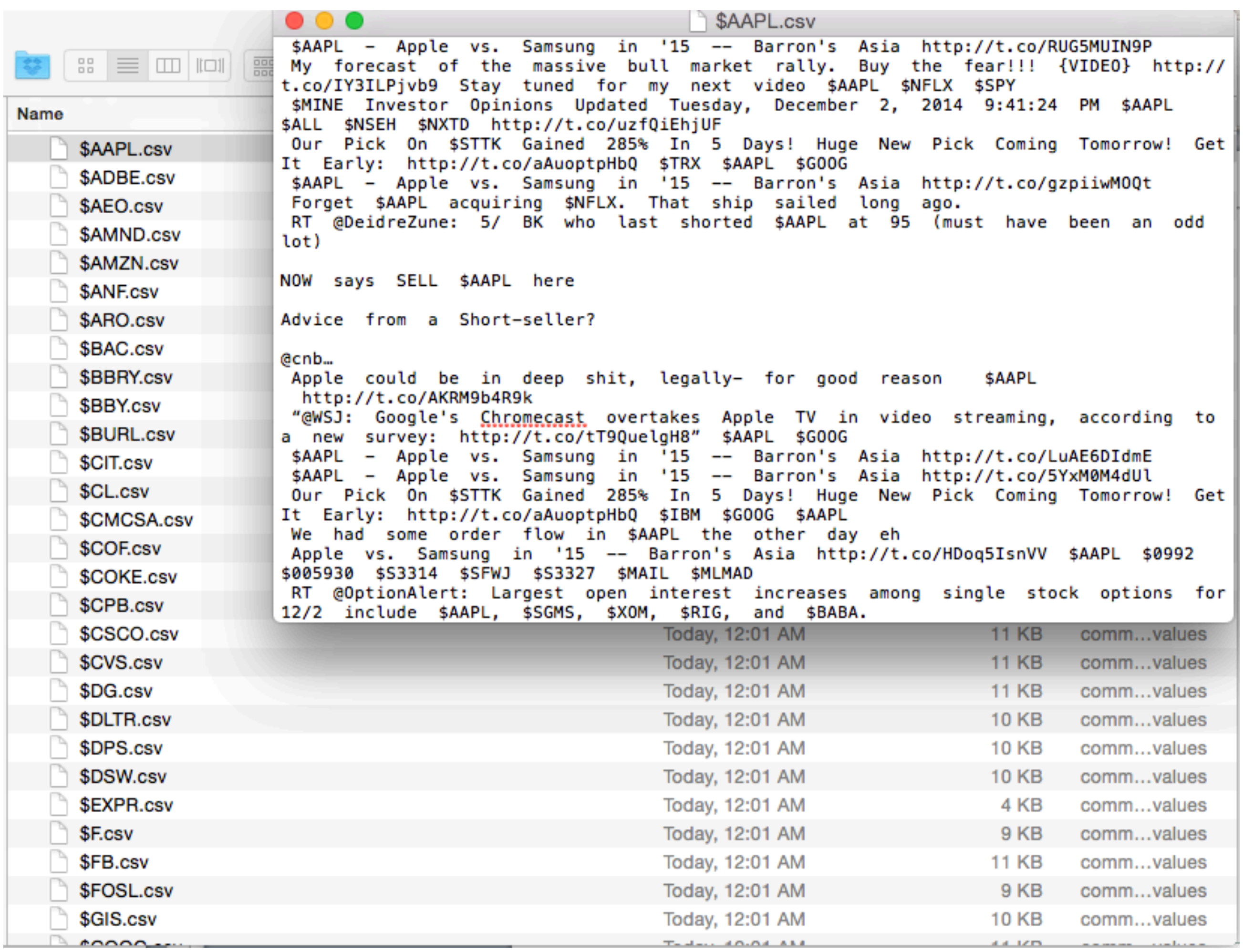


Figure III-2: A screenshot of the collected tweets files

Collecting tweet information every morning and night gave us the most data and the most flexibility in determining how emotions towards a company changed each day. Querying the Twitter API at night may retrieve data that was previously collected in the day. To clean the vast amount of data being collected, we needed to write a program that would remove duplicate values from a file. This was accomplished using a HashSet, a data structure that does not allow duplicate values. The program reads from a .csv file and adds each tweet to the HashSet. Tweets that already exist in the set will not be added. A file writer than creates a new file and writes in the contents of the HashSet, producing a file with no duplicate values.

After cleaning the data the actual sentiment analysis program could be implemented. There are two main steps in calculating the emotional impact of a tweet. The first is to preprocess the data to improve efficiency and accuracy. This is achieved through stemming, a process that shortens words. In the English language there are suffixes that modify words, changing the tense of a verb, or transforming adjectives into adverbs. In many cases, these suffixes can be dropped without significantly changing the original meaning of the word. Consider the words jumping, jump, and jumper. By removing the suffixes, we are left with the verb jump. Obviously changing the words used in the sentence will affect the meaning of the statement, however it should not change the sentiment by a noticeable amount. The benefit is that there are much fewer words that the program needs to recognize.

The program has to parse each word individually to determine recognition. This is handled using an external dictionary that was obtained from SentiWordNet. For faster query times, the program utilizes a HashMap to check if each word exists in the dictionary. If the word exists, the value of the word is factored in with the values of the other words in the sentence.

The word values are the core of the sentiment analysis process and range from negative one to positive one to reflect the emotional charge. Consider a word like malicious. This word is quite negative in its connotation and would receive a score closer to negative one. A word like stun with connotation determined by context would have a neutral score closer to zero. By analyzing each the sentiment of each word in a sentence we hope to determine the overall emotion is a statement. While this is not the way a human would perceive emotion within a text, it may be the most accurate way for a machine as writing a program that understands figures of speech and the nuances of language is very challenging and an issue that is yet to be solved.

After processing the stock information and tweets, we could begin to seek a correlation between sentiments on Twitter and stock market values. The main values to explore are the change and percent change in stock value and the overall sentiment towards a company from that day. The goal was to find all the occurrences where there is an increase or decrease in stock value accompanied by praise or criticism towards a company on Twitter.

II. Data Analyzing and generating

The first step in analyzing the data is having the machine parse the text files from Twitter. Machines do not understand natural language the way that humans do but attempts have been made to help computers navigate the nuances of English. The tool that we opted to use was the Stanford Parser [1][2], a program that helps a computer understand the structure of a sentence. Groups of words are parsed together in an attempt to identify phrases as well as which words are the subject of the statement. This program is not completely accurate but is usually able to determine the type of word like noun, verb, adjective or adverbial.

