Representation Learning and Text Mining for Stock Market Prediction:

George Fei, Leo Laugier, Scott Johnson, and Shikun Ding

Final Capstone Report

Capstone Advisor: Prof. Laurent El Ghaoui

04/30/2018

We affirm that we are the sole authors of this report and we give due credit (*i.e.*, use correct citations) to all used sources.

Leo Laugier *April 30, 2018*

George Fei *April 30, 2018*

Shikun Ding *April 30, 2018*

Scott Johnson *April 30, 2018*

Abstract

Nowadays, social networks are mines of information for revealing people's opinions, from their political views to their thoughts regarding a company or a product. In addition, online journalism produces data in natural English describing and analyzing the market. Therefore, we decided to design and build deep learning and classic machine learning models for predictive tasks useful for financial analysts. Particularly, this capstone project focuses on mining opinion from news headlines to predict stock market trends by using various techniques, including representation learning, sentiment analysis, and keyword extraction. Whereas current methods often rely on human-engineered features, we successfully built a Convolutional Neural Network (CNN) combined with a Support Vector Machine (SVM) to extract meaningful positive/negative keywords and perform accurate predictions of the stock price trends, for specific markets.

Introduction

The boom of web services such as social networks and digital journalism has generated "Big Data" that companies and organizations can process in order to analyze public opinion and journalists' assessments on market.

Moreover, recent progress in deep learning (LeCun *et al.*, 2015) has incentivized data scientists to create semantic representations used in new Representation Learning[1] models (Bengio *et al.*, 2013) for Natural Language Processing. Natural Language Processing (NLP) is the intersection between computer science, linguistics, and mathematics which studies the automatization of human language through computer systems. As stated in (Tractica, "Natural Language Processing Market", 2017), "the market of NLP is expected to reach $22.3 billion by 2025" and according to Allied Market Research, the market for text analysis will be $6.5 billion in 2020 (Singh, 2015). Moreover, the annual growth rate would be more than 20% (Singh, 2015). The goal of this capstone project is to use deep neural networks and new optimization methods to process natural language and classify sentiments in the text in order to predict the stock market.

[1] consists in "learning representations of the data that make it easier to extract useful information when building classifiers or other predictors" (Bengio *et al.*, 2013)

Engineering Leadership

Data-driven companies are much more likely to be profitable

To start with, we found the management consulting company Gallup, Inc., whose goal is to deliver "analytics and advice to help leaders and organizations solve their most pressing problems" (“Pressing problems,” n.d.). For each survey they make, Gallup interviews one thousand people and uses human intelligence to produce a performance-management report. Indeed, companies need Data Analytics to perform. According to a recent report by EY, 81% of organizations believe data should be at the heart of every decision they make (“EY,” 2015). In fact, according to Gallup, “data-driven organizations are 23 times more likely to acquire customers, six times more likely to retain those customers, and 19 times more likely to be profitable than non-data driven organizations” (“Pressing problems,” n.d.). If data-driven companies are much more likely to be profitable, how can they leverage text data found on social networks?

Services and products companies will benefit from processing natural language data from social networks to mine people’s opinions

Online services companies (e.g. Amazon for shopping, Uber for transportation, etc.), business ratings and reviews websites such as Yelp store many valuable service/product reviews from customers that are a gold mine for companies. On the one hand, potential customers pay attention to these reviews before purchasing a service or a product. On the other hand, companies take into account relevant comments on particular features in order to improve the current project

or prepare the next release. Now, think of all the “tweets” and Facebook posts that convey underlying panorama of the society. What if one could mine people’s opinion expressed through social networks? Actually, this information can be used to analyze social trends (e.g. customers’ ratings on new products that have just been released) to quickly figure out how people react to news, releases or events. So companies will benefit from people’s opinions on social networks; how can these social media influence markets?

Furthermore, social networks store valuable text data for market analysts

How did a single tweet shake Wall Street on February 21, 2018? Actually, Kylie Jenner, Kim Kardashian’s sister, wreaked havoc on US business media by tweeting “sooo[*sic*] does anyone else not open Snapchat anymore? Or is it just me... ugh this is so sad”. Soon after this message was posted on Twitter, Snapchat lost 1.3 billion dollars of market capitalization (Figure 1: Kylie Jenner’s tweet was posted between the closing and the opening positions. We observe a drop of 2.3% in the meantime - see 2/22 on the x-axis). A few moments later Mrs. Jenner understood the impact of her tweet and set it right with a second one: “still love you tho snap ... my first love”. Furthermore, on August 3, 2016 shares of Blackberry rose slightly after Kim Kardashian asked on Twitter if she should try a new Blackberry device. Thus, influential people on social networks are powerful enough to either reassure or alert investors. In addition to social media, news headlines represent data that can be processed to predict market trends.

Figure 1: Snap Inc. (SNAP) NYSE - Nasdaq Real Time Price. Currency in USD from Yahoo! Finance.

Another mine of information: news media provide information to monitor, describe, analyze and predict market health and companies shares.

Figure 2: Organizational pattern of market influences

While Natural Language Processing has been investigated for seventy years only for linguistic or human interactions purposes, it will be used more and more in finance. Indeed, machine learning and market finance perfectly match since they both inspect big noisy data to find patterns that can be exploited to predict stock market evolution. We know that traders and quantitative analysts build models from numerical financial data such as the moving average of assets, trade volume, and markets volatility, but they also need to understand signals from the real world because the stock market value is not only determined by past financial data (Figure 2). Therefore, unstructured data, such as text, will play a key role in predicting the stock market. In fact, online news media are good sources of real-world information that cannot be translated into financial indicators: simple rumors, latent fears or panic, mergers and acquisitions announcements, recommendations from brokers and ratings services are sources containing information capturing financial opinions.

This is why traders and quantitative analysts will benefit from the automatization of summarization and sentiment analysis of these trends. Sentiment analysis consists in assessing as positive or negative the shared sentiment of a specific stock among investors and analysts. According to the efficient-market hypothesis (Fama, "Efficient Capital Markets", 1970), this information is likely to be quickly integrated into the stock exchange price.

We are convinced of the strong value of automating sentiment analysis in financial headlines since Murray Frank and Werner Antweiler (Antweiler, "Is All that" 2004) studied speeches from organizations' boards of directors and found predictability between message posting and market trading volume. Moreover, Paul Tetlock (Tetlock, "Giving Content to", 2007) confirmed that the optimism or pessimism in the *Wall Street Journal* articles could predict market price volatility.

