Daily News for Stock Price Analysis

Motivation:

Stock market is one of the most fascinating market in the financial world. The movement of stock price can be influenced by various factors. Among these factors, the behavior of human investors can play a significant role. It is impossible to know the investors' thoughts. However, we can use financial news articles to extract some messages since these news can reflect the human reaction to the market and stocks.

Our project is an attempt to predict stock movement using daily news. To simplify the problem, we choose S&P 500, the most accurate gauge of the performance of American equities, as the representative of the stock market performance.

Background:

There exists some lag between when the market has movement and when financial news is released. However, according to Gidofalvi, Gyozo 's research, there is a relationship between stock price movement and financial news. In his experiment, he classified price movement as "up," "down," or (approximately) "unchanged" relative to the volatility of the stock and the change in a relevant index and used a trained naïve Bayesian text classifier to predict which movement class a piece of financial news belongs to. Through this way, he found the correlations and showed that how short-term (about 20 minutes) stock price movements can be predicted using financial news.

In our project, we choose different kinds of models to analyze daily news and use the result to predict the stock closing price movement of that day.

Data Sources:

1. Description of data sources

S&P 500 price: We download Adjusted close price of S&P 500 from Yahoo Finance with the date range from 01/01/2013 to 10/07/2017. We compare the close price of each day with the previous day and use 1 to represent up or equal, use 0 to represent down.

Daily news: We use BeautifulSoup to scrape tickers of Top 100 weighted companies in S&P 500 from slickcharts.com and daily news of these companies from Thomson Reuters company news pages (e.g. https://www.reuters.com/finance/stocks/company-news/AAPL.O?date=08182016). For each company, we scrape the top story of the day because it is the most influential news of that day selected by Reuters.

2. Descriptive statistics of the data

Raw data:

	Data Type	Number
S&P 500	[int] 0 for down, 1 for up and no change	1200*1
Daily news	[strings]	1200*100

After data cleaning, we get a list of words for each day. The number of words varies from tens to hundreds.

1062014	update	samsung	electronics	guidance	widely	misses	street	estimates	xbox	one	sales	top
1072014	ford	ceo	going	microsoft	stay	ford	analysis	social	media	empowers	anti-mining	activists
1082014	micron	shares	rise	strong	results	possible	apple	contract	microsoft	succession	speculation	focuses
1092014	apple	samsung	ceos	agree	mediation	u.s	patent	fight	microsoft	succession	speculation	focuses
1102014	apple	violate	google	patent	says	u.s	appeals	court	column-franc	says	non	digital
1132014	apple	loses	bid	block	antitrust	monitorship	chevron	gives	green	light	britain	north
1142014	apple	loses	court	bid	block	e-book	antitrust	monitor	woman	rescued	nevada	motel
1152014	apple	refund	least	million	disputed	kids	app	purchases	woman	rescued	nevada	motel
1162014	apple	china	mobile	launch	could	spark	costly	subsidy	war	microsoft	considers	ericsson
1172014	nextel	offer	latest	iphones	brazil	nintendo	heads	third	consecutive	annual	loss	wii
1212014	apple	gets	reprieve	e-book	monitor	oversight	analysis	verizon	deal	may	push	u.s
1222014	icahn	blasts	apple	boosts	position	billion	watch	day	ahead	thursday	jan	facebook
1232014	samsung	electronics	fourth-quart	profit	sags	smartphone	growth	concern	deepens	microsoft	profit	beats
1242014	apple	set	report	record	holiday	china	doubts	stay	us	stocks-wall	st	open
1272014	update	u.s	frees	tech	companies	give	spying	data	ukraine	postpones	signing	gas
1282014	us	stocks-wall	st	rebounds	futures	fly	turkey	rate	hike	facebook	cites	progress
1292014	lenovo	buy	google	motorola	china	largest	tech	deal	ericsson	ceo	vestberg	tells
1302014	apple	samsung	spar	potential	u.s	ban	smartphone	sales	microsoft	board	preparing	name
1312014	nadella	outran	better-know	candidates	microsoft	ceo	big	chill	gives	dow	worst	month
2032014	web	companies	give	first	look	secret	government	data	requests	bank	america	hikes
2042014	apple	u.s	clash	court	e-books	antitrust	monitor	microsoft	hopes	new	old	leaders
2052014	apple	grim	history	buybacks	microsoft	hopes	new	old	leaders	rekindle	magic	exclusive
2062014	taiwan	china	plays	rise	tech	shares	amazon	asks	customers	pick	tv	pilots
2072014	us	stocks-wall	st	rallies	snap	three-week	skid	column-the	year-old	microsoft	memo	came
2102014	icahn	gives	apple	buyback	plan	iss	urges	vote	nokia	unveil	low-cost	android
2112014	sony	talks	supply	camera	sensors	apple	nikkei	bofa	says	boost	profitability	wealth
2122014	doubleline	gundlach	says	firm	sold	apple	shares	cnbc	nokia	renews	legal	battle

