Twitter Sentiment Analysis for Stock Market Direction

Udacity MLND Capstone Project

by Christian Graber 12/10/2016

1. Overview

Using Sentiment

Sentiment analysis allows for extraction of a writer's emotion from the text they authored. This emotion reflects the author's personal opinion on some subject. Using the wisdom of the crowds [B] such opinion in aggregate can provide actionable insights.

More formally, this is how Wikipedia defines sentiment analysis [A]:

Sentiment analysis *(also known as* ***opinion mining****) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.*

Companies are using automated sentiment analysis to improve their business [C]. The obvious thing to do is to monitor how the company is trending on social media and steer their social media outreach to influence reputation. But there are far more applications reaching from marketing to customer service.

In this capstone project the focus will be specifically on mining Twitter. On Twitter users post tweets. Those are small snippets of text limited to 140 characters in length. Users can address their tweets to another user, re-tweet some other user's tweets and use hashtags (#sometopic) to link their tweets to specific topics.

Dow Jones on Twitter

The topic of interest here is the stock market. Is it possible to detect stock market direction from tweets ? This is the question this capstone will tackle. To answer this question, we will use machine learning techniques taught in Udacity's machine learning nanodegree.

The stock market has seen a major runup following the 2016 presidential election. Leading up to the election the market was trending lower. Once the results were getting close, the market turned around and rallied. Especially the Dow Jones was positively affected. Here is a stock chart showing this runup: Figure 1: Dow Jones runup after 2016 US election.

Figure 1: Dow Jones runup after 2016 US election

To extract this move from tweets, we will search for all tweets with hashtag #dowjones on Twitter. The period we're looking into is November 2nd 2016 to December 1st 2016. See appendix for the actual Dow Jones prices [Table 1: Dow Jones prices in reference period].

Metrics

To measure the success of our opinion mining we will simply extract the sentiments 'positive', 'neutral' and 'negative' from Dow Jones tweets and compare them to the actual stock market direction in the analysis period. We won't be looking at the size of the move, except for the definition of what neutral means. A move in the market, comparing open to close prices for a specific day, which is smaller than a certain threshold, will be considered neutral. The best threshold to pick will come out of the actual machine learning results.

Here we are looking at 21 days of trading. So the ideal outcome would be a match of direction detected with actual move in 21 cases. Or so you

would think. One interesting insight from Biz360, a social media mining company, is that humans only agree 79% of the time [C]. So this will be our benchmark. We don't expect to hit better than 79% accuracy.

Unsupervised vs. Supervised vs. API

To train our system we need tweets that are labeled for the three sentiment categories, 'positive', 'neutral' and 'negative'. If such data is not available, one way to generate it is by use of the Amazon Mechanical Turk[D]. That means every tweet is labeled by a human. This capstone will not pursue this route.

The next best thing to labeling your own data is to use data available in the public domain. The closest found is the Twitter Sentiment Corpus by Sanders Analytics [E]. This corpus contains about 5500 hand-classified tweets related to the companies Apple, Google, Microsoft and Twitter.

The Sanders Corpus's tweets are related to the products and services of the mentioned high-tech companies. The language in these tweets used to express sentiment can be expected to be somewhat different from language used to express sentiment related to the stock market. This is not an ideal situation. It turns out that also unsupervised methods for sentiment analysis have been investigated. We will investigate the applicability of one such method using Semantic Orientation by Peter Turney [F].

Finally we will investigate how our efforts compare to professional offerings. One such service is Sentiment140. It allows to discover the sentiment of a brand, product, or topic on Twitter. Sentiment140 was created by computer science graduate students from Standford.

That means we will have three approaches in this capstone project that each have their own data, implementation and results, culminating in a comprehensive conclusion.

2. Twitter Search API

The Twitter Search API returns a collection of relevant tweets matching a specified query [L]. Usage requires an account on Twitter and registration of your app, which uses OAuth for authentication. We'll access the API via Python's Tweepy library. Response is in JSON format.

Here is an example of actual API access:

```
tweepy.Cursor(api.search,q='%23dowjones',since='2016-
11-23',until='2016-11-24').items()
```

Where '%23dowjones' stands for '#dowjones'. The API has some restrictions. It is not an exhaustive search of all tweets, rather a representative search. And limiting the search to a specific language, in this case English, was not successful.

3. Unsupervised Sentiment Analysis via Semantic Orientation

Semantic Orientation

The advantage of an unsupervised technique is that it does not require labeled data. The technique described here uses Semantic Orientation (SO) as described in Peter Turney's paper [F]. The method is laid out by Marco Bonzanini in his book 'Mastering Social Media Mining with Python' [H].