The next procedure is checking if words in the Tweet exist in a dictionary. The dictionary implemented is the SentiWordnet [3], a dictionary that also has emotional values for each word. For example, the word ill has an objective score of .25 and negative score of .75, implying that it will usually be used with a negative connotation. Each tweet is passed through the stemmer and checked against the dictionary. The emotional weight of each word is determined and all of the weights are summed together. If the word does not appear in the dictionary it simply returns the value of zero. An example of stemming, parsing and calculating the emotion of a sentence would be:

Sentence:

"The important thing in life is to have a great aim, and the determination to attain it. "(Johan Wolfgang von Goethe, German Poet and dramatist)

Analyze:

```
Please input the sentence:
The important thing in life is to have a great aim , and the determination to attain it.
thing --- pos: n; value: -0.01037537345531905
life --- pos: n; value: 0.008542901838558783
aim --- pos: n; value: 0.0
determination --- pos: n; value: -0.02737226277372263
important --- pos: a; value: 0.49908759124087587
great --- pos: a; value: 0.2593537414965987
have --- pos: v; value: 0.05020802460556254
attain --- pos: v; value: 0.0
(ROOT
```

```
(S
  (NP
    (NP (DT The) (JJ important) (NN thing))
    (PP (IN in)
      (NP (NN life))))
  (VP (VBZ is)
    (S
      (VP (TO to)
        (VP (VB have)
          (NP
            (NP (DT a) (JJ great) (NN aim))
            (, ,)
            (CC and)
            (NP (DT the) (NN determination)
              (S
                (VP (TO to)
                  (VP (VB attain)
                    (NP (PRP it))))))))))
  (. .)))
```

det(thing-3, The-1)

amod(thing-3, important-2)

nsubj(is-6, thing-3)

nsubj(have-8, thing-3)

prep_in(thing-3, life-5)

root(ROOT-0, is-6)

aux(have-8, to-7)

xcomp(is-6, have-8)

det(aim-11, a-9)

amod(aim-11, great-10)

dobj(have-8, aim-11)

det(determination-15, the-14)

dobj(have-8, determination-15)

conj_and(aim-11, determination-15)

aux(attain-17, to-16)

vmod(determination-15, attain-17)

dobj(attain-17, it-18)

The emotion value of this sentence is: 0.7794446229525542

This sentence has positive emotion :)

The goal is to achieve a particular set of values to be used in the data mining process. The first two are the positive emotion values over the total number of emotion values and the negative emotion values over the total number of emotion values in each tweet. Then the ratio of positive emotion values to total number of emotional values are calculated. The same is done with the neutral and negative emotion values. This data is stored with the stock information in four .csv files that account for numeric and nominal data. Then a csv converter is used to generate .arff files for data mining. Figure III-3 shows the result of the calculations of the emotion values and the stock market information.

$AMZN	Date	pos emo	neg emo	pos/totalabs	neg/totalabs	pos/total	neu/total	neg/total	change %	change value
	15-Nov	6.6657	-5.3344	0.5554	0.4445	0.44	0.27	0.29	11.3042	0.0358
	16-Nov	7.8418	-9.9821	0.4399	0.56	0.52	0.14	0.34	11.3042	0.0358
	17-Nov	7.3552	-4.956	0.5974	0.4025	0.52	0.25	0.23	-4.7554	-0.0146
	18-Nov	7.2141	-6.6448	0.5205	0.4794	0.56	0.12	0.32	1.8742	0.0058
	19-Nov	7.0802	-2.5859	0.7324	0.2675	0.34	0.21	0.45	1.605	0.005
	20-Nov	7.4363	-4.33085	0.6319	0.368	0.445	0.33	0.225	1.605	0.005
	21-Nov	9.98355	-4.2781	0.7	0.2999	0.52	0.18	0.3	2.0837	0.0063
	22-Nov	7.18155	-6.5809	0.5218	0.4781	0.37	0.29	0.34	2.0837	0.0063
	23-Nov	6.92725	-7.3057	0.4867	0.5132	0.39	0.215	0.395	-3.021	-0.009
	24-Nov	7.21385	-6.7571	0.5163	0.4836	0.465	0.215	0.32		
	25-Nov	7.2814	-3.12185	0.6999	0.3	0.415	0.205	0.38	-0.5982	-0.0018
	26-Nov	6.52595	-11.18445	0.3684	0.6315	0.375	0.195	0.43	-1.4656	-0.0044
	27-Nov	7.87655	-6.2525	0.5574	0.4425	0.52	0.22	0.26	-1.4656	-0.0044
	28-Nov	9.73705	-5.97035	0.6199	0.38	0.505	0.22	0.275	5.0548	0.0152
	29-Nov	19.9054	-10.2097	0.6609	0.339	0.51	0.17	0.32	5.0548	0.0152

Figure III-3: Emotion calculation result and stock market information of Amazon.

III. Data Mining

The final part of the project was mining the data to find a meaningful relationship between sentiments on Twitter and changes in stock market value. This was explored using different classifiers from Weka [4]. The data has been tested with all the available classifiers. Table 1 shows the result of applying all the available classifiers on the emotion value data (numeric).

CC = Correlation Coefficient

MAE = Mean Absolute Error

RMSE = Root Mean Squared Error

Category	Classifier	CC	MAE	RMSE
Function	GaussianProcess	-0.057	0.5403	0.9988
Function	IsotonicRegression	0.1052	0.5427	0.9976
Function	LeastMedSq	0.0619	0.5273	0.998
Function	Linear Regression	-0.004	0.5401	0.9991
Function	NeuralNetwork	-0.0232	0.6219	1.0461
Function	PaceRegression	0.0177	0.5395	0.9973
Function	RBFNetwork	-0.0152	0.5357	0.9973
Function	SMOreg	0.0524	0.5278	1.0002
Lazy	Ibk	0.0256	0.8138	1.3955
Lazy	Kstar	0.0769	0.5322	0.9988
Lazy	LWL	0.0554	0.5497	1.0058
Meta	AdditiveRegression	0.0426	0.5535	1.04
Meta	Bagging	0.0753	0.5503	1.0103
Meta	CVParameterSelction	-0.1322	0.5382	0.9972
Meta	MultiScheme	-0.1322	0.5382	0.9972
Meta	RandomSubSpace	0.0174	0.5338	1.0008
Meta	RegByDiscretization	-0.1016	0.54	1.0145
Meta	Stacking	-0.1322	0.5382	0.9972
Meta	Vote	-0.1322	0.5382	0.9972
Rules	ConjunctiveRule	-0.0415	0.5392	0.999
Rules	DecisionTable	-0.1322	0.5382	0.9972
Rules	M5Rules	-0.004	0.5401	0.9991
Rules	ZeroR	-0.1322	0.5382	0.9972
Trees	DecisionStump	0.0377	0.5454	1.0045
Trees	M5P	-0.004	0.5401	0.9991
Trees	REPTree	0.0641	0.5523	1.0081

Table 1: result of applying all the available classifiers on the emotion value data (numeric)

Figure III-4 shows the distribution of each attribute in the data set.