Analyzing the statements from the Federal Open Market Committee[2], David Lucca and Francesco Trebbi (Lucca, "Measuring central bank", 2009) showed that long-dated Treasury yields have been impacted by central bank communication. Therefore, our Capstone Project explored analyzing sentiments through financial headlines in order to predict the evolution of the stock market. Lastly, we need to assess the political and ethical stakes of opinion mining.

Figure 3: Facebook, Inc. (FB) NasdaqGS - NasdaqGS Real Time Price. Currency in USD from Yahoo! Finance. This plot shows the $25B loss in the Facebook stock price after the Guardian and the New York Times released articles about the Facebook–Cambridge Analytica data scandal.

[2] The Federal Open Market Committee (FOMC) is a Federal Reserve System body which monitor the US open market operations.

Sentiment analysis raises privacy concerns as opinions can be easily quantified

Sentiment analysis allows for a quick way to find people's attitude toward a topic. Combined with the vast amount of data on social media platforms, it can be useful for companies in a number of ways. However, as with any tool, there can be questionably ethical ways to use it. A country could use it to help them censor opinions on certain topics that differ from what they want their population to think. More specifically, it could help them remove posts, or even go after people, that do not show its government in a positive light.

Using Natural Language Processing is questionably ethical when Admiral Insurance, one of the biggest insurance companies in Britain, tried to use NLP on car owner's social media to "offer discounts" on insurance (Ruddick, "Admiral to price," 2016). Nevertheless, this was determined to violate Facebook's privacy policy (Ruddick, "Facebook forces," 2016), specifically section 3.15: "Don't use data obtained from Facebook to make decisions about eligibility, including whether to approve or reject an application…" ("Facebook Platform Policy", 2018). Although this case is more general than just sentiment analysis, the ethical dilemma is the same. At what point is using public information found on social media platforms a violation of privacy? How to avoid another Facebook and Cambridge Analytica data breach?

As proposed by the Fields Medal winner Cédric Villani (Villani, "Donner un sens", 2018), we recommend to organize the equivalent of the Intergovernmental Panel on Climate Change (IPCC) for Artificial Intelligence and opinion mining that is to say an international expertise that can assess and lead the collective and democratic debate in a totally independent, autonomous and transparent way to feed it. The aim is to avoid an Orwellian syndrome where

Natural Language Processing would become a control instance rather than an instrument for liberty and economic development. How? The key is to make the algorithms public, track their biases, not to leave them with the monopoly of the decision, and to reserve the possibility to enrich them by human decision. In the following sections, we will describe the technical details of the individual components of our pipeline that embeds words in a vector space, performs sentiment classification, predicts the stock trends and extracts the keywords from the headlines and the documents (see figure 4 for an overview of this pipeline).

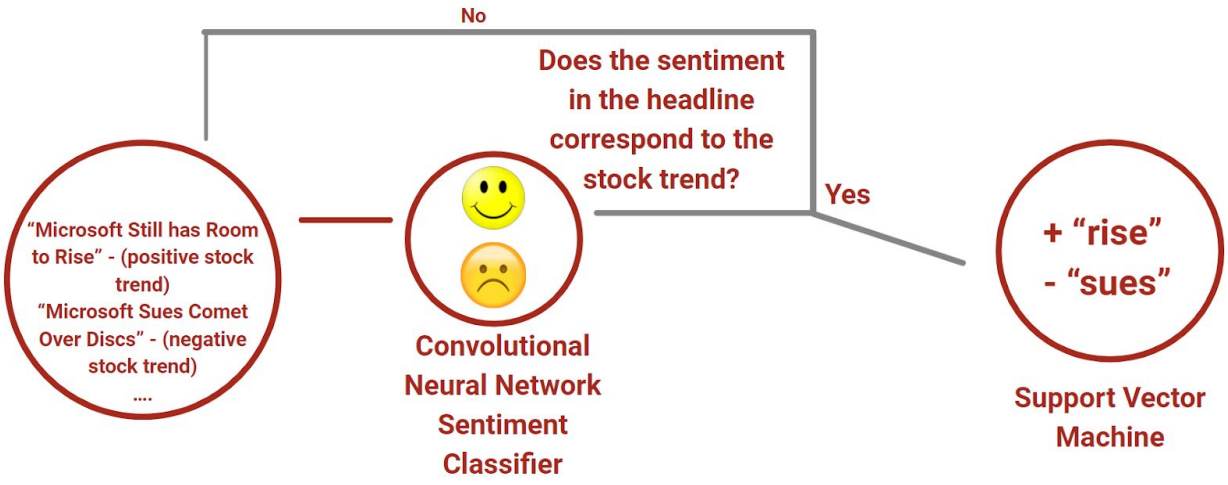

Figure 4: Overview of our method for extracting positive and negative keywords in predicting stock trends from article text

Technical Contributions

Word2vec requires less space than regular one-hot embedding and encodes the semantics of words

Before training any machine learning models we need to prepare the input data, both for training and testing. Natural language words are not in the format that can be directly processed by computers. Therefore, we generate word embeddings, which are essentially dense vector representations of words, to turn natural language words into vectors so that it is readable by computers (Mikolov, Chen, Corrado, & Dean, 2013). We use word2vec to embed the words because it requires less storage space (fewer dimensions) than traditional one-hot vector representation, whose size is equal to the length of the dictionary. The embeddings are also meaningful in the sense that one can compute the similarity between two words by calculating the dot product between the two word vectors corresponding to the words. Furthermore, word2vec also allow us to have word analogies. For instance, the result of the vector for “king” minus the vector for “man” plus the vector for “woman” will result in a vector that is very similar to the vector for “queen”.

The word2vec model is essentially a three-layer neural network consisting of an input layer, an embedding layer, and an output layer. The input and output are both one-hot vectors representing words. This algorithm has two variants. First is the Skip-Gram model. This model’s input is a one-hot vector of the target word and its output will be the one-hot vectors of the context words. The second is continuous bag of words (CBOW) model that exchange the input and output, so it aims to predict the target words using the context words.

For the specific variant of word2vec we are using, namely the Skip-Gram model, the input vector represents a target word, and the output vector represents one of the context words that are in the vicinity of the input word. When the neural network finishes training, all the contextual relationships are captured by the embedding layer, and the weights between the input layer and the embedding layer are the word embedding we seek.