Semantic Orientation of a word is the difference between its association with positive and negative words. That is to say we want to calculate how close a word is to terms like 'good' and 'bad'. The measure for this closeness is Pointwise Mutual Information, or PMI for short. The original paper calculated PMI against the words 'excellent' and 'poor'. This vocabulary can be extended. Here we will use the opinion lexicon by Bing Liu [I]. This lexicon is a list of around 6800 words of English positive and negative opinion or sentiment.

Here is a sample of the lexicon's words:

```
positive_words = [
'a+',
'abound',
'abounds',
'abundance',
'abundant',
```

```
'accessable',
'accessible',
…

negative_words = [
'2-faced',
'2-faces',
'abnormal',
'abolish',
'abominable',
'abominably',
'abominate',
'abomination',
'abort',
```

To calculate PMI we need the probability of observing a particular term, and the probability of observing 2 terms together. Probabilities are calculated via term frequencies and term co-occurences in a document, where document is a tweet in this context.
Expressed in formulas, this is what we're calculating [H]:

Equation 1: Document Frequency DF

$$P(t) = \frac{\mathrm{DF}_{(t)}}{|D|}$$
$$P(t_1 \wedge t_2) = \frac{\mathrm{DF}_{(t_1 \wedge t_2)}}{|D|}$$

Equation 2: Pointwise Mutual Information PMI

$$\mathrm{PMI}(t_1, t_2) = \log\left(\frac{P(t_1 \wedge t_2)}{P(t_1) \cdot P(t_2)}\right)$$

Equation 3: Semantic Orientation SO

$$\mathrm{SO}(t) = \sum_{t' \in V^+} \mathrm{PMI}(t, t') - \sum_{t' \in V^-} \mathrm{PMI}(t, t')$$

So in summary Semantic Orientation is the difference of the sum of all positive PMIs and the sum of all negative PMIs. This will yield a floating point number. A positive sign indicates positive sentiment and a negative sign the opposite. A neutral sentiment would be a SO of 0. This is very unlikely to happen since we're summarizing many floating point numbers. Accordingly, with this approach we only look at 2 sentiments, positive and negative. If a zero occurs we will count it as positive.

Example from actual tweets

Here is an example from a tweet issued on 2016-11-02. The tweet had the following text:

"O/N #dowjones down 0.4% #ASX futures down 0.3% US appears more comfortable with Trump presidency or is more informed or refuses 2 believe :)"

The PMI for the term 'appears' and the other terms in this tweet is:

{u'trump': 8.679480099505447, u'presidency': 8.679480099505447, u'futures': 7.679480099505446, u'informed': 8.679480099505447, u'us': 7.679480099505446, u'comfortable': 8.679480099505447, u'n': 8.679480099505447, u'refuses': 8.679480099505447, u'believe': 8.679480099505447}

From there we find for positive word co-occurences that PMI('appears' ^ 'trump') = 8.679480099505447 and PMI('appears' ^ 'comfortable') = 8.679480099505447. The PMIs are summed up over all terms in a tweet.

Results for Dow Jones data

Finally we apply Semantic Orientation to the Dow Jones test data. We determine the SO for every tweet in a day and sum them up.

Here is the result:

	Date	gain	d1	simple	sc1
20	2016-11-02	-58.080078	-1.0	166.060717	False
19	2016-11-03	-48.080078	-1.0	404.780957	False
18	2016-11-04	-40.070312	-1.0	638.268249	False
17	2016-11-07	264.958984	1.0	341.378936	True
16	2016-11-08	81.359375	1.0	1115.193898	True
15	2016-11-09	272.429687	1.0	3176.102345	True
14	2016-11-10	204.740234	1.0	133.272472	True
13	2016-11-11	66.009765	1.0	247.038761	True
12	2016-11-14	-8.080078	-1.0	272.724391	False
11	2016-11-15	64.849609	1.0	962.345663	True
10	2016-11-16	-41.708984	-1.0	790.117311	False
9	2016-11-17	37.599609	1.0	397.586906	True
8	2016-11-18	-37.400390	-1.0	34.941611	False
7	2016-11-21	58.009765	1.0	-1.104000	False
6	2016-11-22	53.478516	1.0	1202.734479	True
5	2016-11-23	67.660157	1.0	231.070613	True
4	2016-11-25	58.419922	1.0	161.294869	True
3	2016-11-28	-24.240234	-1.0	142.067622	False
2	2016-11-29	57.529297	1.0	107.123092	True
1	2016-11-30	-12.060547	-1.0	511.229361	False
0	2016-12-01	42.730469	1.0	522.645770	True

One interesting observation is that the SO hardly ever turns negative. In the column 'd1', which shows the direction of the Dow Jones, we see several negative days. In particular the downtrend from 11-02 to 11-04 is missed. The uptrend is captured well during 11-08 to 11-09.

Overall pretty mixed results. Let's see if we can do better with Supervised Learning.