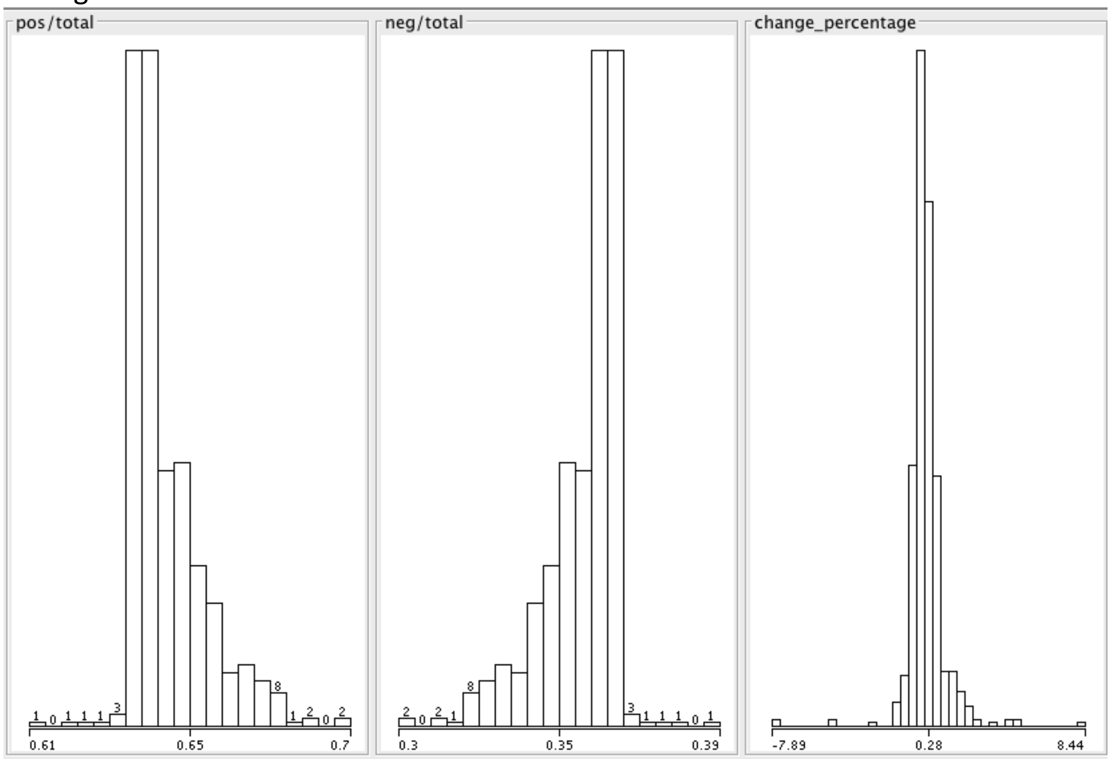


Figure III-4: distribution of each attribute in the data set

Table 2 shows the result of applying all the available classifiers on the number distribution of the emotion data (numeric).

Category	Classifier	CC	MAE	RMSE
Function	GaussianProcess	0.0633	0.5379	0.9949
Function	IsotonicRegression	0.081	0.5518	1.0025
Function	LeastMedSq	0.1213	0.5273	0.9931
Function	Linear Regression	0.0409	0.5384	0.998
Function	NeuralNetwork	0.0056	0.6158	1.0377
Function	PaceRegression	0.0574	0.5392	0.9955
Function	RBFNetwork	0.065	0.5363	0.9932
Function	SMOreg	0.096	0.5253	0.9958
Lazy	Ibk	0.0039	0.7677	1.3036
Lazy	Kstar	0.0914	0.5617	1.0086
Lazy	LWL	0.1152	0.5427	0.9899
Meta	AdditiveRegression	0.0619	0.551	0.9983
Meta	Bagging	0.0774	0.5488	1.002
Meta	CVParameterSelction	-0.1322	0.5382	0.9972
Meta	MultiScheme	-0.1322	0.5382	0.9972
Meta	RandomSubSpace	0.0389	0.5389	0.996
Meta	RegByDiscretization	-0.0612	0.5471	1.0073
Meta	Stacking	-0.1322	0.5382	0.9972
Meta	Vote	-0.1322	0.5382	0.9972
Rules	ConjunctiveRule	-0.0478	0.5402	0.9992
Rules	DecisionTable	-0.0397	0.5391	1.0054
Rules	M5Rules	0.0409	0.5384	0.998
Rules	ZeroR	-0.1322	0.5382	0.9972
Trees	DecisionStump	0.0221	0.557	1.0157
Trees	M5P	0.0409	0.5384	0.998
Trees	REPTree	0.0342	0.5688	1.0283

Table 2: result of applying all the available classifiers on the number distribution of the emotion data (numeric)

Figure III-5 shows the distribution of each attribute in the data set.