In this paragraph, we will give an example of how to generate the training pairs for word2vec. Let's say we have the sentence "I went to school yesterday" and the window size is 1, which means we consider 1 word before and after the center word to be the context of that word. Let's say the current center words is "school" then we can use the pair ([0 0 0 1 0], [0 0 1 0 0]) to represent the pair of words "school" and "to", and use the pair ([0 0 0 1 0], [0 0 0 0 1]) to represent the word pair "school" and "yesterday". Then we will have two data instances. The first one has input [0 0 0 1 0] and output [0 0 1 0 0], and the second one has input [0 0 0 1 0] and output [0 0 0 0 1]. These two pairs are fed into the neural network for training to understand the semantic of the word "school" in this example. We will talk more about the how we use the data in our capstone in next section.

Amazon Book Reviews are converted to 1-hot vectors pairs to train the neural network

The training data we are using is the Amazon book review data (McAuley, 2016). The data has 8,898,041 reviews from Amazon book purchase record. The size of it is about 8.8 GB. We used these data to train our model. In order to ensure the quality of our word embedding, we used all 8.8 GB of review data in our training. We preprocessed the data by getting rid of all the punctuation marks and special symbols, converting all the words to lowercase and tokenize all

the English words. Finally, we prepare the training pairs by creating a set of target words, each with a corresponding set of context words that surround them, and we will represent them in the form of one-hot vectors. These training pairs are fed into the aforementioned neural network.

Convolutional Neural Net Enables Fast Training and A Holistic View of the Input Sequence

The LSTM (long short-term memory) model is a widely used model in NLP. It is a special type of recurrent neural network model that encodes and "memorizes" important features from the original text. However, we decided to use a convolutional neural network (CNN) in our capstone project at this phase, due to two reasons: faster training time and holistic view of the input.

Conventionally, CNN serves as the backbone for computer vision due to its ability to detect small local patterns to help the computer later correctly identify the object based on the detected patterns. It has applications in NLP, though. In 2014, Yoon Kim from New York University proposed the method of using CNN in sentence classification in his paper "Convolutional Neural Networks for Sentence Classification", and his result showed that CNN could achieve a close to the state-of-the-art accuracy for classifying sentences (Kim, 2014). The first advantage of CNN is that it significantly speeds up the training process using parallelization. LSTM performs training strictly sequentially, one word has to finish training the model before the second word can come in. By contrast, CNN model can be trained on all the words in a given input concurrently because the filters in the convolutional layer operate independently of each other (Perry, 2017). In a recent Facebook research in machine translation,

their CNN model achieved a 9X speed-up compared to RNN models while obtaining state-of-the-art results (Gehring, Auli, Grangier et al, 2017).

The second advantage of CNN in sentiment classification is that it examines the entire input sequence. The CNN first finds local patterns, and later aggregates the small location-invariant features to discover larger structures of the input data. Therefore, CNN has a rather holistic view of the input. On the contrary, LSTM predicts the sentiment using the first n words, and the earlier words get down-weighted later on in the training process. This structure gives more importance to the words in the end, and this unequal weighting is problematic can introduce bias. One remedy for this problem is to have two LSTM's, one working from left to right, and the other one working from right to left. However, this fix doubles the already very long training time of LSTM (Perry, 2017).

CNN Architecture: Detecting Local Patterns and Weighted Voting for Sentiments

After we accurately embedded words in a vector space, we are ready to use them to help predict sentiment. As mentioned previously, we decided to use the convolutional neural network, which is organized in layers. It is a sequential model where the input data goes through each of the layers before finally makes a weighted voting at the very end. The input of our CNN will be in the form of a 2-dimensional array (matrices). The rows of the matrix correspond to the individual words in a particular document. Each row vector has 300 elements, which corresponds to the length of word embedding we generated previously for each word. The neural network architecture we used is shown in figure 5. The input layer is fed 80,000 Amazon book reviews, half positive (1 or 2 stars) and half negative (4 or 5 stars). The words from these reviews are then

converted to vectors in the Embedding layer. The Embedding layer is initialized using word embeddings generated previously. The dropout layer randomly makes some values zero, which helps combat overfitting. Then, these word vectors are sent through three different convolutional layers (Conv1D), with filter sizes of 3, 4, and 5. The convolutional layer contains parameters consisting of a set of filters that "scan" through the layer and detect the presence of notable features. The feature maps resulting from the dot product between the filters and the previous layer are stored in the convolutional layer. One simple example of a feature map in our case is a vector that captures the meaning of a short phrase that contains multiple words. After the convolutional layer, the max-pooling layer down-samples the output of the previous layer by keeping the maxes of non-overlapping regions. For example, pass the vector [2, 5, 9, -1, 7] into a max-pooling layer with pool size 2 will result [5, 9] as the output.

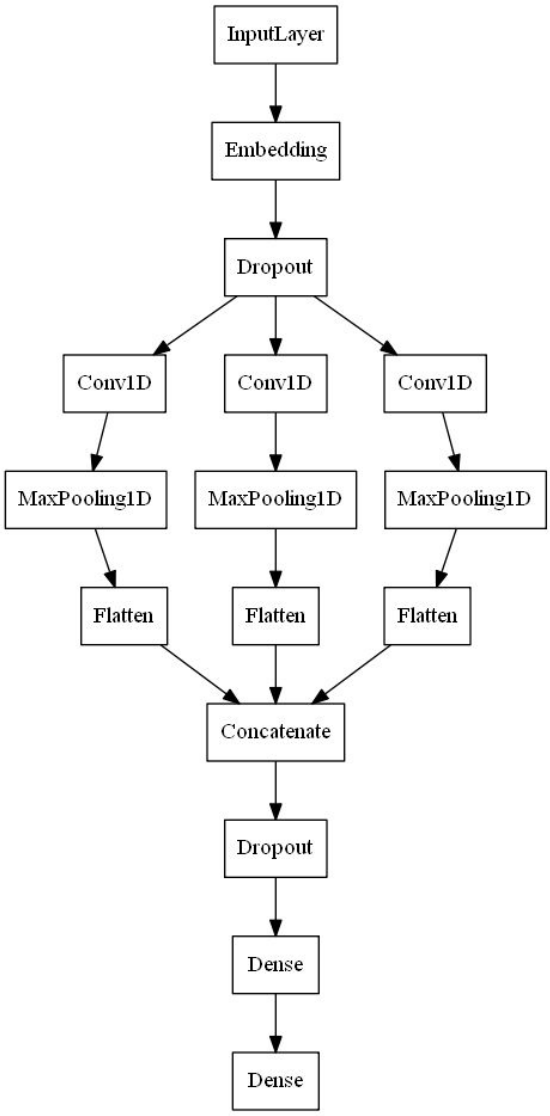

Figure 5: Convolutional Neural Network Architecture

Each of these convolutional layers output is sent to a max pooling layer of size 2 (MaxPooling1D), which outputs the two largest values from the input vector. For example, if the vector [2, 5, 8, -1, 7] were passed through a max-pooling layer with pool size 2, the output