4. Supervised Sentiment Analysis with Sanders Corpus

Sanders Corpus

First we have to obtain the Sanders Corpus [E]. This turns out to be more challenging than expected because it is not a simple download. Due to copyright issues the tweets cannot be hosted by Sanders Analytics for download. Instead a Python script is provided which will use the Twitter API to download them one by one. There is a 6 second pause between each download. The whole process to download those ~5,500 tweets takes several hours. The result is a CSV file.

Here is a short excerpt of the corpus:

```
"Topic","Sentiment","TweetId","TweetDate","TweetText"
"apple","positive","126415614616154112","Tue Oct 18 21:53:25 +0000
2011","Now all @Apple has to do is get swype on the iphone and it will
be crack. Iphone that is"
"apple","positive","126402758403305474","Tue Oct 18 21:02:20 +0000
2011","Hilarious @youtube video - guy does a duet with @apple 's Siri.
Pretty much sums up the love affair! http://t.co/8ExbnQjY"
"apple","positive","126397179614068736","Tue Oct 18 20:40:10 +0000
2011","@RIM you made it too easy for me to switch to @Apple iPhone. See
ya!"
"apple","positive","126379685453119488","Tue Oct 18 19:30:39 +0000
2011","The 16 strangest things Siri has said so far. I am SOOO glad
that @Apple gave Siri a sense of humor! http://t.co/TWAeUDBp via
@HappyPlace"
```

We will find 4 different sentiments in the corpus with the following number of tweets: 'irrelevant' 1477, 'negative' 482, 'neutral' 2049, 'positive' 429.

TF-IDF vectorization

In the unsupervised section we used simple term frequencies. That's not an ideal measure because it does overweight some words that appear more frequently in general. This can be compensated by discounting words that appear in many documents, which is what TF-IDF does (TermFrequency-InverseDocumentFrequency) [M].

The SciKit-Learn feature extraction library has a TF-IDF vectorizer:

```
vectorizer = TfidfVectorizer(min_df= 2, max_df = 0.9,
sublinear_tf=True, use_idf=True)
```

Instead of simple term frequencies and term co-occurrences we will only use TF-IDF vectors going forward.

First Assessment

For the first assessment we will use several classifiers from Scikit_Learn Python library [K] and look at the respective F1 score for the different classes.

Here is the result:

```
F1 scores by class:    ['irrelevant' 'negative' 'neutral'    'positive']
F1 SVC(kernel=rbf):    [ 0.           0.          0.63174114  0.        ]
F1 SVC(kernel=linear):[ 0.88536155  0.51351351  0.79787234  0.31404959
F1 LinearSVC():        [ 0.87804878  0.52229299  0.78747204  0.42384106
F1 DecisionTree():     [ 0.78756477  0.3583815   0.71759259  0.3375    ]
F1 GaussianNB():       [ 0.84363636  0.4953271   0.66844208  0.36781609]
F1 MultinomialNB():    [ 0.87477314  0.5398773   0.77813505  0.31007752]
F1 BernoulliNB():        0.86643836  0.54545455  0.7497006   0.40251572]
```

What we can learn from this is that detection of sentiments 'irrelevant' and 'neutral' is strong. But detection of sentiments 'negative' and 'positive' is weaker. With the latter being the weakest.

For further analysis we will treat sentiments 'irrelevant' and 'neutral' the same.

Training 2 Classifiers

The Naïve Bayes classifier is well suited for this problem [J]. Specifically we will use the MultinomialNB classifier from the Scikit-Learn Python library [K]. This classifier assumes features to be occurrence counts, which is our case. In practice this classifier also works well with TF-IDF vectors [J].

Since the classification of 'positive' vs. rest and 'negative' vs. rest is relatively weak it might make sense to train 2 classifiers for our system. The first classifier would detect if a tweet has positive or negative sentiment vs. having no sentiment. The second classifier would then distinguish between positive and negative sentiment for those that have.

Pos vs. Neg classifier

We will look into the 2nd classifier first. If this one does not perform well we don't have a solution. First isolate tweets with positive and negative

sentiment. Split those tweets in a training and test set, assuring equal numbers of each sentiment in each set by using stratify option.

Here are the results for accuracy and the classification report:

```
*** Results for MultinomialNB() ***
Accuracy: 0.781420765027  (fraction of correctly classified samples)

  Classification report:
               precision    recall  f1-score   support
    negative       0.79      0.80      0.80        97
    positive       0.77      0.76      0.76        86
```

The precision/recall curve gives even more insight. The area under the curve (AUC) is a measure for the average precision Figure 2: P/R curve pos vs neg.