Methodology

- Features we use

We regard the words in the news as the initial features, then it is necessary to do text cleaning and feature engineering to make these words more efficient use in the following analysis.

Input: As we focus on analyzing the text of the news, all input of these methods are the cleaned data after our explorative analysis on text.

1. Classification

Our data is labelled, meaning it is assigned a class, for example the stock price result is up or down. The decision being modelled is to assign labels to new unlabelled pieces of data. This can be thought of as a discrimination problem, modelling the differences or similarities between groups.

As most of research shows that the SVM, NB, Random Forest have the relative better performance on classification, we decide to use these three algorithms and as the output can be accuracy, precision, recall, F1_score...which can be used to evaluate the model and compare the fitting level of these models.

Input: Document term matrix (dtm) generated from TfidfVectorizer and the movement of every work day.

OutPut: Binary classification, like 1 means stock price goes up while 0 means price goes down.

Performance evaluation: We use 5-fold cross validation, and calculate F1-score and accuracy to measure performance.

1) Naive Bayes Classification:

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector $x_1$through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive independence assumption that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set.

Naive Bayes actually gets three classifiers as MultinomialNB, BernoulliNB and GaussianNB. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$. In our project, we choose the MultinomialNB and BernoulliNB classifiers because our input data is dtm(the tfidf matrix) and GaussianNB classifier requires the form of input data to be array. Different kinds of input data may cause inaccuracy in further comparison; thus, we decided not to user GaussianNB classifier.

2) Support Vector Machines:

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

The advantages of support vector machines are:
- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of

samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

Here we choose Linear Support Vector Classifier. Linear SVC implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

3) Decision Tree- Random Forest:

decision tree: the entire data set of observations is recursively partitioned into subsets along different branches so that similar observations are grouped together at the terminal leaves. Decision trees are unstable. Small changes in training data can lead to wildly different trees. Training a group of trees and “ensembling” them to create a “random forest” model enables more robust prediction.

To eliminate this unstable, we add bagging(sampling with replacement) method by using decision tree as base estimators.

The construction procedures of forest is to randomly select a subspace of features at each node to grow branches of a decision tree, then to use bagging method to generate training data subsets for building individual trees, finally to combine all individual trees to form random forests model.

Here we use GridsearchCV function to calibrate parameters in the RandomForestClassifier function to make sure the best performance in the model.

2. K-Means Clustering

K-means clustering will group similar observations into clusters. The basic idea of K Means clustering is to form K seeds first, and then group observations in K clusters on the basis of distance with each of K seeds. The observation will be included in the nth seed/cluster if the distance between the observation and the nth seed is minimum when compared to other seeds.

The reason why we try with K-Means, an unsupervised learning method, is that some of the words represent the economic information which will have an influence on the movement of stock price. For example, the stock price will move down after the issue of stock dividend.

Input: Document term matrix (dtm) generated from TfidfVectorizer.

Output: 2 clusters. This is just a general clusters and to find out the characteristics of the cluster we need to list the top word in this cluster.

Performance evaluation: precision, recall, f1-score.

3. Deep Learning (CNN)

Prediction Model

Intuitively, the relationship between events and the stock market may be more complex than linear, due to hidden and indirect relationships. We exploit a deep neural network model, the hidden layers of which is useful for learning such hidden relationship.