would be [5, 8]. After max pooling, the output is sent through a Flatten layer, which takes a set of vectors and "flattens" them into a single 1-D vector. Then, the three Flatten layers are sent to the Concatenate layer, which combines them into one long vector. Another Dropout layer follows to help against overfitting. Two Dense layers ensue. The dense layer (AKA fully connected layer) conducts a weighted voting, where each element of the feature vector "votes" for the final classification. Finally, the final classification result, which is either 0 (negative sentiment) or 1 (positive sentiment), is contained in the output layer.

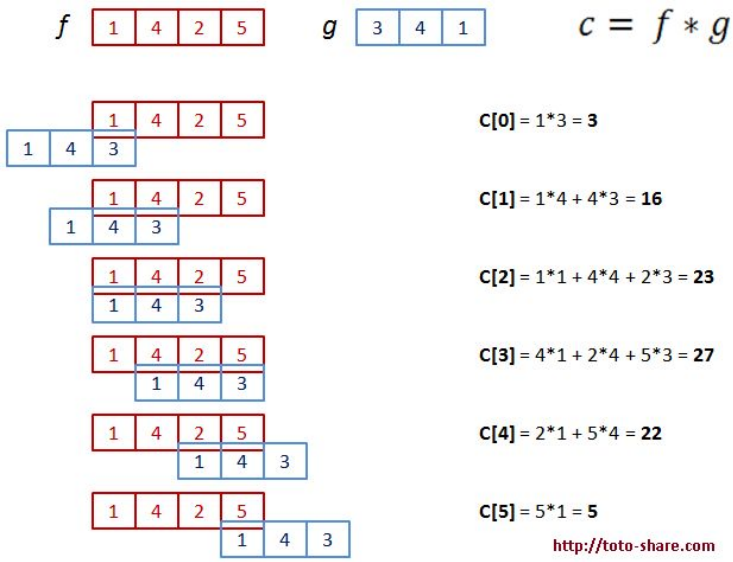

Figure 6: Convolution in 1D [1]

(http://www.stokastik.in/understanding-convolution-for-deep-learning/, 2016)

Figure 7: Max-pooling in 1D

[2](https://software.intel.com/en-us/daal-programming-guide-1d-max-pooling-forward-layer , 2017)

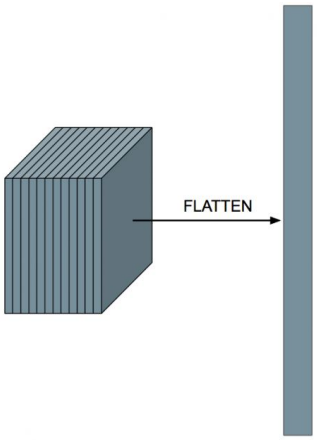

Figure 8: Flattening Layer

(https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/, 2017)

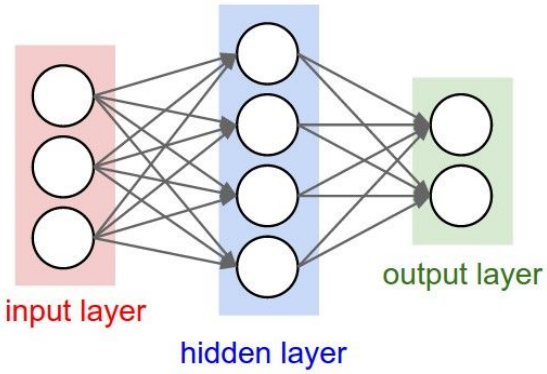

Figure 9: Dense layer (http://cs231n.github.io/neural-networks-1/)

CNN Training Data: Amazon Book Reviews with Sentiment Annotations

Since training with CNN requires considerable computing power without GPU acceleration, we decided to only perform the training using a small subset of the 8.3GB Amazon

book reviews dataset. We selected 100,000 book reviews from our original dataset, with 50,000 of them having 4 or 5 stars, and the other half having 1 or 2 stars. The former represents the reviews with positive sentiments, and the latter represents the reviews with negative sentiments. Reviews with 3 stars are deliberately left out in the selection process because they tend to have neutral sentiments and we are only interested in the polarities of the sentiments. For each instance of the data selected, we then relabeled it using our binary system, ie, 4 or 5 stars reviews are labeled as "pos" and 1 or 2 stars reviews are labeled as "neg". The original Amazon reviews can have lengths up to 5000 words, and the long lengths of some reviews slow downs the training process. Therefore, we have decided to limit the maximum number of words in a given data instance to 400 words. When a review exceeds 400 words, we simply truncate the original review and only keep the first 400 words. We further randomly split the data into 80% training data and 20% validation data. We will train on the training set and use the validation accuracy resulted from testing on the validation set as a guide to help us tweak our parameters.

Our word2vec embedding achieves near the state-of-the-art quality

The outcome of the word2vec embedding training is that we can make use of the embedded result that reveals the similarity between words. The Spearman correlation coefficient result from applying word2vec on Amazon data is currently 0.422. The-state-of-art is 0.46 (Sun, Y., Rao, N., & Ding, W., 2017). The Spearman correlation is a measurement of the distance between our model's similarity score compared to a human annotated similarity score, using the same list of word pairs. We used SimLex-999, which has 999 pairs of words, along with the

annotated similarity. It is considered a gold standard for word embedding quality evaluation (“SimLex-999”, n.d.).