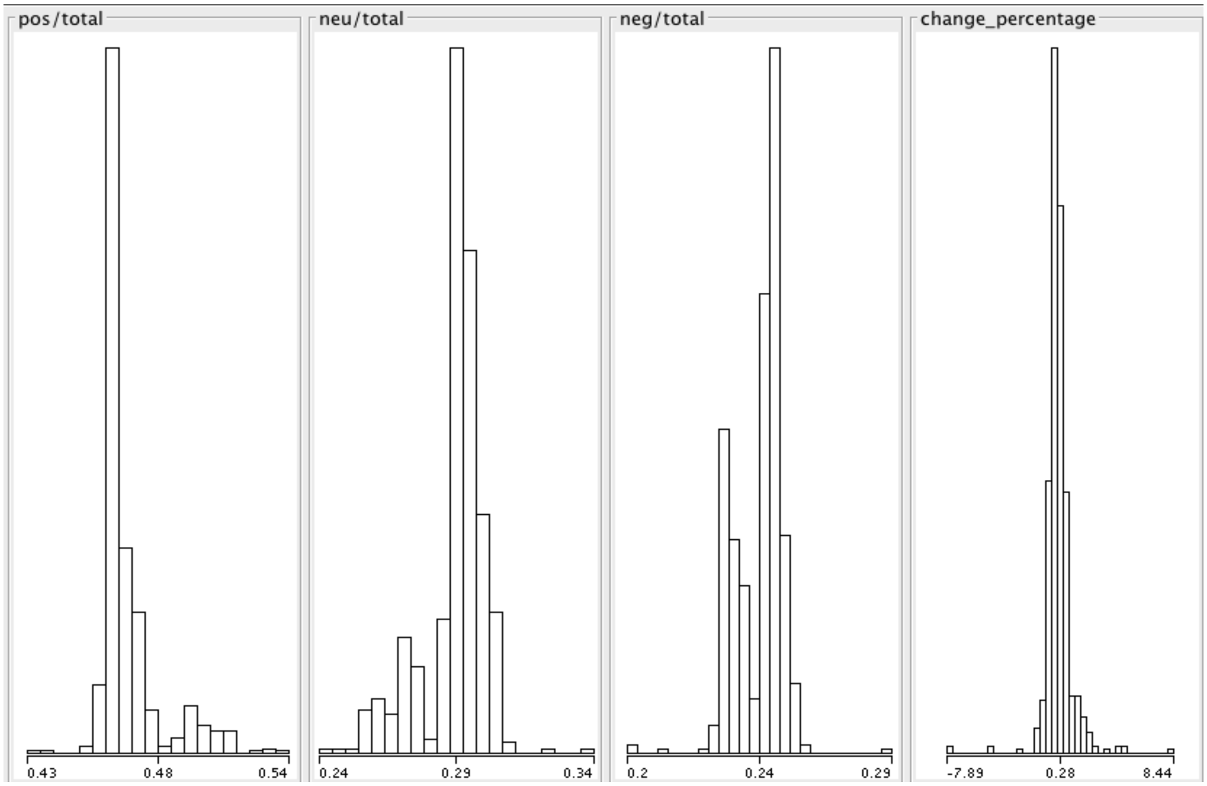


Figure III-5: distribution of each attribute in the data set

Table 3 shows the result of applying all the available classifiers on the emotion value data (nominal).

CC = Correctly Classified Instances

MAE = Mean Absolute Error

RMSE = Root Mean Squared Error

Category	Classifier	CC	MAE	RMSE
Bayes	BayesNet	57.56%	0.3276	0.4247
Bayes	ComplementNavieBayes	58.76%	0.2749	0.5243
Bayes	DMNBtext	54.47%	0.3704	0.4223
Bayes	NaiveBayes	51.03%	0.354	0.4332
Bayes	NavieBayesMultinomial	54.47%	0.354	0.4204
Bayes	NaiveBayesMultiUpdate	54.47%	0.3368	0.4248
Bayes	NaiveBayesSimple	50.86%	0.354	0.4335
Bayes	NaiveBayesUpdate	51.03%	0.354	0.4332
function	LibSVM	54.47%	0.3036	0.551
function	Logistic	52.92%	0.3504	0.419
function	NeuralNetwork	59.11%	0.3338	0.411
function	RBFNetwork	59.11%	0.3366	0.4127
function	SimpleLogistic	54.47%	0.4444	0.4714
function	SMO	54.47%	0.3326	0.4294
lazy	IB1	50.69%	0.3288	0.5734
lazy	IBK	51.37%	0.3306	0.569
lazy	Kstar	60.65%	0.3315	0.4114
lazy	LWL	58.76%	0.3377	0.4124
meta	AdaBoostM1	59.11%	0.4136	0.4446
meta	AttributeSelectedClassifier	59.28%	0.3335	0.4117
meta	Bagging	58.42%	0.3293	0.4234
meta	ClassificationViaClustering	54.30%	0.3047	0.552
meta	ClassificiationViaRegression	57.56%	0.3374	0.415
meta	CNParameterSelection	54.47%	0.354	0.4204
meta	Dagging	54.47%	0.3338	0.4291
meta	Decorate	59.97%	0.3717	0.4215
meta	END	57.56%	0.3369	0.4133
meta	FilteredClassifier	58.42%	0.3393	0.4138
meta	Grading	54.47%	0.3036	0.551
meta	LogisticBoost	58.59%	0.3284	0.411
meta	MultiBoostAB	59.11%	0.4093	0.4431
meta	MultiClassClassifier	52.91%	0.4068	0.4384
meta	MultiScheme	54.47%	0.354	0.4204
meta	RandomSubSpace	59.45%	0.3331	0.4133
meta	Stacking	54.47%	0.354	0.4204
meta	Vote	54.47%	0.354	0.4204
rules	ConjunctiveRule	58.25%	0.339	0.4156
rules	DecisionTable	58.42%	0.3406	0.4138
rules	DTNB	58.42%	0.34	0.4138
rules	Jrip	59.28%	0.3368	0.4149
rules	OneR	58.59%	0.2761	0.5254
rules	PART	59.11%	0.3336	0.4117
trees	DecisionStump	59.11%	0.3378	0.4123
trees	J48	59.28%	0.3334	0.4116
trees	LADTree	58.08%	0.3283	0.4171
trees	LMT	59.79%	0.4069	0.45
trees	NBTree	58.76%	0.3241	0.4213
trees	SimpleCart	59.79%	0.3326	0.4187

Table 3: result of applying all the available classifiers on the emotion value data (nominal)

Figure III-6 shows the distribution of each attribute in the data set.