A Recurrent Neural Network (RNN) is a more 'natural' approach, given that text is naturally sequential. However, RNNs are quite slow and fickle to train. For tasks where feature detection in text is more important, for example, searching for angry terms, sadness, abuses, named entities even. It is a better choice to build the mode base on CNN.

Our mission comprises how many layers should be set and how to choose the parameters among various choices. Moreover, we are really interested in combining the newest technical application on Neural Network with our research, it must be a big breakthrough for us students.

Performance evaluation:

Here I use Tensorflow to conduct the CNN experiment to predict the stock price, which is developed by the Google Brain Team for the purposes of conducting machine learning and deep neural networks research.

This part contains a Binary classification task. Hence, there are only two labels:

"1" when S&P 500 Close value rised or stayed as the same;

"0" when S&P 500 Close value decreased.

Thus the input only contains binary label and tokenized text and then we split by 90% and 10%.

The performance evaluation under the model shows the final accuracy on testing set is reaching up to about 55%.

Experiment results and analysis:

1. Naive Bayes

Cross validation result for MultinomialNB:

```
Test data set average fscore using MultinomialNB model:
[ 0.38907343  0.3866021   0.43570451  0.43832528  0.4013951 ]

Test data set average accuracy using MultinomialNB model:
[ 0.5186722   0.525       0.54166667  0.54583333  0.55230126]
```

Cross validation result for BernoulliNB:

```
Test data set average fscore using BernoulliNB model:
[ 0.53115306  0.49071845  0.5137335   0.53252174  0.49053292]

Test data set average accuracy using BernoulliNB model:
[ 0.53941909  0.50416667  0.52083333  0.53333333  0.50209205]
```

Comparing the two naive bayes classifiers, we can find that although both of the results are not satisfied enough(explained in the conclusion part), the f-score of BernoulliNB is still higher than that of MultinomialNB. We assume that this is because that BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Since we have fitted the classifiers with dtm and target and used the “Up or Down” to judge, BernoulliNB is more suitable in this case and could get higher fscore.

2. SVM

We use 5-fold cross validation and use F1-score and accuracy as metrics to measure performance.

```
Test data set average fscore:
[ 0.50345413  0.4978451   0.53652638  0.55285473  0.5221808 ]

Test data set average accuracy:
[ 0.50622407  0.5125      0.5375      0.55416667  0.53556485]
```

The highest accuracy is 55%. Comparing with Luss and d'Aspremont's research[2012] using bag-of-words feature and SVM model, achieved 56.42% accuracy, our model gets a relatively reasonable result. There is some room to improve our model, and the methods will be explained in detail in the Conclusion and Future Work part.

3. Random Forest

Number of trees grid search results(calibrate parameter 'n_esitimator'):

```
[mean: 0.51896, std: 0.01051, params: {'n_estimators': 10},
 mean: 0.51002, std: 0.01593, params: {'n_estimators': 20},
 mean: 0.50974, std: 0.02031, params: {'n_estimators': 30},
 mean: 0.49899, std: 0.01619, params: {'n_estimators': 40},
 mean: 0.50741, std: 0.02719, params: {'n_estimators': 50},
 mean: 0.51249, std: 0.02566, params: {'n_estimators': 60},
 mean: 0.50878, std: 0.02593, params: {'n_estimators': 70},
 mean: 0.50798, std: 0.02143, params: {'n_estimators': 80},
 mean: 0.50551, std: 0.02176, params: {'n_estimators': 90},
 mean: 0.49860, std: 0.02028, params: {'n_estimators': 100},
 mean: 0.49983, std: 0.01816, params: {'n_estimators': 110},
 mean: 0.49899, std: 0.01906, params: {'n_estimators': 120},
 mean: 0.49997, std: 0.01687, params: {'n_estimators': 130},
 mean: 0.49844, std: 0.01971, params: {'n_estimators': 140},
 mean: 0.49553, std: 0.01928, params: {'n_estimators': 150},
 mean: 0.49781, std: 0.01905, params: {'n_estimators': 160},
 mean: 0.49803, std: 0.01871, params: {'n_estimators': 170},
 mean: 0.49965, std: 0.01899, params: {'n_estimators': 180},
 mean: 0.50050, std: 0.02230, params: {'n_estimators': 190}]
```