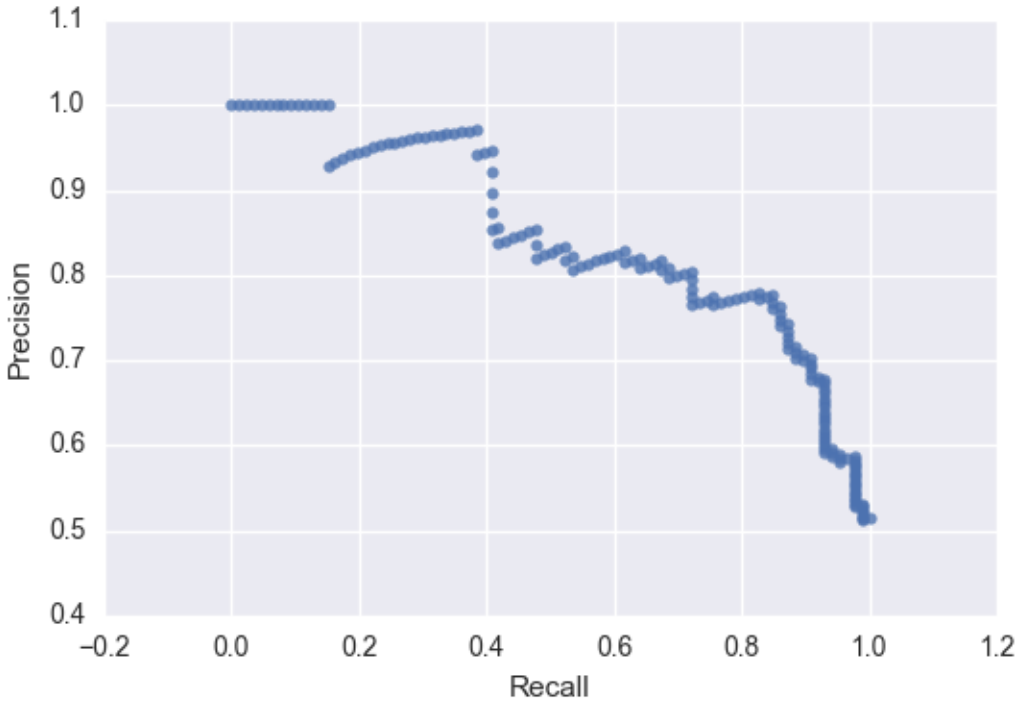


Figure 2: P/R curve pos vs neg

Next we run crossvalidation over 10 sets to get averages for accuracy and AUC (results graph in Appendix):

```
acc = accuracy (fraction of correctly classified samples)
auc = area under curve (average precision)

acc             auc
```

```
mean     std      mean     std
0.781   0.000   0.864   0.000
0.792   0.011   0.873   0.009
0.775   0.026   0.870   0.009
0.775   0.023   0.867   0.009
0.786   0.030   0.874   0.016
0.784   0.028   0.866   0.024
0.783   0.026   0.869   0.023
0.785   0.025   0.875   0.027
0.785   0.023   0.876   0.025
0.781   0.026   0.872   0.026

*** Positive vs. Negative MultinomialNB() ***
Final mean acc: 0.780821917808
Final mean auc: 0.872453456064
```

The results are very good. This is a strong classifier.

Pos/Neg vs. Irrelevant/Neutral classifier

This classifier will actually be the first in the system. It tells tweets with sentiment apart from tweets without sentiment.

Here are the summarized results:

```
*** Positive/Negative vs. Irrelevant/Neutral ***
Final mean acc: 0.797370892019
Final mean auc: 0.626440740388
```