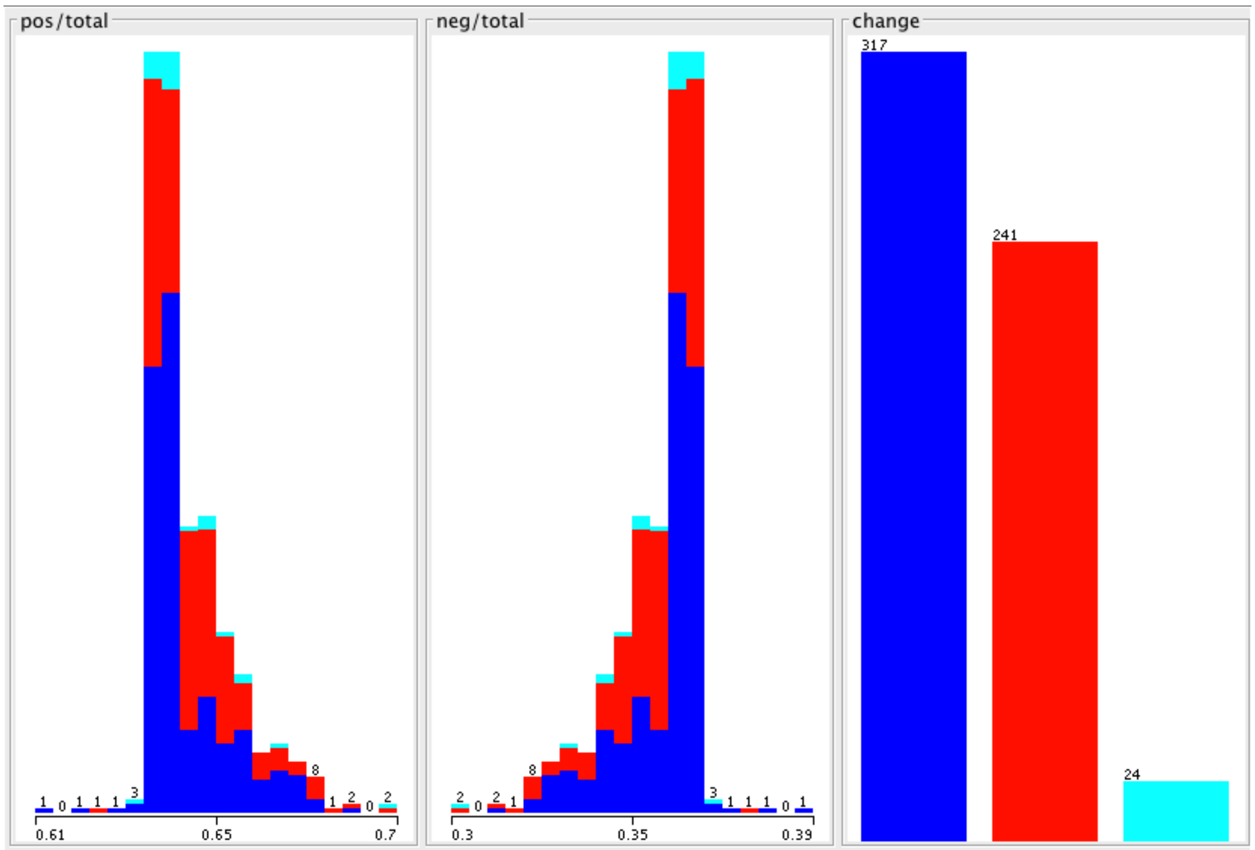


Figure III-6: distribution of each attribute in the data set

Table 4 shows the result of applying all the available classifiers on the number distribution of the emotion data (nominal).

Category	Classifier	CC	MAE	RMSE
Bayes	BayesNet	60.14%	0.3257	0.4212
Bayes	Complement	60.82%	0.2612	0.511
Bayes	DMNBtext	54.47%	0.3704	0.4223
Bayes	NaiveBayes	56.87%	0.3314	0.4356
Bayes	NavieBayesM	54.47%	0.354	0.4204
Bayes	NaiveBayesM	54.47%	0.3368	0.4248
Bayes	NaiveBayesS	56.87%	0.3315	0.4357
Bayes	NaiveBayesU	56.87%	0.3314	0.4356
function	LibSVM	54.47%	0.3036	0.551
function	Logistic	59.45%	0.3428	0.4151
function	NeuralNetwo	60.82%	0.3374	0.4148
function	RBFNetwork	59.45%	0.3385	0.4131
function	SimpleLogist	59.62%	0.3477	0.416
function	SMO	54.81%	0.3318	0.4286
lazy	IB1	51.89%	0.3207	0.5663
lazy	IBK	52.40%	0.3231	0.5627
lazy	Kstar	58.59%	0.3312	0.4238
lazy	LWL	58.42%	0.3381	0.4139
meta	AdaBoostM1	58.59%	0.4116	0.4443
meta	AttributeSele	56.36%	0.3421	0.4193
meta	Bagging	59.11%	0.3272	0.4228
meta	Classification	56.70%	0.2887	0.5373
meta	Classificiatioı	59.79%	0.3381	0.4133
meta	CVParameter	54.47%	0.354	0.4204
meta	Dagging	54.64%	0.3302	0.4215
meta	Decorate	58.42%	0.3617	0.4193
meta	END	57.22%	0.3415	0.4161
meta	FilteredClass	59.97%	0.338	0.414
meta	Grading	54.47%	0.3036	0.551
meta	LogisticBoost	59.11%	0.3348	0.4176
meta	MultiBoostAI	57.04%	0.3819	0.4382
meta	MultiClassCla	59.45%	0.4038	0.4357
meta	MultiScheme	54.47%	0.354	0.4204
meta	RandomSubS	59.62%	0.3344	0.4127
meta	Stacking	54.47%	0.354	0.4204
meta	Vote	54.47%	0.354	0.4204
rules	ConjunctiveR	55.67%	0.3421	0.4223
rules	DecisionTabl	59.97%	0.3391	0.4133
rules	DTNB	60.14%	0.3289	0.4164
rules	Jrip	58.93%	0.3369	0.415
rules	OneR	56.87%	0.2875	0.5362
rules	PART	57.22%	0.3409	0.4182
trees	DecisionStun	56.70%	0.3418	0.4173
trees	J48	56.36%	0.3421	0.4193
trees	LADTree	54.98%	0.3381	0.4307
trees	LMT	60.48%	0.3459	0.417
trees	NBTree	60.14%	0.3261	0.4212
trees	SimpleCart	57.39%	0.3365	0.4298

Table 4: result of applying all the available classifiers on the number distribution of the emotion data (nominal)

Figure III-7 shows the distribution of each attribute in the data set.