Cross validation result:

```
({'n_estimators': 10}, 0.5189567640205817)

Test data set average fscore:
[ 0.47171928  0.47467167  0.58749284  0.55415893  0.51443183]
```

The highest accuracy is 58.7%, some terms or features in text data which are uninformative a class. During this forest building process, topic-related or informative features would have the large chance to be missed, if we randomly select a small subspace from high dimensional text data. As a result, weak trees will be created from these subspaces, the average discriminative strength of those trees in reduced and the error bound of the random forest is enlarged. Therefore, the forest has a large likelihood to make a wrong decision which mainly results from those “weak” trees’ classification power.

4. K-means clustering

We use K-Means to group daily news into two groups and get the top 20 words of each cluster.

```
Cluster 0: brief; announces; agreement; dividend; file; share; mln; note; fargo; offer; boeing; amazon; new; pct; report; ceo;
board; apple; pfizer; quarterly
Cluster 1: deal; apple; goldman; gm; morgan; new; sale; stocks; stanley; bank; boeing; wall; source; billion; drug; million; ce
o; jpmorgan; brief; china
```

When we check the top 20 words, it is hard to tell which cluster should represent up or down. So I compare the predicted clusters with the actual clusters to see which one can lead to a better performance.

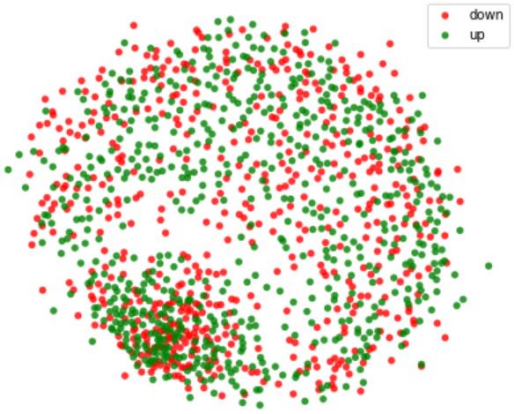
The performance and confusion matrix are as follows:

	precision	recall	f1-score	support
0	0.46	0.32	0.38	549
1	0.54	0.69	0.61	651
avg / total	0.51	0.52	0.50	1200

predict	actual	
0	0	175
	1	204
1	0	374
	1	447

dtype: int64

The performance is only about 0.50. The unsatisfied performance is because of the K-Means works better on a data distribution possessing hyper-ellipsoidal shape. And if we plot the actual clusters (shown below) we can find the two clusters blends very well and have little differences. It is hard to group the news in two clusters.



5. CNN

As this is the first time we explore the neural network and apply this magic method to our project. We suppose that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Here we first apply the approach learned from the lectures. With different dataset, it is necessary to find the parameters according to the Practical Recommendations for Gradient-Based Training of Deep Architectures. The following is some useful techniques to set.

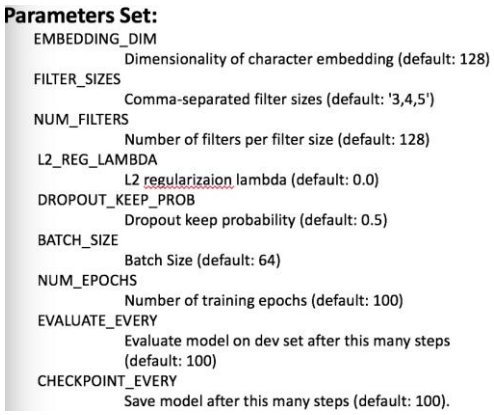
By drawing the relevant graph, the best MAX_NB_WORDS and MAX_DOC_LEN is 1330 and 200 respectively. It is significant that applying these fantastic parameters can get a better result. The evaluation performance can reach up to about 85%.

It’s indeed a good result, however, compared with other prediction models, we thought it must be an unreliable result with a too high accuracy. Thus we build a baseline model which consider all hyperparameters. For the data processing stage, we did not remove the stop words as it may influence the static part, and also not change the sequence of the text.