The model, on its own, allows us to perform three different tasks. First, we are able to find the words that are similar in meaning to an input word of choice, through a method called most_similar( ). For example, when we input “friend” as an argument, the resulting output words are “bff”, “friends”, “roommate”, “co-worker”, “bestfriend”, “bestie”, “freind (sic.)”, “cousin”, “coworker” and “sister”. Second, we can perform the word analogy inference by providing three arguments like “man”, “king”, “woman”, and the output will be “queen”. Third, we can visualize the semantic distance of different words by plotting the word vectors in a lower dimension using PCA. One plot we generated is presented below in Figure 10. The figure shows that similar words are close to each other. For example, the words “long”, “longer” and “longest” have similar positions in the graph. In addition, we can also find an example of word analogy in the graph. If we calculate the subtract “cats” and “cats”, we will obtain a vector similar to the resulting vector by subtracting “dog” from “dogs”.

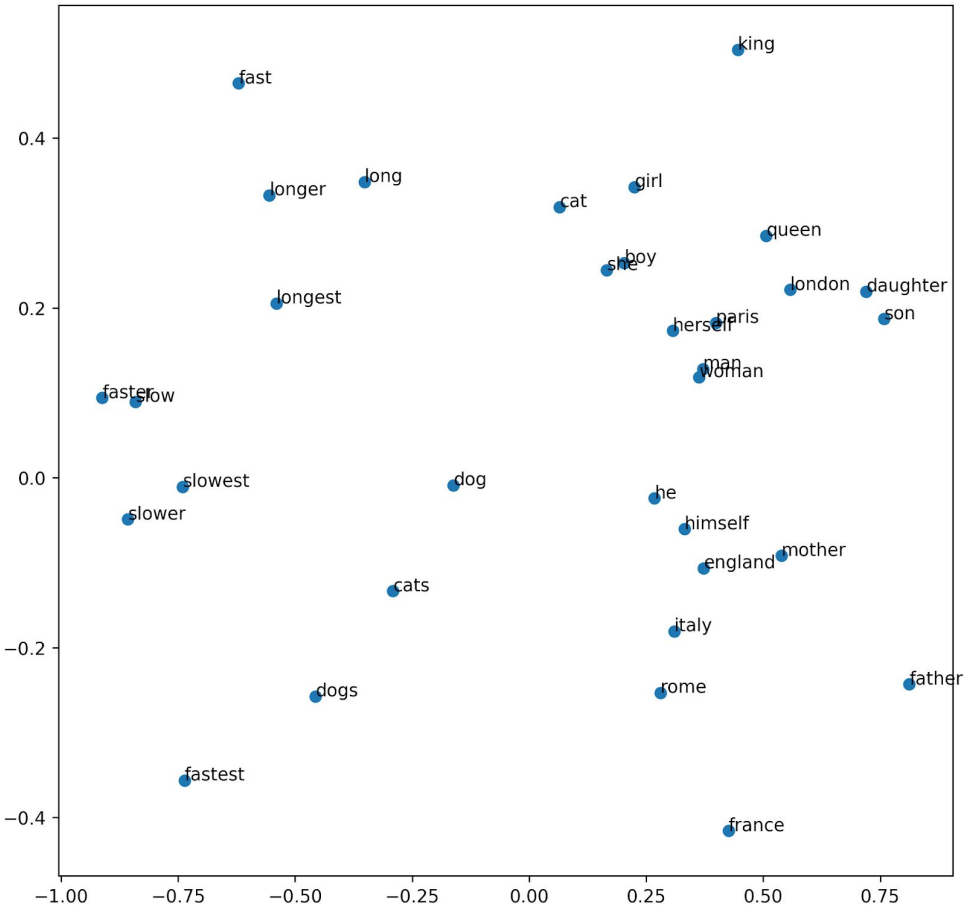

Figure 10: Example of similar words that are close when projected in a 2D space

We will talk more about details for the machine learning algorithms we used in the following sections.

Our CNN accurately detected sentiment in Amazon Reviews but does significantly worse predicting the sentiment of financial headlines

After training, we predicted the sentiment of 20,000 Amazon reviews not included in our training set with 93% accuracy. We also used our neural network to predict a handful of IMDb reviews, where it performed remarkably well. For instance, when the “most helpful” review for a notoriously bad movie was input, the output was 7.48e-08, an extremely negative sentiment.

Unfortunately, the accuracy of our model is only so high when the input text comes from reviews.

Ideally, a sentiment analysis model should be able to detect sentiment across different domains. To test our model, we manually labeled 74 financial headlines as positive or negative. A positive headline is something like “Economic forecast strong for Europe”, whereas a negative one is “Why are global stock markets falling?” We then predicted the sentiment using our CNN. Only 44 of the 74 headlines were correctly predicted (59.4%). For example, our CNN incorrectly predicted “Bitcoin remains in freefall, dives below $7000” as being positive, but it correctly said “Why you shouldn't panic about the market meltdown (yet)” was negative. One possible contribution to the poor performance may be our word embeddings. The fact that the word embeddings are learned from reviews on books means they may not be well suited for giving meaning to the quite different domain of financial headlines. For example, “crash” would likely refer to a reduction in the value of assets, whereas in book reviews it is more likely to mean something physically crashing.

Instead of looking at the sentiment of headlines, a more applicable output from them is stock trends. We collected new financial data set used for stock prediction: we paired financial news headlines with the change of stock value for any specific company. For instance, on February 5th, a news titled “Microsoft released its next generation operation system” and the next day its stock value increased. We could say that the words in the sentences tend to be positive. We used data like the example to train our new model and use our improved application to predict changes in public companies’ stock price based on news headlines. We found a comparison of various algorithms for doing this and found neural networks did not perform well,

which matches our result. The best performing algorithm used SVM combined with tf-idf (Lseiyjg, 2017). Below is a brief technical overview of the new method we are going to use to extract the keywords in the headlines and the documents that strongly correlate with stock trends.