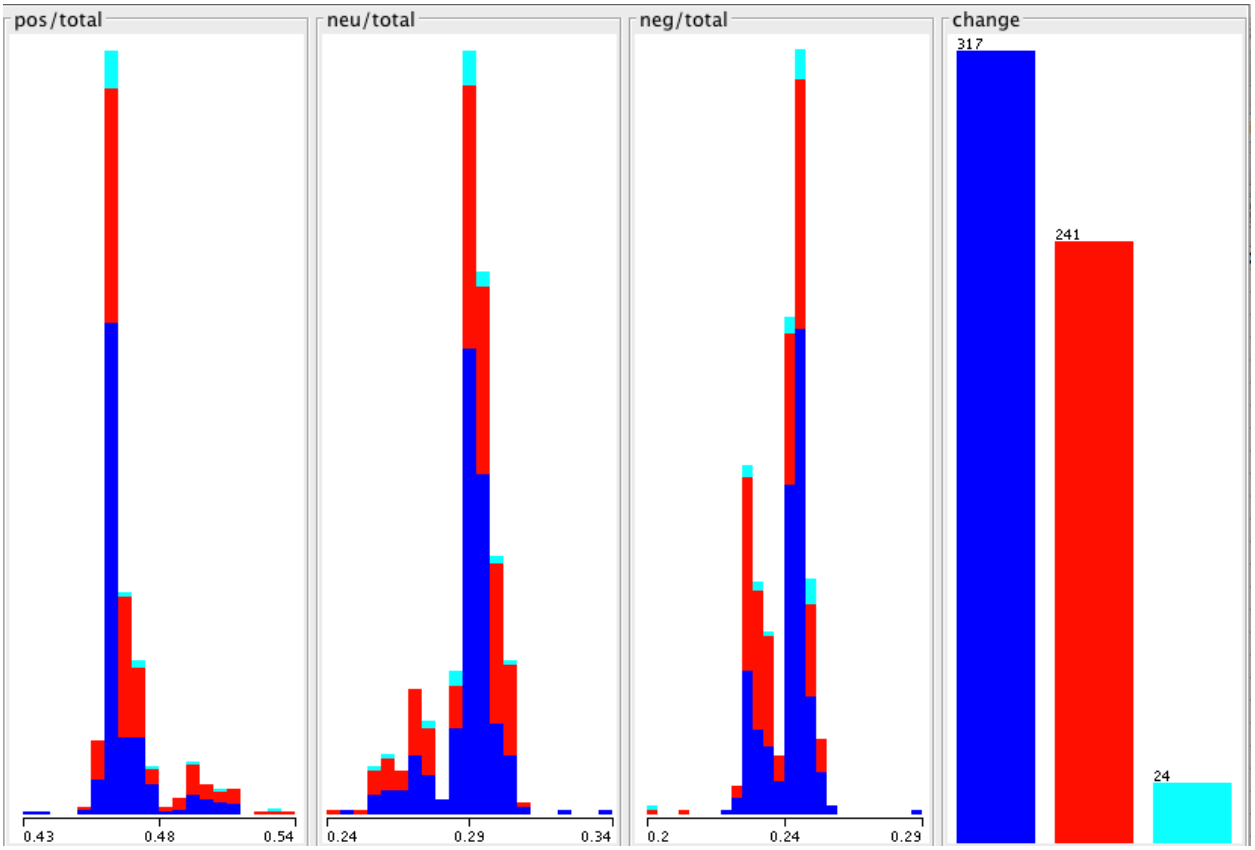


Figure III-7: distribution of each attribute in the data set

Overall, the algorithms that produced the best results were Lazy IBk, Lazy KStar, Neural Network, J48 Tree, and Bayes Network. Unfortunately all five of these algorithms returned an accuracy of 60% or lower. While we were pleased to find that the accuracy was greater than 50%, it is not high enough to be useful for predictions.

IV. Challenges and future works

Throughout the course of this project we faced many challenges. The first was the limited amount of data. Data collection began as soon as we finalized the idea for the project. This only allowed for twenty days’ worth of stock and twitter information. This experiment would be much more conclusive with a data from an entire year. Collecting the information from the APIs was also a challenge because there are limits to how many times they can be queried. Using the asking and bidding prices of stocks would be ideal but the API did not provide this data for the majority of the companies. Analyzing the information in a Tweet is difficult because they could have been written by anyone. Some tweets are nonsensical and may not be worth analyzing. Analyzing the emotional weight of each word was also limited to the words found in the SentiWordNet dictionary.

Moving forward, there are many improvements to be made. The data can be made more reliable through more intense cleaning and smarter parsing. A better algorithm may be able to find a better connection between Twitter and the stock market. More data and data that is processed and analyzed in real time would provide better results. With more time, correlations with different values can be explored. In this experiment we were only able to calculate using change and percentage change. Perhaps there is a stronger relationship with the daily high or low price or some other value.

V. Conclusion and Discussion

After completing this project, we were disappointed to find that there is not enough evidence to support the existence of a relationship between tweets about a company and the stock value of that company. That is not to say that Twitter cannot reflect changes in the stock market. It is possible, however it would probably take a large campaign for there to be a noticeable effect on the company; normal day to day operations should not affect the company. Still, the information gained from this experiment can be used in other manners. While people may not be able to alter the stock market through social media, companies are still interested in what people are saying about them over the Internet. It is not enough for people to simply tweet about a company. There were many tweets about Comcast but most of them were of negative emotions. Corporations should hope that the things being said are positive. The sentiment analysis portion of this project could be used to help companies keep tabs on their online presence and ensure that their customers remain positive.

VI. Reference

[1]. Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. *Generating Typed Dependency Parses from Phrase Structure Parses*. In LREC 2006.

[2]. Marcus,M.,Kim, G.,Marcinkiewicz, M. A.,MacIntyre, R.,Ferguson, M.,Katz, K. and Schasberger, B. 1994. *The Penn Treebank: Annotating Predicate Argument Structure*. HLT '94 Proceedings of the workshop on Human Language Technology, Page 114-119

[3]. S. Baccianella, A. Esuli, and F. Sebastiani. 2010. *SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining*. In Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC ’10), pages 2200–2204, Valletta, Malta.

[4]. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); *The WEKA Data Mining Software: An Update*; SIGKDD Explorations, Volume 11, Issue 1.

VII. Appendix

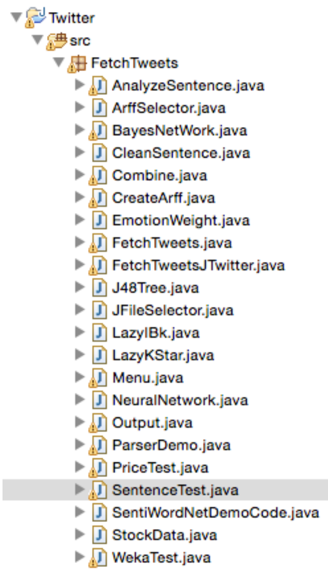
I. Included directories and files

TwitterStock.jar : a runnable jar file of the whole program

TwitterStock_lib : some libraries that are necessary for our program

Data : Contains some arff files that are created by the data analyzing process

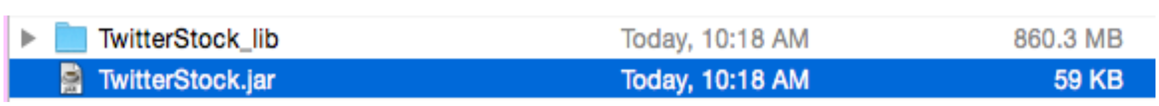
The list of classes:



