# Machine Learning Engineer Nanodegree

## Capstone Proposal

Michael Boker
May 1st, 2017

### Domain Background

For a large part of the history of the stock market, math played almost no part in the speculations made by investors. Predictions were made on a qualitative basis, affected by the reputations and characteristics of the people at a business, the product or service sold by the company, and the general landscape of the market at the time. Beginning in the 1950's, quantitative analysis began to make its way onto the scene.[1] This involved mathematical models used to find predictive trends in the market, which could be used to assist in making wise investments. As computers became more capable and ubiquitous, quantitative analysis saw more adoption by investors. Eventually, many of the trader middle-men were cut out, and automated trading began gaining popularity. Now, some estimates say that 70 percent or more of trades are made automatically by computers, with no human involvement.[2]

The analysis behind this automatic trading has mostly been comprised by mathematical models based on trends in the stock values. Quantitative analysis is very powerful, but it cannot completely replace the usefulness of a human in stock analysis. A human can pay attention to the things going on in the world, and provide some insight which may not be found in the numbers. Current events around the world can have a major impact on the stock market, and these effects cannot be predicted by stock values alone until the values are already changing, and it is too late.

### Problem Statement

Advancements in deep learning have made it possible to train computers to analyze events in the world, like a human would. Deep neural networks are commonly used for determining the sentiment of a body of text. This would be useful in augmenting the existing quantitative analysis being used today. A model could be trained to learn the correlations between current events and fluctuations in the stock market.

The idea for this project is to use news headlines to predict market behavior. Deep neural networks seem useful for analyzing the news for a day, and predicting the resulting effect on the stock market. Specifically, predictions will be made as to whether the values of stocks would rise or fall in a given day. Ideally, models could be created for predicting the behavior of individual stocks, and predicting the amount of change. However, this project will focus on one of the world’s most popular stock indexes, and will predict a general increase or decrease in value.

### Datasets and Inputs

The Dow Jones Industrial Average (DJIA) is a stock index, which incorporates the stock values of 30 of the largest companies in America.[3] An index is the average of a group of stocks. It reflects the overall performance of that group. The DJIA is commonly used as an indicator of the overall performance of the stock market as a whole. The behavior of individual stocks is less predictable than that of a group of stocks, due to the effects of probability being more reliable over the aggregate a group. The DJIA would be a good source of information for testing the usefulness of deep learning in stock price prediction.

A Kaggle dataset already exists for purposes similar to this project, and it contains data from 1990 days, ranging from the 8th of June, 2008 to the 1st of July, 2016.[4] This dataset pairs a collection of the top 25 news headlines from each day with a label indicating whether the DJIA rose or fell that day. The news headlines are taken from the WorldNews subreddit, and are ranked based on user up-votes. This seems to be a pretty good indicator that these are articles that the general public finds interesting. Therefore, they are articles that are most likely to elicit a response from the public, which could have an effect on the DJIA.

The data is formatted as a CSV file, with 27 columns. The first column is the date. The second column is a binary indicator of the movement of the DJIA for that day - 0 for down, 1 for up or no movement. The remaining 25 columns each contain a headline, starting with the most popular headline at column 3 and listed in descending order of popularity to column 27. The dataset contains 1990 rows, and the data is split fairly evenly between days where the DJIA rose or stayed the same and days where it dropped. Out of the 1990 days represented, the DJIA rose or stayed the same on 1065 days and dropped on the remaining 925.

## Solution Statement

Recurrent Neural Networks (RNN) are a popular deep learning technique for text analysis and can be powerful for sentiment analysis. RNN’s can be used to learn context in a body of text, and not simply learn based on the words present in some text, as would be done with a simple feed-forward neural network and a bag of words. For example, given a headline such as “United States Prepares to Wage War on Obesity,” an RNN would take the entire headline into consideration. It might be able to determine that the phrase “Prepares to Wage War” is not referring to literal warfare. A traditional feed-forward network would see the word “War” and probably determine that this article is about actual warfare, because it does not consider the surrounding context. The long short-term memory (LSTM) model is a type of RNN cell, which is a particularly useful tool for sentiment analysis. It can be used to learn from and predict on entire bodies of text, remembering the context of the entire piece as it goes along. Traditional, "vanilla" RNN’s have limits in the length of context that they can remember, due to problems encountered during back propagation when too much context is retained.[5] Therefore, an RNN with LSTM cells will be used for this project.

## Benchmark Model

For a benchmark model, a cue will be taken from work produced by Bryce Taylor, who attempted to create a useful model similar to what we will be producing for this project. For a given time period, Taylor compared the performance of two algorithms - one which always predicted an increase, and one which always predicted a decrease. He used the better performing of these two for his baseline. This model can never have error above 50%, and in the relatively infant field of using natural language processing to predict stock behavior, this seems like a sufficient benchmark.[6]

## Evaluation Metrics

For evaluating the performance of the baseline and final models, a combination of prediction accuracy and F score will be used. Accuracy will indicate the ratio of correct to incorrect predictions, but can be misleading if the data is skewed. F score seems to be a good metric for ensuring that the final model truly learned from the data, and did not just learn how to cheat from the data. If the values in the training data skew towards one label or the other, and the final model learns to be biased towards that label, it will receive low F scores. F score works by taking into account both the precision and the recall of a model. Precision is the probability that, given the model predicts a label, that label is actually correct. Recall is the probability that, given a label is correct, the model predicts that label.