The tf-idf representation extracts the features for training from each headline

"Tf and idf stand for term frequency and inverse document frequency respectively" (Ravindran, K., & Iannelli, M. , 2017). The "term frequency is the number of times a given word appears in the document divided by the total number of words in that document" (Goldstein, I., Arzumtsyan, A., & Uzuner, Ö. 2007). Thus, a word is weighted more heavily if it occurs many times in a document. The "idf" of a word, by definition, is taking "the log of the ratio of the number of all the documents to the number of documents that contain that word" (Ravindran, K., & Iannelli, M. , 2017). The log here is used here to dampen the strength of down-weighting. Therefore, the more frequently a word appears in a collection of documents (AKA the corpus), the more down-weighted that word is. idf serves as an offset that helps to mitigate the effect of common words like the, is, etc on the tf score. The tf-idf statistic is simply the product of tf and idf. Consequently, the higher the tf-idf score, the more important a word is to a document.
We use the tf-idf representation to extract the features of the headlines, so the features can be used in the classification algorithm introduced in the next section. Before transforming the original headlines into the tf-idf representation, we first get rid of all the punctuation and stop words. Then we convert the set of all input headlines into a matrix form. The rows of the matrix correspond to each headline, and the columns of the matrix correspond to each unique word that appears in our corpus. Each entry of the matrix is then the tf-idf score of a particular word in a

particular document. Now we are ready to input the tf-idf matrix as features into our classification algorithm.

Support vector machine classifies the headlines and encodes relative importance of keywords in its weights

Support vector machine, or SVM, is a machine learning algorithm used for classifications. In our case, we want to classify the headlines to determine whether they positively or negatively affect the stock price. For each headline, we associate it with the sign of the difference between the stock's closing price and the opening price on the date the news article was released. During training, the support vector machine uses stochastic gradient descent to solve the following optimization problem: given the set of data instances, which are points in high dimensional space, find a hyperplane that separates the instances with positive label (stock price increases) from the instances with the negative label (stock price decreases), such that the distance between the closest instances to the hyperplane is maximized. Moreover, in our model, the loss function also contains a term that regularizes the l1 norm of the weights. For a graphical illustration, see Figure 11 below. The squares and circles each represent a class. The green line represents the separating hyperplane.

The normal vector orthogonal to the hyperplane contains the weights of this model. We denote the weight vector by w. In our model, we do not apply any transformation to the input data (we are using the linear kernel), so each element in the weight vector corresponds to each word in the corpus. A large positive weight means the word is strongly associated with the stock increase. A large negative weight means the word is strongly associated with the stock decrease.

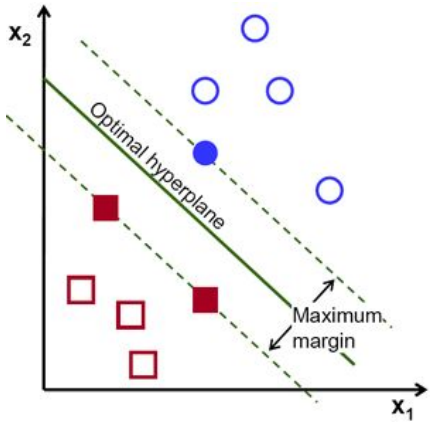

Figure 11: An example of an SVM hyperplane separating two classes

(https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html)

Support vector machine gives good testing results but produces disappointing sets of positive and negative words

We trained the SVM model on news headlines related to the keyword "Bitcoin" and, later, the keyword "Microsoft." These headlines were scraped from LexisNexis' database[3] of articles. We first selected only the headlines with the number of words greater than or equal to 5. This filters out headlines with very little information. Then, we filtered out stop words and numbers. Next, we obtained the corresponding label for each headline, *i.e.* the stock price trend, by comparing the average 2-day closing price and the closing price of that stock on the day when the headline was released. We assigned a "1" to the headline instance when the stock price increased during the period, and a "-1" to the headline instance when the price decreased. We also leveraged the CNN sentiment classifier we trained to help us prefilter the training data for SVM. For each of the headline, we classified its sentiment and compared it to the stock price

[3] https://www.lexisnexis.fr/

trend; instances whose trend and sentiment do not match were discarded. This filtering step allows us to ensure the headline's expectation for the stock price matches with the actual price change, so we did not introduce noisy training examples with headline's sentiment contradicting the actual market trend. Finally, we constructed the input to the SVM by constructing a tf-idf matrix for the headlines, as well as the full articles.

We tuned the following parameters and found the best values for them using grid search: {'alpha' (constant that multiplies the regularization term): 0.0006309, 'max_df' (max document frequency): 0.1650, 'max_iter' (max number of iterations): 5, 'min_df' (min document frequency): 0.01, 'ngram_range' (we use unigrams and bigrams): (1, 2), 'random_state': 43, 'sgd_loss': 'hinge', 'sgd_penalty': 'l1', 'tfidf_norm': 'l1', 'use_idf': True}.

Results

In Table 1 the "Headline" row means that we only used headlines from news media to train our model. In contrast, the "Full Article" entry means we used the entire news document to train our model. The "Filtered" rows mean that we filtered out the training instances that have contradicting sentiment and stock trend. The "Unfiltered" rows mean that we did not filter out any training example. Our models' accuracies range between 59% - 66%. After examining the accuracies, we cannot pinpoint a universal pattern to judge the best model to use. Indeed, for instance, it seems that, concerning the Microsoft market, the "Headline"/"Filtered" model had the best performance, whereas for the Bitcoin market it is the "Full Article"/"Filtered" model that is the best one. However, for full articles, we achieved higher accuracy if the text is not filtered than filtered. While for "Headline" only, we achieved higher accuracy when we filter

than when we do not. Thus, we would recommend testing different models and keep the best model even though the performance may not be extrapolatable for predicting future stock trends. The average accuracy for “Full Article” is better than that for “Headline,” and the average accuracy for “Filtered” is better than that for “Unfiltered.” However, these differences are not statistically significant. Please refer to the Appendix A for more examples of extracted keywords.

			Accuracy	Precision	Recall
Microsoft Market	**Full Article**	**Filtered**	0.6044	0.5902	0.9890
		Unfiltered	0.6308	0.6450	0.7680
	Headline	**Filtered**	**0.6480**	0.6478	0.8602
		Unfiltered	0.5903	0.5992	0.8379
Bitcoin Market	**Full Article**	**Filtered**	0.6600	0.6716	0.7895
		Unfiltered	**0.6650**	0.6563	0.7850
	Headline	**Filtered**	0.6400	0.6164	0.8491
		Unfiltered	0.6350	0.5854	0.9505

Table 1: Accuracy, Precision and Recall scores for the Microsoft and Bitcoin markets

Table 2 presents the best result we obtained for keyword extraction. They are for the Bitcoin market and trained under the ”Full Article”/“Filtered” scenario. We determined that the bold phrases were well-classified: for instance, the word “alert” matches human intuition on the impact of this word in a news article on the Bitcoin market.