We give a baseline CNN configuration described in Table below. We argue that it is critical to assess the variance due strictly to the parameter estimation procedure. Most prior related research, unfortunately, has not reported such variance, despite a highly stochastic learning procedure. This variance is attributable to estimation via SGD, random dropout, and random weight parameter initialization.

Description	Values
input word vectors	Google word2vec
filter region size	(3,4,5)
feature maps	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
*l*2 norm constraint	3

With the base hyperparameters, we create the CNN model and train the data based on the model.



Parameters Set:

EMBEDDING_DIM
Dimensionality of character embedding (default: 128)
FILTER_SIZES
Comma-separated filter sizes (default: '3,4,5')
NUM_FILTERS
Number of filters per filter size (default: 128)
L2_REG_LAMBDA
L2 regularizaion lambda (default: 0.0)
DROPOUT_KEEP_PROB
Dropout keep probability (default: 0.5)
BATCH_SIZE
Batch Size (default: 64)
NUM_EPOCHS
Number of training epochs (default: 100)
EVALUATE_EVERY
Evaluate model on dev set after this many steps (default: 100)
CHECKPOINT_EVERY
Save model after this many steps (default: 100).

The regularization parameter (lambda) is an input to your model so what you probably want to know is how do you *select* the value of lambda. The regularization parameter reduces overfitting, which reduces the variance of your estimated regression parameters; however, it does this at the expense of adding bias to your estimate. Increasing lambda results in less overfitting but also greater bias. So the real question is "**How much bias are you willing to tolerate in your estimate**?"

One approach we take is to randomly subsample our data a number of times and look at the variation in our estimate. Then repeat the process for a slightly larger value of lambda to see how it affects the variability of our estimate. Keep in mind that whatever value of lambda we decide is appropriate for our subsampled data, we finally can use a smaller value to achieve comparable regularization on the full data set.

```
2017-12-07T12:35:28.721837: step 85, loss 0.771742, acc 0.714286
2017-12-07T12:35:28.903375: step 86, loss 0.872056, acc 0.75
2017-12-07T12:35:29.086691: step 87, loss 0.672853, acc 0.765625
2017-12-07T12:35:29.268255: step 88, loss 0.80608, acc 0.734375
2017-12-07T12:35:29.470168: step 89, loss 0.578473, acc 0.734375
2017-12-07T12:35:29.648310: step 90, loss 0.691411, acc 0.78125
2017-12-07T12:35:29.830751: step 91, loss 0.798413, acc 0.71875
2017-12-07T12:35:30.015499: step 92, loss 0.699149, acc 0.796875
2017-12-07T12:35:30.199039: step 93, loss 0.71896, acc 0.75
2017-12-07T12:35:30.386141: step 94, loss 0.644855, acc 0.84375
2017-12-07T12:35:30.570588: step 95, loss 0.7998, acc 0.8125
2017-12-07T12:35:30.748988: step 96, loss 0.676583, acc 0.796875
2017-12-07T12:35:30.927476: step 97, loss 1.02928, acc 0.6875
2017-12-07T12:35:31.110853: step 98, loss 0.521752, acc 0.796875
2017-12-07T12:35:31.289122: step 99, loss 0.897805, acc 0.6875
2017-12-07T12:35:31.487651: step 100, loss 0.877194, acc 0.75

Evaluation:
2017-12-07T12:35:31.582195: step 100, loss 0.821526, acc 0.55
```

As we considered that the evaluate step with 100 can make an overtraining situation. After several trials, we change the evaluate steps to 10 steps. And the training process will terminate as soon as the accuracy is good to make the best model.

```
2017-12-08T11:52:27.527198: step 11, loss 2.4259, acc 0.453125
2017-12-08T11:52:30.652742: step 12, loss 2.45375, acc 0.53125
2017-12-08T11:52:33.845183: step 13, loss 2.70186, acc 0.46875
2017-12-08T11:52:37.089941: step 14, loss 2.37228, acc 0.484375
2017-12-08T11:52:40.669839: step 15, loss 3.15698, acc 0.375
2017-12-08T11:52:43.576151: step 16, loss 2.71699, acc 0.421875
2017-12-08T11:52:46.106036: step 17, loss 2.1784, acc 0.571429
2017-12-08T11:52:49.063781: step 18, loss 2.03957, acc 0.546875
2017-12-08T11:52:52.317395: step 19, loss 2.10115, acc 0.53125
2017-12-08T11:52:56.073714: step 20, loss 2.13183, acc 0.609375

Evaluation:
2017-12-08T11:52:58.306029: step 20, loss 0.934871, acc 0.55

Saved model checkpoint to /Users/Dido/runs/1512751903/checkpoints/model-20
```