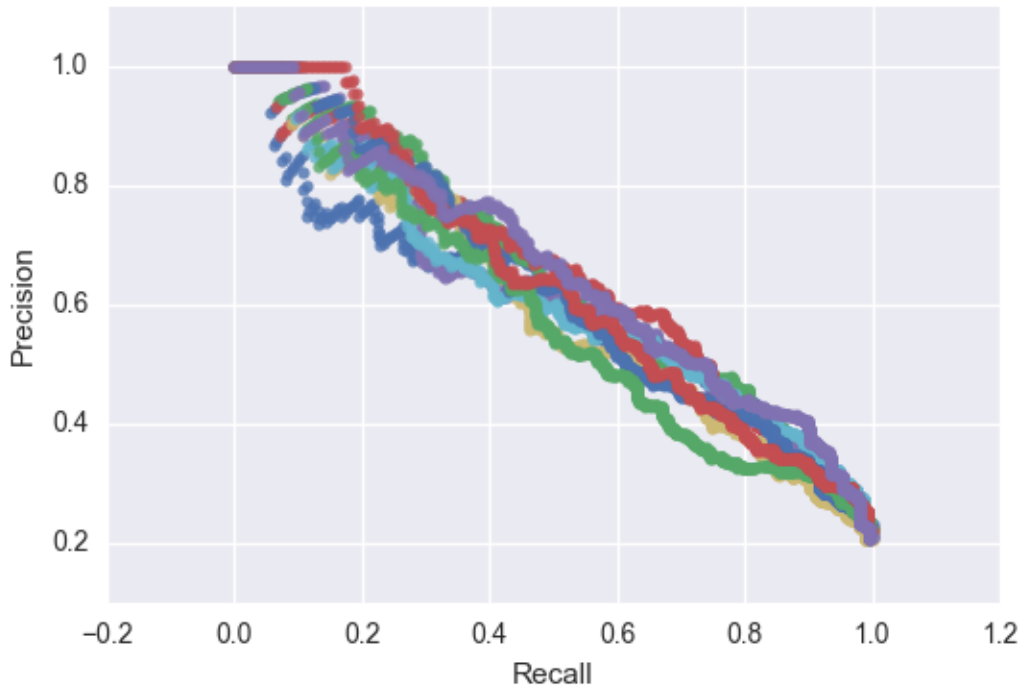


Figure 3: P/R curve pos/neg vs. irrelevant/neutral

The results are acceptable. This is a usable classifier. That means we expect a system comprised of both classifiers to perform reasonably well.

Hyperparameter search for both classifiers

The training so far has used default parameters for the TF-IDF vectorizer and the MultinomialNB. Before we actually use this system of classifiers we do hyper-parameter search to find the best parameters. This can be done via Scikit-Learn GridSearchCV function. We can combine the vectorizer and classifier into a single classifier with the help of a pipeline.

Here are the results for best parameters:

Best parameters classifier 1

```
Pipeline(steps=[
('vect', TfidfVectorizer(analyzer='word', binary=False, decode_error
=u'strict', dtype=<type 'numpy.int64'>, encoding=u'utf-8', input=u'c
ontent', lowercase=True, max_df=1.0, max_features=None, min_df=1, ng
ram_range=(1, 2), norm=u'l2', preprocessor=<function prep...e_idf=Fa
lse, vocabulary=None)),
('clf', MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True))]
)
```

Best parameters classifier 2

```
Pipeline(steps=[
('vect', TfidfVectorizer(analyzer='word', binary=False, decode_error
=u'strict', dtype=<type 'numpy.int64'>, encoding=u'utf-8', input=u'c
ontent', lowercase=True, max_df=1.0, max_features=None, min_df=1, ng
ram_range=(1, 1), norm=u'l2', preprocessor=<function prep...se_idf=T
rue, vocabulary=None)),
('clf', MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True))]
)
```

Sanders Corpus Classifier System Results

We are finally ready to classify the Dow Jones data with our 2 classifier system. Note that we are classifying a move as neutral when smaller than +20 or -20. Here are the results:

	Date	gain	d2	sanders	sc2	pos	neg	neu
20	2016-11-02	-58.080078	-1.0	-1	True	0	1	43
19	2016-11-03	-48.080078	-1.0	-1	True	0	1	96
18	2016-11-04	-40.070312	-1.0	0	False	1	1	90
17	2016-11-07	264.958984	1.0	-1	False	4	5	136
16	2016-11-08	81.359375	1.0	-1	False	1	2	211
15	2016-11-09	272.429687	1.0	-46	False	68	114	1829
14	2016-11-10	204.740234	1.0	-3	False	5	8	545
13	2016-11-11	66.009765	1.0	-2	False	0	2	31

```
12   2016-11-14   -8.080078  0.0       -2  False    0    2  197
11   2016-11-15   64.849609  1.0        1   True    2    1  185
10   2016-11-16  -41.708984 -1.0       -3   True    1    4  163
9    2016-11-17   37.599609  1.0        0  False    2    2  155
8    2016-11-18  -37.400390 -1.0        0  False    0    0  133
7    2016-11-21   58.009765  1.0       -3  False    0    3  122
6    2016-11-22   53.478516  1.0       -1  False   13   14  539
5    2016-11-23   67.660157  1.0       -3  False    0    3  342
4    2016-11-25   58.419922  1.0       -4  False    0    4  129
3    2016-11-28  -24.240234 -1.0       -1   True    0    1   87
2    2016-11-29   57.529297  1.0        0  False    0    0  108
1    2016-11-30  -12.060547  0.0        0   True    1    1   99
0    2016-12-01   42.730469  1.0        0  False    0    0  144

Score sanders:   6 (28.57%)
```

The score in the 'sanders' column is simply the difference of all positive and negative tweets per day. From that score we're only interested in the sign. The final score in the 'sc2' column shows whether our prediction matched the Dow Jones direction from column 'd2'. Every match will count towards the final score. In this case the score is only 6. That's a pretty weak result. If we were to pick always the same sentiment we would do better. Why is the score so poor despite good training results?

Tweaking Sanders Corpus Classifier System

Inspection of the classification results reveals that obviously positive stock market sentiment was either labeled neutral or in some cases even negative. Here are some examples: 'dowjones is up', 'dowjones hit high', 'record high', etc. We would like to see a positive sentiment here.

One reason for the misclassification is probably the fact that the Sanders corpus only contains tweets that are related to products. Those are the products of the companies Apple, Google, Microsoft and Twitter. A positive, or negative, sentiment for a product probably uses somewhat different language as the same sentiment expressed in the stock market.