Extracted keywords - Bitcoin Market - Full articles, filtered			
Positive Words	Weights	Negative Words	Weights
equity	3.26	**south korean**	-4.75
portfolio	3.05	**alert**	-4.42
fintech	2.94	**committee**	-3.29
launch futures	2.85	sums	-3.23
young	2.71	**people bank**	-3.02
russian	2.60	**decline**	-2.78
exchange traded	2.54	premium	-2.76

Table 2: Example of keyword extraction results for the Bitcoin market, filtered, full articles

Discussion

As shown by the testing results, our SVM model appears to be promising even though it did not consistently extract meaningful positive and negative keywords for predicting trends in the stock market or the Bitcoin exchange market. At the same time, our testing accuracy still has some potentials for improvement. After discussing with our advisor, we have identified several limitations of the current model.

First, we encountered the problem of "garbage in, garbage out". After browsing through the headlines we scraped from LexisNexis, we noticed headlines have varying levels of relevance to the changes in stock prices. A few of them have no immediate relevance to the company we searched for. Incorporating these headlines into our training data only introduces

noises. Furthermore, our model lacks the ability to discern the quality of the source of the headline. Certain sources are more reliable than others, and preferably, they should have been given a higher weight.

Second, the headline can convey conflicting opinions with regard to the same stock on a given day. Stock transactions occur when there are two parties with the direct opposite views on the value of the same stock. Similarly, there are lots of headlines that express differing views on the same stock, despite the fact that the stock price can only eventually go in one direction. Labeling two data instances with opposite views (because they were published on the same day) with the same label (direction of change in stock price) can severely hinder the prediction accuracy of our model.

Third, the current method we use to match the headlines with the changes in the stock price trends is problematic. To prepare the training dataset, we match each headline with the sign of the difference of the two-day average closing price and the closing price on the day the news article is released. This practice ignores the possibility that the event that contributes to the changes in the stock price trend may take effect slowly over a period of weeks or even months. Our current implementation assumes the effect is immediately visible, which might not be the case in some scenarios.

To address some of the issues discussed above, we have prescribed the following modifications/improvements to our models, which hopefully can inspire future capstone teams when they design their own models:

1) Incorporate the sentiments of the texts (from the CNN classifier) as a separate feature in the SVM model, or alternatively, add word frequency features, *e.g.* tf-idf, bag-of-words in the CNN classifier.
2) Introduce another classifier that determines the relevance/quality of the headlines, and discard the noisy training instances before training the stock price trend classifier.
3) Train a general-purpose classifier that can predict the stock price trend for any company, rather than one specific company or sector.
4) Use more sophisticated models like bidirectional LSTM (long short-term memory)/GRU (gated recurrent unit) Recurrent Neural Network (RNN) to predict the stock market trend.

Conclusion

In this project, we introduced how text mining and NLP could change the way companies collect and analyze consumers' feedbacks. We also described how social networks and digital journalism can influence markets and why we see it as a critical field for financial analysts. We underlined the risks raised by opinion mining and proposed a way to mitigate it. After conducting a leadership reflection on practical applications of opinion mining, we discussed the technical aspects; there are three main modules in our pipeline. Firstly, we embedded words into the vector space by using Word2vec and obtained a word embedding whose quality is close to the state-of-the-art. Secondly, we introduced a sentiment classifier that accurately predicts the sentiment of Amazon reviews, but does not predict the stock trend as well. Lastly, we trained an SVM classifier that achieved up to 66% accuracy predicting the market price trend for Bitcoin, and we extracted positive and negative keywords that influence the stock price. For future work,

exploring conflict attitudes between news headlines on the same day is important. Furthermore, how to assign weights to different data sources is still unclear, and an advanced study of the difference between headlines' long term and short term effects is necessary.

Acknowledgment

This work would not have been possible without support from the Electrical Engineering and Computer Science Department of the University of California, Berkeley, the Fung Institute for Engineering Leadership and all the people who contributed to the success of this capstone project. Especially, we would like to express our gratitude to Professor Laurent El Ghaoui for his valuable guidance and help during the past year. We would also like to thank the teaching staff of E295, particularly our GSI, Jonathan McKinley, for his insightful feedback on our previous revisions of this report.

References

[1] Anonymous (2016, September 16). Convolution in 1D, from http://www.stokastik.in/understanding-convolution-for-deep-learning/

[2] Anonymous. (2017). Max-pooling in 1D, from https://software.intel.com/en-us/daal-programming-guide-1d-max-pooling-forward-layer

[3] Anonymous. (n.d.). Support Vector Machine, from

https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[4] Bengio, Y., Courville, A. & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence, 35*, 1798-1828.

[5] Deshpande, M. (2017, September 24). Flattening Layer, from https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks

[6] Dense Layer, from http://cs231n.github.io/neural-networks-1/

[7] *EY*. Becoming an analytics–driven organisation to create value. (2015). Retrieved from http://www.ey.com

[8] Facebook Platform Policy. (n.d.). Retrieved February 10, 2018, from https://developers.facebook.com/policy/

[9] Fama, E. "Efficient Capital Markets: A Review of Theory and Empirical Work," The Journal of Finance, May 1970, pages 383-417

[10] Goldstein, I., Arzumtsyan, A., & Uzuner, Ö. (2007). Three approaches to automatic assignment

of ICD-9-CM codes to radiology reports. In *AMIA Annual Symposium Proceedings*(Vol. 2007, p. 279). American Medical Informatics Association.

[11] Gehring , J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, May 9). A novel approach to neural machine translation. Retrieved February 11, 2018, from https://code.facebook.com/posts/1978007565818999/a-novel-approach-to-neural-machine-translation/

[12] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1181

[13] Lseiyjg (2017). Use News to Predict Stock Markets. Retrieved March 05, 2018, from https://www.kaggle.com/lseiyjg/use-news-to-predict-stock-markets

[14] LeCun, Y., Bengio, Y. & Hinton, G. E. (2015). Deep learning. *Nature, 521*, 436-444.

[15] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*. arXiv:1301.3781v3

[16] David O. Lucca, Francesco Trebbi. (2009) Measuring central bank communication: an automated approach with application to FOMC statements. National Bureau of Economic Research.

