**Machine Learning Engineer Nanodegree**

**Capstone Proposal**

Vadzim Dabravolski   
June 1, 2017

**Proposal**

In my capstone project, I’d like to explore a possibility to forecast stock price trend using novice machine learning techniques (such as word vector embeddings and Recurrent Neural Networks).

The business problem is certainly not new and many smart people approached it before me as well as are trying to solve right now. A significant analytical and math apparatus was developed along the way. However, as of now there is no uniformly accepted best approach. And while data mining & machine learning are certainly giving new opportunities to predict stock movement, many stock traders are still relying on people skills.

I’m skeptical if this project can turn things around, however, I still feel that this may be an interesting exercise. As my current full-time job is closely related to investment banking and stock trading, I’m looking forward to apply skills which I hope to gain to solve more narrow and practical use cases in future.

**Domain Background**

There are several most common approaches to do stock market predictions ([wiki](https://en.wikipedia.org/wiki/Stock_market_prediction)):

* **Fundamental analysis** involves experts who analyze the company and industry performance in comparison with others. This type of analysis is more focused on the company (performance indicators, strength and weaknesses of leadership, market share dynamics etc.) than on the studying the stock behavior itself. The idea behind is to find over- or under-valued companies on the market and leverage this knowledge.
* **Technical analysis** trading school tries to predict stock movements based on certain observed trading patterns and behavioral dynamics (“head and shoulders” strategy, stock momentum, mean reversion and others).
* **Data mining** which to some extend combines all available pieces of data about company, stock and market, and trying to capture hidden patterns and connections between different pieces. Various datasets and technical approaches have been suggested (see Reference section for some examples).

In this proposal, I will use data mining approach. Based on my domain knowledge I hypothesize that in most common case following factors influence on the company stock price:

1. Overall stock market trends (“bullish” vs “bearish” market);
2. Trends in specific industry sectors (e.g. decline of classical retailers such as Sears, Macy’s because of Amazon);
3. Known and hidden trading patterns on the market;
4. Companies fundamental parameters reported quarterly (such as profits, revenue, cost of revenue, market capitalization etc.);
5. Public sentiment of specific company (e.g. public company bummers, M&A, rotations of C-staff etc.);

Of course, the market can be manipulated using one of the known illegal schemas (like “pump and dump”) as well as trading based on the insider information. For the sake of my project, I assume that these manipulations are neglectful.

**Problem Statement**

Develop machine learning algorithm (or ensemble of algorithms) which will recommend whether to buy or sell stock price for given company based on publicly available data such as stock prices, company performance and company news. The algorithm will be trained for industry leading companies with high market capitalization.

**Datasets and Inputs**

My choice of the data sets is driven by hypothesized factors in the “Domain background” section and should represent these factors in some form:

* Trading patterns and trends are represented by historic stock prices (provided by Quandl);
* Public sentiment is represented by company and industry news (provided by Reuters, Google and Yahoo Finance);
* Fundamental parameters of companies are represented in Earnings reports published by SEC (available via this [API](http://usfundamentals.com/)).
* General company information is captured in company profile (available on Yahoo Finance)

**Solution Statement**

My solution will solve the binary classification problem: for a given company to predict whether the stock price will grow or fall on next day. This should allow traders to choose between short position (if stock is believed to decline) and long position (if stock is believed to grow).

The envisioned solution has following simplifications and limitations:

* Only fact of increase or decrease of stock price will be forecasted (not the size of the increase/decrease).
* The intraday movements of the stock won’t be considered.
* The price at market close will be assumed next day market open price.

From implementation perspective, the solution is an ensemble of machine learning algorithms which designed to address various factors:

* Public sentiment and its influence on the stock will be modeled using Convolution Neural Network (CNN) and word (or document) embeddings.
* Fundamental company parameters will be models via CNN.
* Stock trends will be modeled using Recurrent Neural Networks (RNN).

Each component of the ensemble will have binary output with certain degree of certainty and will vote for overall decision.

**Benchmark Model**

Historically, the most commonly used approach for stock model validation is backtesting, i.e. validation of model is happenings based on the past stock performance. I think the same approach can be used in my case as well.

In addition to backtesting I’d like to use ARIMA model, which is used for pure time series forecasting. This model is rather generic and didn’t account for any information other than structure of the time-series itself. The implementation of [ARIMA model](http://www.statsmodels.org/0.6.1/generated/statsmodels.tsa.arima_model.ARIMA.html) is available in statsmodel Python package.