Investigation of the Sanders corpus for positive language reveals that the most common terms are 'awesome', 'love' and 'great'. What we could try now is to translate the sentiment to better match the stock market.

Here is an example (see details in IPython notebook):

```
re.sub(r"dowjones (\w+\s)?up", "GREAT AWESOME LOVE", terms)
re.sub(r"dowjones (\w+\s)?hit(\w+)? (\w+\s)?high", "GREAT AWESOME
LOVE", terms)
re.sub(r"record high", "GREAT AWESOME LOVE", terms)
```

Results

	date	gain	d2	sanders	sc2	pos	neg	neu
20	2016-11-02	-58.080078	-1.0	-1	True	0	1	43
19	2016-11-03	-48.080078	-1.0	-1	True	0	1	96
18	2016-11-04	-40.070312	-1.0	0	False	1	1	90
17	2016-11-07	264.958984	1.0	2	True	7	5	133
16	2016-11-08	81.359375	1.0	0	False	2	2	210
15	2016-11-09	272.429687	1.0	12	True	121	109	1781
14	2016-11-10	204.740234	1.0	57	True	62	5	491
13	2016-11-11	66.009765	1.0	-2	False	0	2	31
12	2016-11-14	-8.080078	0.0	2	False	3	1	195
11	2016-11-15	64.849609	1.0	4	True	5	1	182
10	2016-11-16	-41.708984	-1.0	-2	True	2	4	162
9	2016-11-17	37.599609	1.0	2	True	3	1	155
8	2016-11-18	-37.400390	-1.0	0	False	0	0	133
7	2016-11-21	58.009765	1.0	-3	False	0	3	122
6	2016-11-22	53.478516	1.0	3	True	16	13	537
5	2016-11-23	67.660157	1.0	-2	False	1	3	341
4	2016-11-25	58.419922	1.0	-4	False	0	4	129
3	2016-11-28	-24.240234	-1.0	-1	True	0	1	87
2	2016-11-29	57.529297	1.0	0	False	0	0	108
1	2016-11-30	-12.060547	0.0	0	True	1	1	99
0	2016-12-01	42.730469	1.0	1	True	1	0	143

```
Score sanders:   12 (57.14%)
```

We now have a score of 12, or 57%. We wouldn't expect anything better than 79%, as mentioned in the introduction. So this is a very good score.

This simple tweak enhanced the score significantly. Without this tweak the only option would be to label our own data, which exceeds the scope of this capstone project.

5. Sentiment Analysis via Sentiment140 API

The final question we want to answer is how our own efforts compare to commercial offerings. One prominent candidate to try out is Sentiment140 [G]. They allow bulk classification of tweets via their API.

Results

And here are the results for the Dow Jones data:

	Date	gain	ml140	sc3	pos	neg	neu
20	2016-11-02	-58.080078	0	False	0	0	44
19	2016-11-03	-48.080078	0	False	1	1	95
18	2016-11-04	-40.070312	-1	True	3	4	85
17	2016-11-07	264.958984	5	True	7	2	136
16	2016-11-08	81.359375	6	True	8	2	204

```
15   2016-11-09  272.429687     33   True   121   88   1802
14   2016-11-10  204.740234     27   True    44   17    497
13   2016-11-11   66.009765      0  False     2    2     29
12   2016-11-14   -8.080078     14  False    15    1    183
11   2016-11-15   64.849609      5   True     8    3    177
10   2016-11-16  -41.708984      9  False    11    2    155
9    2016-11-17   37.599609     43   True    43    0    116
8    2016-11-18  -37.400390      3  False     3    0    130
7    2016-11-21   58.009765      1   True     1    0    124
6    2016-11-22   53.478516     16   True    24    8    534
5    2016-11-23   67.660157      5   True     5    0    340
4    2016-11-25   58.419922      4   True     4    0    129
3    2016-11-28  -24.240234     -1   True     0    1     87
2    2016-11-29   57.529297      0  False     0    0    108
1    2016-11-30  -12.060547      2  False     2    0     99
0    2016-12-01   42.730469      3   True     3    0    125

Score ML140:   13 (61.90%)
```

The score is slightly better than our own with 13 matches.

6. Conclusion

We have looked into an unsupervised method with weak results. We then have trained a classifier system on the Sanders corpus that achieves a score of 12, or 57%, on our Dow Jones benchmark. We wouldn't expect any system to be better than 79% [C]. A commercial offering, Sentiment140, is only slight better with a score of 13, or 62%.

This is a usable system.

7. Improvements

Ideally we would have labeled data for stock market tweets. Obtaining such data exceeds the scope of this capstone project.

The Twitter search API does not produce ideal data. We cannot filter for language. The Sanders corpus and Sentiment 140 only understand English and Spanish. All other language tweets are irrelevant.