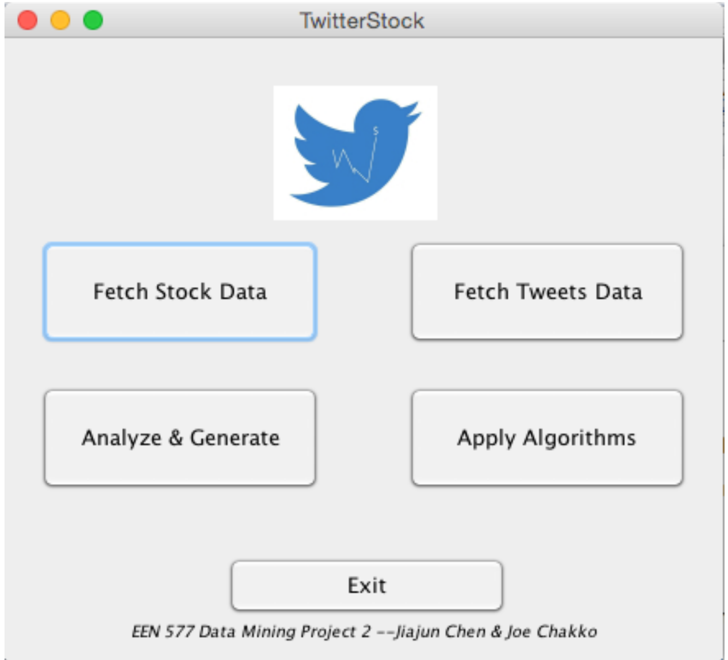
II. TwitterStock Program Manual

Demonstration:

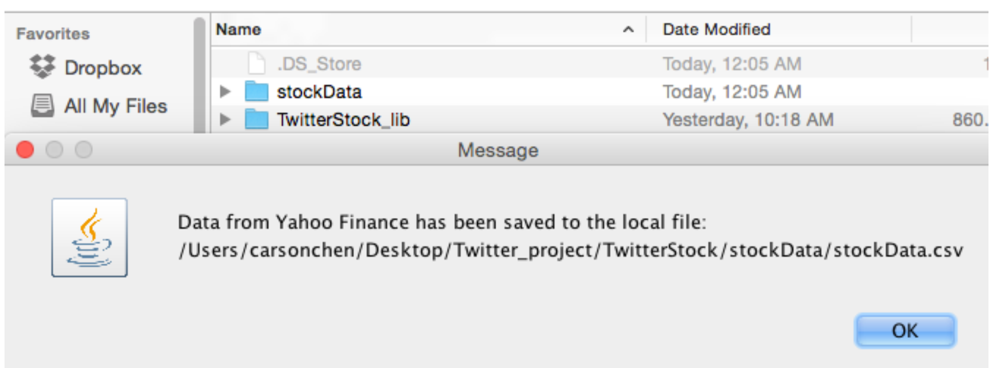
I. First, locate the TwitterStock runnable jar file and then double clicks on the file



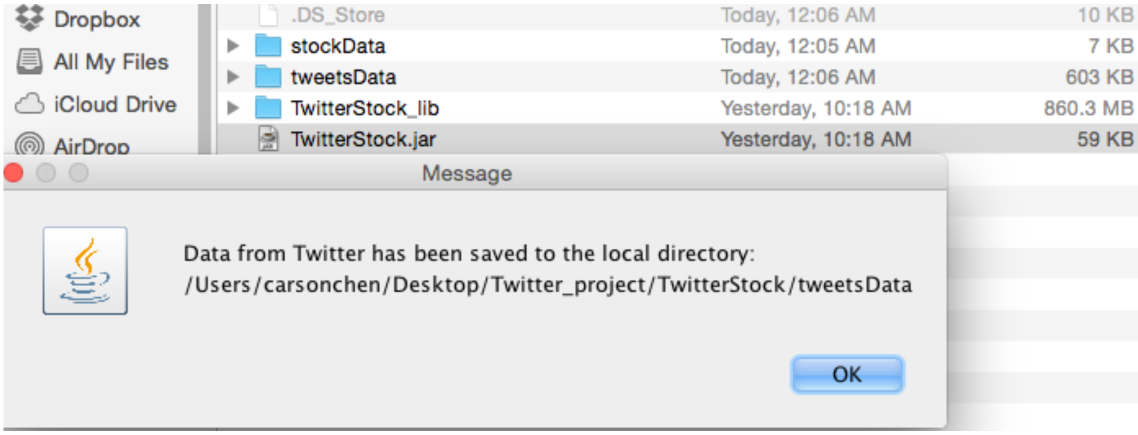
II. The Menu interface will show up



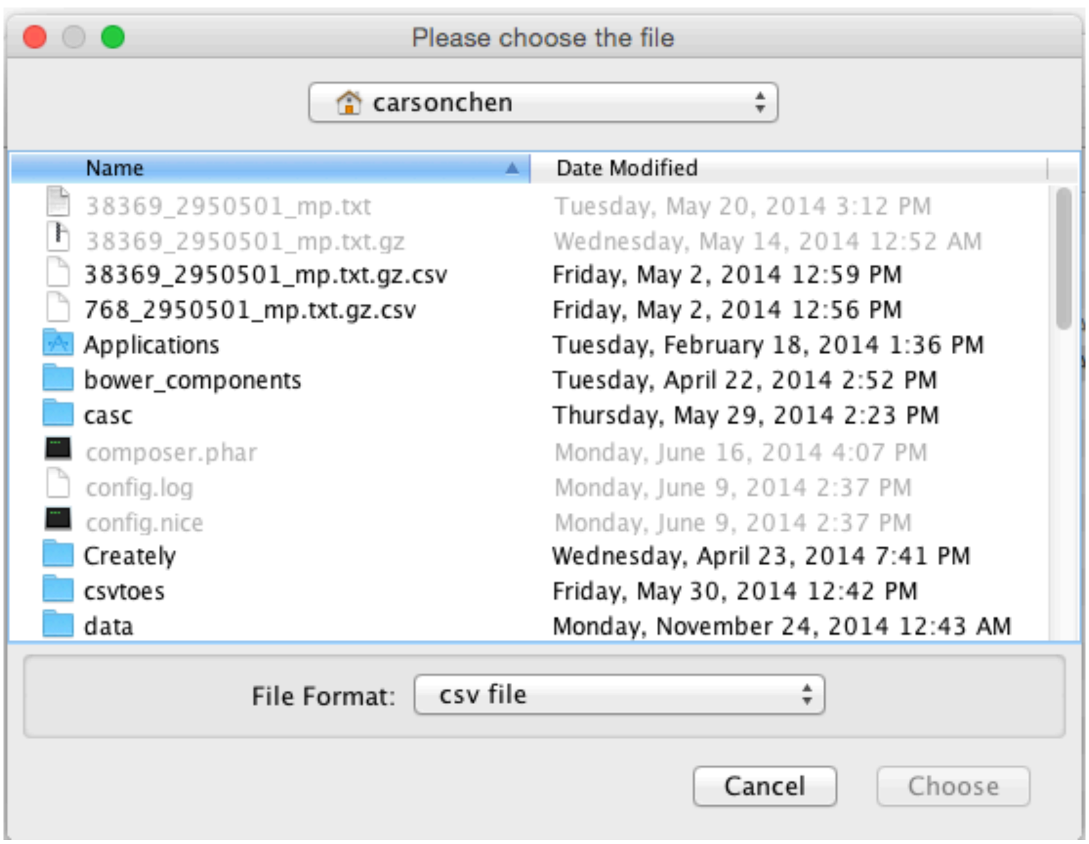
III. If the user presses the “Fetch Stock Data” button, the program will firstly create a directory called “stockData”, then it will download the stock market data and save the data into a csv file.