The benchmark model will have a reasonably high accuracy - at least 50%. However, this will come at the cost of F score, as this model blindly chooses the same label for every point. Therefore, the F score for one of the labels will always be 0, and the f-score can never get as high as .5. A successful final model will surpass both the accuracy and f-score of the baseline model.

## Project Design

For preprocessing the data for this project, some modifications will be made to the headline strings. For each day in the dataset, each of the 25 news headlines will be concatenated into one string, separated by a delimiting value, “[NEW HEADLINE],” which is used to indicate the start of a new headline. Punctuation will be replaced with tokens representing the respective symbols, for example "[COMMA]" will replace occurences of "," in the text. Alternatively, punctuation could be removed, since we are not performing text generation. However, the punctuation could have an effect on the meaning of some headlines, and may provide valuable information, at a relatively low cost in terms of feature space. Finally, stop words will be removed from the data. Stop words include common words such as "the" and "is" which do not provide much meaning to the text. Similar to punctuation, these words would be left in if we were training a model to generate text. However, for training a classifier, they do not serve a purpose.

Once preprocessing is complete, the data will be split into 90% training and 10% testing. The training set will then be divided into 90% training and 10% validation. Then, a bag of words will be built from the training set, pairing each word with the number of occurrences of that word in the training set. Next, words which occur fewer than 5 times in the training set will be removed from the bag. This threshold of 5 occurrences has been found to effectively reduce noise caused by infrequent words, while not removing so much of the vocabulary as to hamper training.[7]

Each of the words in the training vocabulary, as well as tokens for punctuation and the start of new headlines, will now be assigned an integer index value, which is what will be used to represent the respective token to our model.

The word indexes will be used to create vectors which represent the sequences of words. These vectors will contain the indexes of the words, in the sequences in which they appear in the headline strings, and will each be left padded with zeros to a uniform size. For example, with a vocabulary of ["she", "knows", "that", "is", "strong"], the sentence, "She knows that she is strong," and a uniform size of 10, the encoding would be [0, 0, 0, 0, 1, 2, 3, 1, 4, 5]. The uniform size should be large enough to fit the contents of most strings, but not unnecessarily large, because this would hamper performance of the model. Upon quick observation, headlines appear to be have a range of about 5 to 45 words, and lengths look to be normally distributed through this range. With 25 headlines for each day, it seems reasonable to make the uniform vector length 650 words. The reasoning for this is that 25 headlines of an average of 25 words equals 625 words, plus 25 words for the "new headline" delimiters. Removal of infrequent words and stop words will likely be offset by inclusion of punctuation tokens.

These vectors will be passed through an embedding layer, to reduce the dimensionality of the input. This layer will reduce the dimensionality of the vocabulary to 125 dimensions. Research has shown that there are often only marginal gains in performance for embedding outputs greater than 50 dimensions. There are almost no improvements seen above 200 dimensions.[8] The midpoint of this range seems like a good place to start.

The output of the embedding layer will be input into one or two (sequential) layers of LSTM cells. The LSTM layers will most likely consist of 256 or 512 units each. The output of the LSTM cells will be fed into one or two (again, sequential) fully connected layers, of 128 or 256 units each. The output of these layers is finally fed into a single output cell with a sigmoid activation. The decisions between one or two LSTM layers and one or two fully connected layers, as well as the number of cells in each layer, will be made based on experimental performance of the model. One of each layer, each containing the respective lower number of nodes, will be used at first. If the model is underfitting, another LSTM layer will be added. If it still underfits, another fully connected layer will be added. The various combinations of layer sizes will be experimented with to see which result in optimal performance.

This model will be trained using gradient descent with mini-batches of 128 days. A decaying learning rate, initialized at 0.1 will be used. This initial learning rate may be adjusted downwards based on expirimentation.

## Citations

1. McWhinney, James E. "A Simple Overview of Quantitative Analysis" *Investopedia*
2. Salmon, Felix and Stokes, Jon "Algorithms Take Control of Wall Street" *WIRED*
3. "Dow Jones Industrial Average" *Wikipedia*
4. User Aaron7sun "Daily News for Stock Market Prediction" *Kaggle*
5. Olah, Chris "Understanding LSTM Networks" *Colah's Blog*
6. Taylor, Bryce "Applying Machine Learning to Stock Market Trading" *Stanford.edu*
7. Fred, Ana and De Marsico, Maria "Pattern Recognition Applications and Methods" *International Conference on Pattern Recognition*
8. Lai, Siwei, et al. "How to Generate a Good Word Embedding?" *arXiv*