Some tweets are also repeated often, causing a certain spamminess of the data. Those tweets are mostly trying to influence the market vs expressing a genuine opinion and should be filtered to some degree.

We have made no effort to take word types into account, like nouns, verbs, adjectives. Positive and negative tweets are more colorful than neutral ones and require more adjectives and words [J].

8. References

A. "Sentiment Analysis", Wikipedia, https://en.wikipedia.org/wiki/Sentiment_analysis
B. "The wisdom of the crowds", Wikipedia, https://en.wikipedia.org/wiki/The_Wisdom_of_Crowds
C. "How companies can use sentiment analysis to improve their business", by Maria Ogneva, Director of Social Media at Biz360., Apr 19 2010, http://mashable.com/2010/04/19/sentiment-analysis/#bIBaK4KTEkqB
D. Amazon Mechanical Turk, https://www.mturk.com/mturk/welcome
E. Sanders Twitter Sentiment Corpus, http://sananalytics.com/lab/twitter-sentiment/
F. Thumbs up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, Peter Turney, https://arxiv.org/abs/cs/0212032
G. Sentiment140, http://www.sentiment140.com/
H. "Mastering Social Media Mining with Python", Marco Bonzanini, https://www.amazon.com/Mastering-Social-Media-Mining-Python/dp/1783552018
I. Opinion Lexicon by Bing Liu, https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon
J. "Building Machine Leaning Systems with Python", Luis Pedro Coelho, Willi Richert.
K. Scikit-Learn Python library, http://scikit-learn.org/stable/index.html#
L. Twitter Search API, https://dev.twitter.com/rest/reference/get/search/tweets
M. TF-IDF Wikipedia, https://en.wikipedia.org/wiki/Tf%E2%80%93idf
N.

9. Appendix

Dow Jones prices in reference period

Table 1: Dow Jones prices in reference period

Time Period: Nov 02, 2016 - Dec 01, 2016 Show: Historical Prices Frequency: Daily Apply