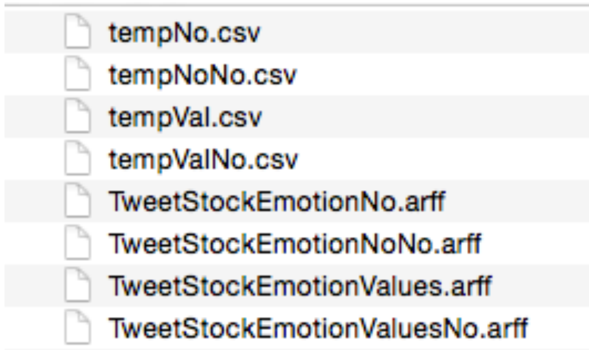
IV. If the user presses the “Fetch Tweets Data” button, the program will first create a directory called “TweetsData”, then it will download the tweets data and save the data into the file.



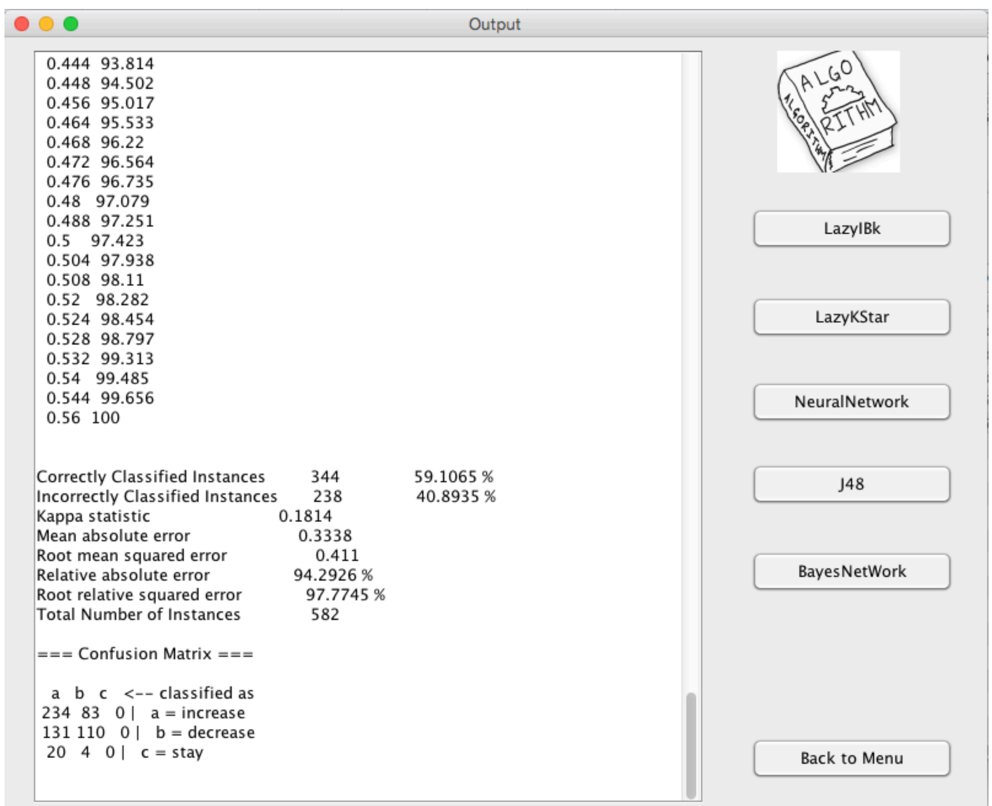
V. If the user presses the "Analyze & Generate" button, the program pops up a selector and asked the user to select the file(s) that are eligible for analyzing. The process will take a long time if the size of the selected amount of data is large.



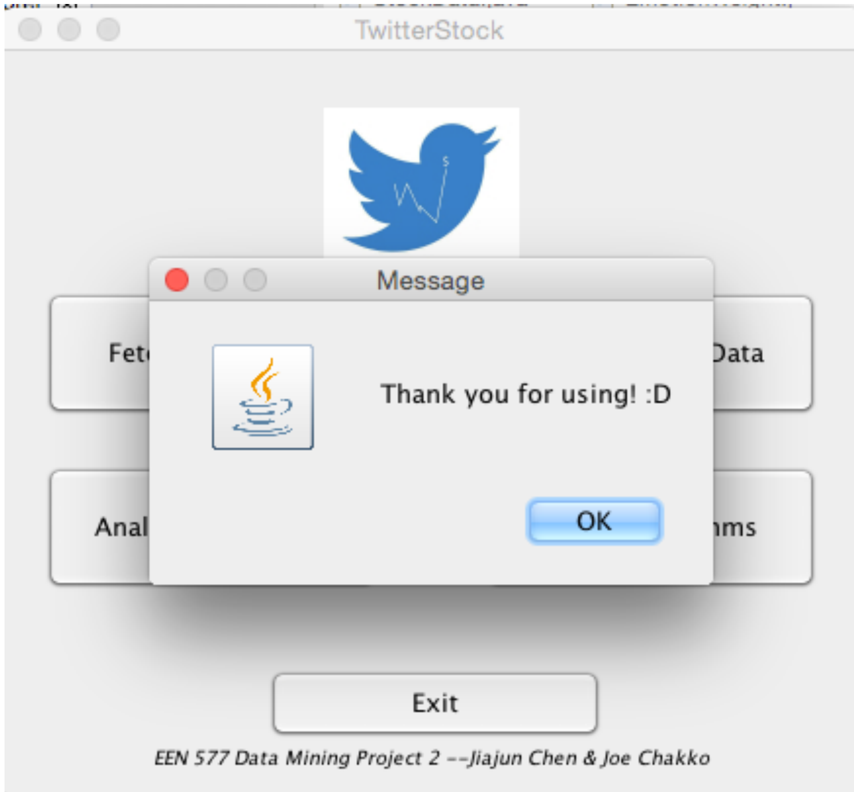
The program will generate the following files when the process is done.



VI. If the user presses the “Apply Algorithms” button, the program will ask the user to locate the arff files and shows the Output interface. Whenever the user selects an algorithm to train the data, the program will print out the result.



VII. If the user presses the “Exit” button, the program will be closed.


TwitterStock
Message
Thank you for using! :D
OK
Exit
EEN 577 Data Mining Project 2 --Jiajun Chen & Joe Chakko