[17] McAuley, J. (2016). Amazon product data. Retrieved February 11, 2018, from http://jmcauley.ucsd.edu/data/amazon/

[18] Perry, T. (2017, May 22). Convolutional Methods for Text. Retrieved February 11, 2018, from https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f

[19] Pressing problems, solved. (n.d.). *Gallup.* Retrieved from http://www.gallup.com

[20] Ruddick, G. (2016, November 01). Admiral to price car insurance based on Facebook posts. Retrieved February 10, 2018, from https://www.theguardian.com/technology/2016/nov/02/admiral-to-price-car-insurance-based-on-facebook-posts

[21] Ruddick, G. (2016, November 02). Facebook forces Admiral to pull plan to price car insurance based on posts. Retrieved February 10, 2018, from https://www.theguardian.com/money/2016/nov/02/facebook-admiral-car-insurance-privacy-data

[22] Ravindran, K., & Iannelli, M. (2017, July). Replicated Sensing in Wireless-Networked Smart Systems with Cloud-assisted Support. In *Proceedings of the ACM Workshop on Distributed Information Processing in Wireless Networks* (p. 3). ACM.

[23] SimLex-999. (n.d.). Retrieved March 05, 2018, from https://www.cl.cam.ac.uk/~fh295/simlex.html

[24] Singh, R. (2015). Text Analytics Market by Deployment Model (On-premise model and Cloud based model) and Application (Competitive Intelligence, Customer Relationship management, Predictive Analytics, Fraud detection and Brand Reputation) - Global Opportunity Analysis and Industry Forecast, 2013 - 2020. *Allied Market Research.* Retrieved November 2, 2017, from https://www.alliedmarketresearch.com.

[25] Sun, Y., Rao, N., & Ding, W. (2017). A simple approach to learn polysemous word embeddings. arXiv preprint arXiv:1707.01793.

[26] Tetlock, Paul C., Giving Content to Investor Sentiment: The Role of Media in the Stock Market. Journal of Finance, 2007

[27] Tractica, (2017, August 21), Natural Language Processing Market to Reach $22.3 Billion by 2025, from https://www.tractica.com/newsroom/press-releases/natural-language-processing-market-to-reach-22-3-billion-by-2025/

[28] Villani, C., Donner un sens à l'intelligence artificielle : pour une stratégie nationale et européenne, 2018.

[29] Antweiler, W. and Frank, M.Z. 2004. Is All that Talk just Noise? The Information Content of Internet Stock Message Boards. Journal of Finance, pages 1259– 1294.

Appendix A: Full Results

		Filtered				Not filtered			
Microsoft	Full text	**Accuracy**	0.6044	**Naïve model accuracy**	0.5670	**Accuracy**	0.6308	**Naïve model accuracy**	0.5639
		Precision	0.5902	**Recall**	0.9890	**Precision**	0.6450	**Recall**	0.7680
		Negative keywords	**Weights**	**Positive keywords**	**Weights**	**Negative keywords**	**Weights**	**Positive keywords**	**Weights**
		patents	-10.6	azure	4.86	economy	-3.50	brief	3.99
		shareholder	-9.70	registered	4.75	accelerate	-3.37	leadership	3.90
		sinofsky	-8.86	photos	3.99	creative	-3.29	facing	3.26
		finnish	-8.06	unveils	3.98	photo document	-3.25	windows software	2.81
		linkedin	-8.06	tie	3.58	powerful	-3.19	tie	2.78
		common	-6.26	expanding	3.07	ebay	-3.14	battery	2.74
		handset	-5.64	sign	3.05	departure	-3.08	enforcement	2.72
	Headlines	**Accuracy**	0.6480	**Naïve model accuracy**	0.5794	**Accuracy**	0.5903	**Naïve model accuracy**	0.5670
		Precision	0.6478	**Recall**	0.8602	**Precision**	0.5992	**Recall**	0.8379
		Negative keywords	**Weights**	**Positive keywords**	**Weights**	**Negative keywords**	**Weights**	**Positive keywords**	**Weights**
		skype	-5.57	world	2.96	skype	-4.75	bid	1.15
		giant	-5.14	takes	2.62	linkedin	-3.96	make	0.81
		phone	-4.96	make	2.62	boot	-2.96	tablet	0.75
		ballmer	-3.99	help	2.28	mobile	-2.41	brief	0.70
		internet	-3.05	explorer	2.07	giant	-2.34	unveils	0.67
		windows	-2.82	unveils	1.94	billion	-2.18	gets	0.66
		ceo	-2.80	operating	1.43	apple	-2.11	surface	0.64
Bitcoin	Full text	**Accuracy**	0.6600	**Naïve model accuracy**	0.5700	**Accuracy**	0.6650	**Naïve model accuracy**	0.5350
		Precision	0.6716	**Recall**	0.7895	**Precision**	0.6563	**Recall**	0.7850
		Negative keywords	**Weights**	**Positive keywords**	**Weights**	**Negative keywords**	**Weights**	**Positive keywords**	**Weights**
		south korean	-4.75	equity	3.26	team	-4.01	easy	4.59
		alert	-4.42	portfolio	3.05	monthly	-3.65	creating	3.83
		committee	-3.29	fintech	2.94	short term	-3.61	official	3.65
		sums	-3.23	launch futures	2.85	committee	-3.47	faster	3.48
		people bank	-3.02	young	2.71	processes	-3.12	let	2.93
		decline	-2.78	russian	2.60	net	-3.07	bull	2.85
		premium	-2.76	exchange traded	2.54	percent	-2.99	wannacry	2.84
	Headlines	**Accuracy**	0.6400	**Naïve model accuracy**	0.5300	Accuracy	0.6350	**Naïve model accuracy**	0.5050
		Precision	0.6164	**Recall**	0.8491	Precision	0.5854	**Recall**	0.9505
		Negative keywords	**Weights**	**Positive keywords**	**Weights**	**Negative keywords**	**Weights**	**Positive keywords**	**Weights**
		korean	-11.3	tech	10.1	ethereum	-3.91	exchange	0.70
		offerings	-11.1	investment	7.55	fears	-3.77	bubble	0.36
		china	-10.8	business	7.36	korea	-3.58	value	0.23
		south	-9.20	record	7.15	ban	-2.72	currencies	0.23
		latest cryptocurrency	-9.12	street	6.74	latest	-2.37	says	0.15
		launch	-8.21	gold rush	6.63	world	-2.09	money	0.12
		falls	-8.19	rise	6.02	coin	-2.06	investors	0.03