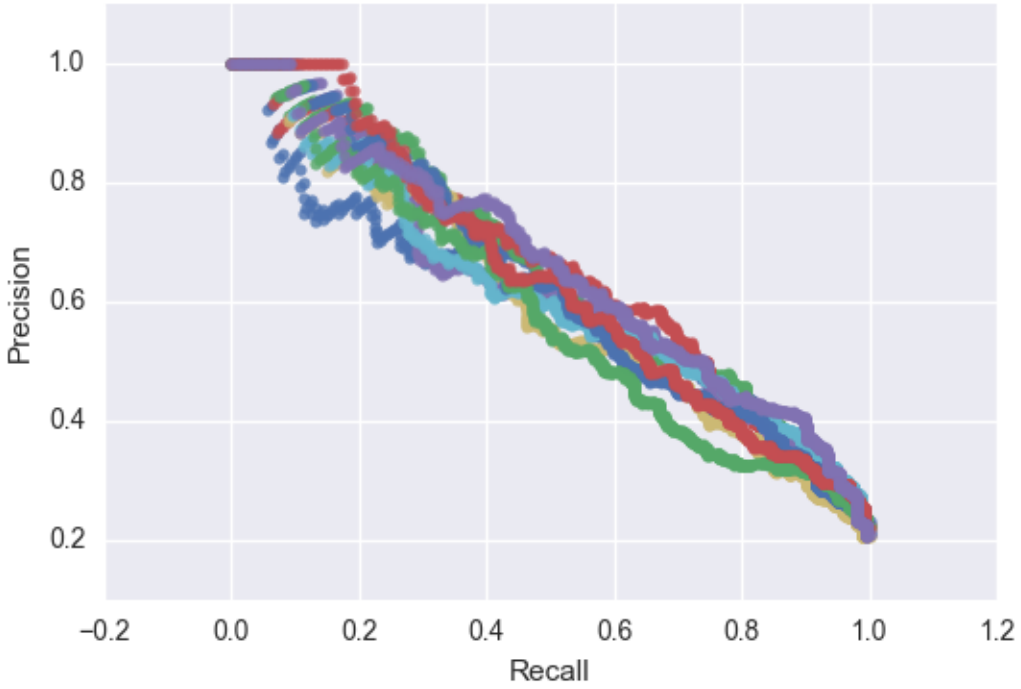
Currency in USD. Download Data

Date	Open	High	Low	Close	Adj Close*	Volume
Dec 01, 2016	19,149.20	19,214.30	19,138.79	19,191.93	19,191.93	108,800,000
Nov 30, 2016	19,135.64	19,225.29	19,123.38	19,123.58	19,123.58	164,570,000
Nov 29, 2016	19,064.07	19,144.40	19,062.22	19,121.60	19,121.60	81,510,000
Nov 28, 2016	19,122.14	19,138.72	19,072.25	19,097.90	19,097.90	88,460,000
Nov 25, 2016	19,093.72	19,152.14	19,093.72	19,152.14	19,152.14	45,890,000
Nov 23, 2016	19,015.52	19,083.76	19,000.38	19,083.18	19,083.18	77,880,000
Nov 22, 2016	18,970.39	19,043.90	18,962.82	19,023.87	19,023.87	85,310,000
Nov 21, 2016	18,898.68	18,960.76	18,883.10	18,956.69	18,956.69	80,520,000
Nov 18, 2016	18,905.33	18,915.74	18,853.83	18,867.93	18,867.93	109,880,000
Nov 17, 2016	18,866.22	18,904.03	18,845.27	18,903.82	18,903.82	89,940,000
Nov 16, 2016	18,909.85	18,909.85	18,825.89	18,868.14	18,868.14	87,320,000
Nov 15, 2016	18,858.21	18,925.26	18,806.06	18,923.06	18,923.06	100,660,000
Nov 14, 2016	18,876.77	18,934.05	18,815.75	18,868.69	18,868.69	112,250,000
Nov 11, 2016	18,781.65	18,855.78	18,736.96	18,847.66	18,847.66	107,300,000
Nov 10, 2016	18,603.14	18,873.66	18,603.14	18,807.88	18,807.88	164,390,000
Nov 09, 2016	18,317.26	18,650.06	18,252.55	18,589.69	18,589.69	173,110,000
Nov 08, 2016	18,251.38	18,400.50	18,200.75	18,332.74	18,332.74	79,820,000
Nov 07, 2016	17,994.64	18,263.30	17,994.64	18,259.60	18,259.60	93,450,000
Nov 04, 2016	17,928.35	17,986.76	17,883.56	17,888.28	17,888.28	97,760,000
Nov 03, 2016	17,978.75	18,006.96	17,904.07	17,930.67	17,930.67	77,860,000
Nov 02, 2016	18,017.72	18,044.15	17,931.89	17,959.64	17,959.64	88,610,000

*Close price adjusted for dividends and splits.

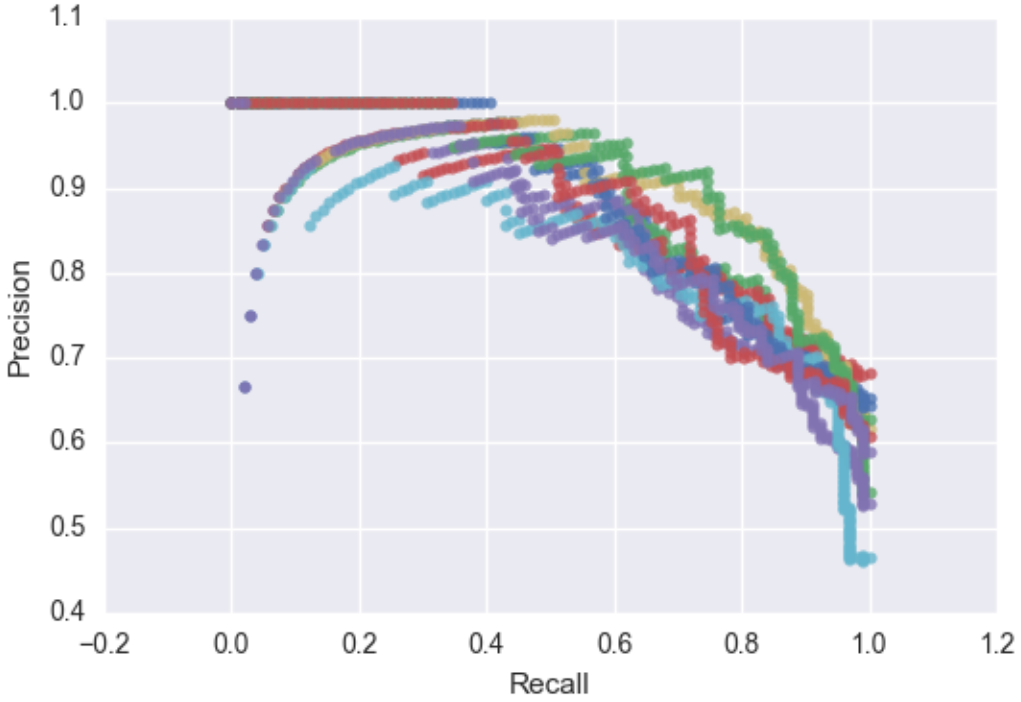
Cross-validation for classifier 1

```
*** Positive/Negative vs. Irrelevant/Neutral ***
Final mean acc: 0.797370892019
Final mean auc: 0.626440740388
```



Cross-validation for classifier 2

```
*** Positive vs. Negative MultinomialNB() ***
Final mean acc: 0.780821917808
Final mean auc: 0.872453456064
```


1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Precision
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Recall