**Evaluation Metrics**

Since we are solving classification problem, it’s useful to visualize incorrect and correct predictions using contingency matrix:

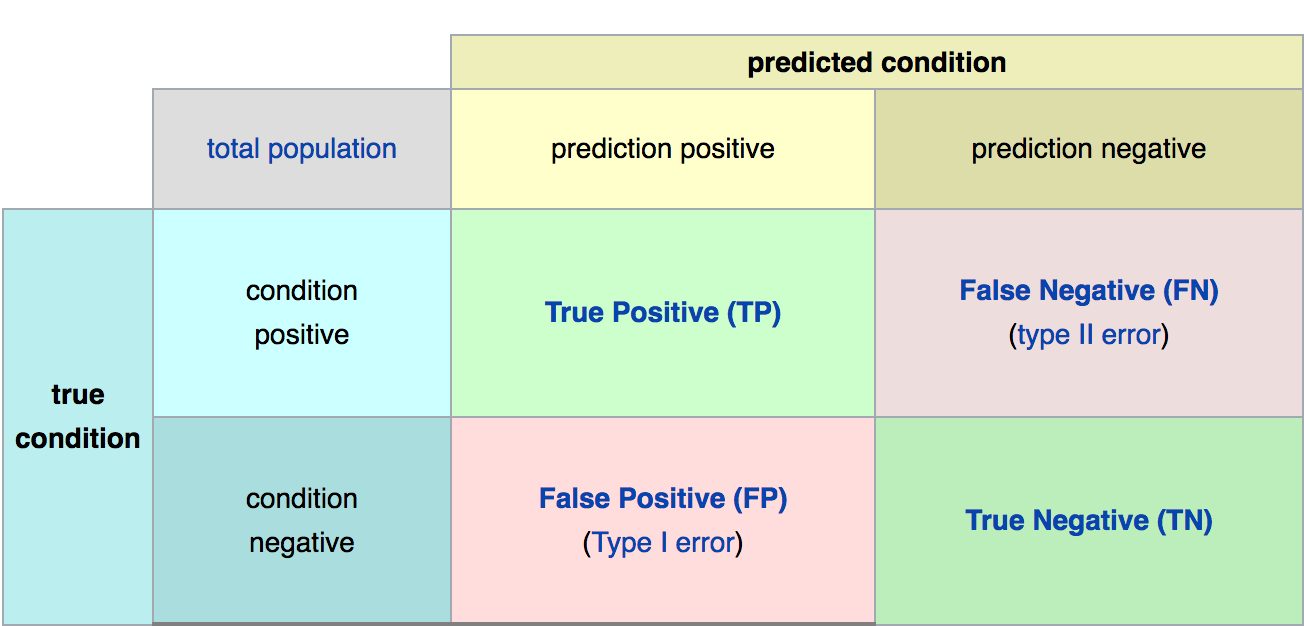
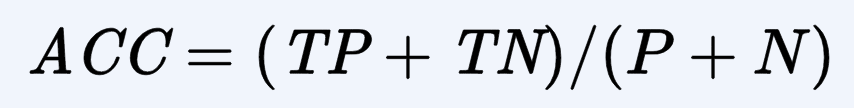


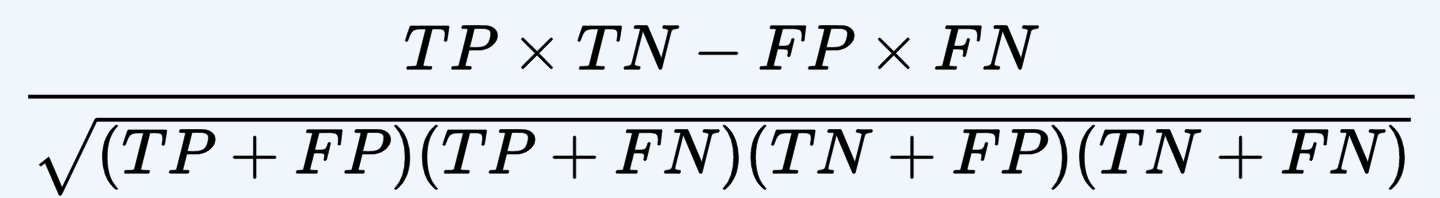
Figure 1 – Contingency matrix ([source](https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers))

Given contingency matrix the following metrics can be useful for evaluation purposes:

* Binary accuracy. It’s the most intuitive metric, but it performs poorly if we have not balanced classes (which I believe is the case of market stock which generally grows), so should be used with caution. Accuracy lies within [0,1] range.