Finally, we got the accuracy which is about 55%. It is an unexpected result. First of all, our classifier seems not do better than a random choice, there is a risk that there simply is no connection between features and class. A good question to ask in such a position, is weather we or a domain expert could infer the class (with an accuracy greater than a random classifier) based on given features. If no, then getting more data rows or

changing the classifier won't help. What we need to do is get more data using different features.

On the other hand, we think the information needed to infer the class is already on the labels, we should check whether our classifier suffers from a high bias or high variance problem.

To do this, graph the validation error and training set error, as a function of training examples.

If the lines seem to converge to the same value and are close at the end, then our classifier has high bias and adding more data won't help. A good idea in this case is to either change the classifier for a one that has higher variance, or simply lower the regularization parameter of our current one.

If on the other hand the lines are quite far apart, and we have a low training set error but high validation error, then our classifier has too high variance. In this case getting more data is very likely to help. If after getting more data the variance will still be too high, we can increase the regularization parameter.

Conclusion and future work：

Model	f1-score
Naive Bayes	53.9%
SVM	55.4%
Random Forest	58.7%
K-Means	50%
CNN	55%

Comparing with Luss and d'Aspremont's research[2012] using SVM model achieving 56.42% accuracy, and Ding's research[2015] using CNN, with 61.73% accuracy, there's some room for improvement of our project. Possible reasons are:

- Data volume is not enough. For some days, some companies have no news. Most companies have one or two pieces of news each week.

- Headlines just convey the fact not attitude. We performed sentiment analysis on news headlines, and most of them are neutral. It is because news headlines contain a lot of nouns, such as company names, which has no sentiment.
- Tf-idf features cannot capture exactly what happens in news headlines. As Tf-idf features measure word frequency, it cannot process the text from semantic aspect. Analyzing by single words is not accurate, because we cannot know the relations among words. Extracting event tuples from news headlines may work better.
- Company news are not weighted. We take news of Top 100 companies in S&P 500, and the companies are of the same weight. However, in S&P 500, companies have different weights, for example, weight of AAPL is 3.87%. So news of companies with larger weights should have larger influence on price movement. So giving news from top companies larger weights, and other companies smaller weights will probably improve the performance.

For our future work, we can improve from two aspects:

1. Data

   - To increase the data volume, we are going to expand the time range from 5 years to 10 years.

   - Get related news of more companies. For the companies where availability of financial news is a challenge, we will use Twitter message.

   - In the model, we use daily news to predict the daily movement of the adjusted close price in the same day. Maybe we can try to use daily movement of the adjusted open price in the next day or use several days' news, like a week, to predict the daily movement.

2. Approach

   - Use word embeddings/event embeddings instead of Tf-idf features. Word embedding process maps words from vocabulary to vectors in a high dimension space. Relevant words have similar vectors, so news with similar meanings can be modelled similar in the space.

   Ding et al. [2015] reported using event embeddings as features when predicting stock market movement with CNN model. In his work, an event is represented as a tuple E = (O_1, P, O_2), where O_1 is Actor, P is Action, and O_2 is Object. For example, the event "Jan 13, 2014 - Google Acquires Smart Thermostat Maker Nest For for $3.2 billion." is modeled as: (Actor = Google , Action = acquires ,

Object = Nest). Then, they proposed a novel neural tensor network for learning event embeddings. This method could capture more structured information in news headlines, and would improve the accuracy.

- Give weightings to each company, so that news of top companies such as Apple and Microsoft, have larger weights and influences more to the prediction result.

- Test other feature weighting methods for optimizing the random sampling subspace used in random forest.

Team Members:
Qianhui Li
Hangyu Li
Han Wu
Yuchen Xie
Jingting Zhang
Dec. 8th, 2017