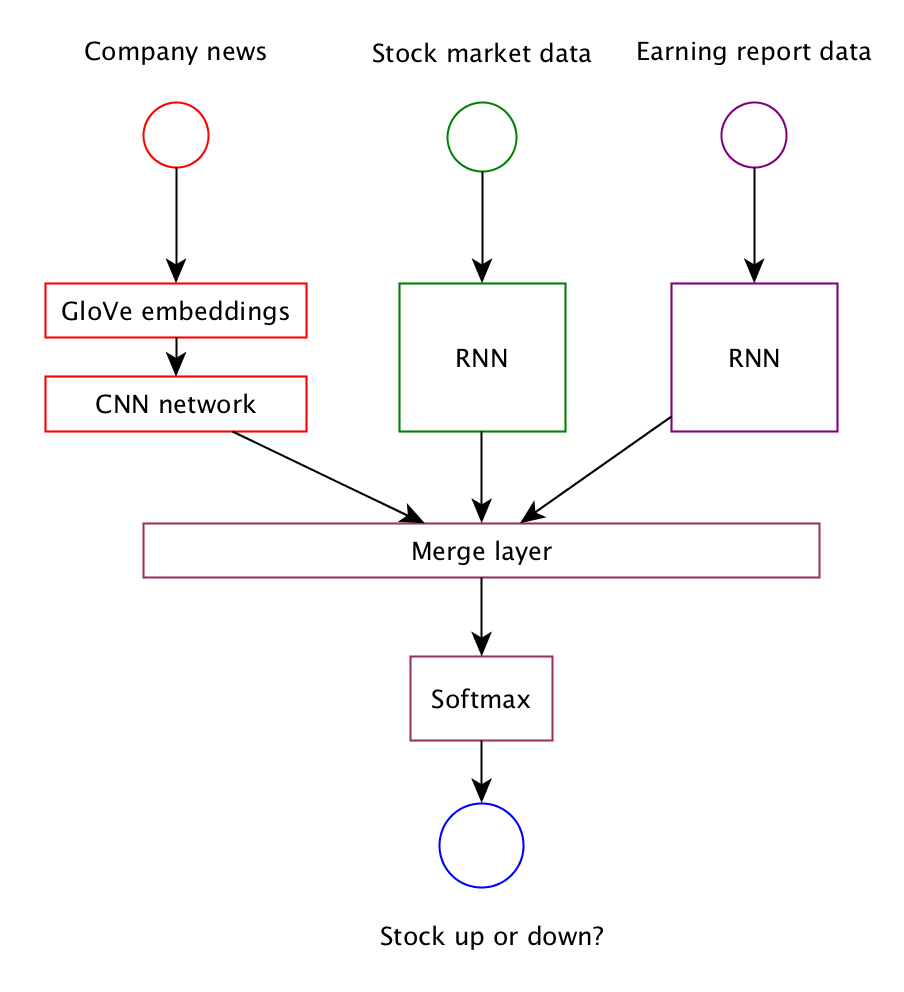


* Mathew’s correlation coefficient (MCC). It addresses the flaws in accuracy and can be used for cases of imbalanced classes. MCC lies within [-1, 1] range.



**Project Design**

As I mentioned above, the idea behind this proposal to build a complex machine learning model which will account for various factors which influence on stock movements. As the different factors has very different dataset (different sources and internal data structure), I propose to select a several separate models which will be assembled together to make a final prediction.



**Model selections**

* **Word embeddings.** I spent some time choosing between word2vec GloVe embeddings. Based on some of the published cases, both embeddings seems to have similar performance.
  + Using pre-trained GloVe weights (trained on wiki data).
  + Train GloVe on selected company news. The reason why I may choose this option is that I’m expecting some confusion in embeddings in case such as “Apple” (company) and “apple” (fruit).
* **CNN for company’s news feed**. I considered this model for following reasons:
  + Faster comparing to RNN.
  + As embeddings is effectively a 2D problem, CNN demonstrated effectiveness in working with such types of data (learning high-level features which in case of word embeddings can be associated with meaning of the phrase)
* **RNN for stock market data and earning report data.** 
  + Non-linear, robust to noise, can handle multivariate data.
  + Can learn the temporal context of the sequence. Avoid creating a fixed lag data sets.

**Technology stack**

To keep it simple, I will use later keras 2.0 as the main machine learning library. As of now it supports of the required functionality and yet provide an elegant API. I will use [GloVe](https://nlp.stanford.edu/projects/glove/) for word embeddings.

**References**

As part of work on this proposal I reviewed available literature and existing solutions. Some of them I found valuable, specifically:

* “[Deep Learning for Event-Driven Stock Prediction](https://www.ijcai.org/Proceedings/15/Papers/329.pdf)” by Ding et al.
* “[Market-Timing Strategies That Worked](https://www.kansascityfed.org/publicat/reswkpap/pdf/rwp02-01.pdf)” by Pu Shen
* “[Twitter mood predicts the stock market](https://arxiv.org/pdf/1010.3003.pdf)” by Bollen et al
* [“Machine Learning Strategies for Time Series Prediction”](http://www.ulb.ac.be/di/map/gbonte/ftp/time_ser.pdf) by Bontempi
* “[Financial time series forecasting with machine learning techniques: A survey](http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1113&context=infotech_pubs)” by Krollner et al.
* [“Deep Learning for Time-Series Analysis”](https://arxiv.org/pdf/1701.01887.pdf) by Ganmoa
* <https://github.com/edenbaus/Event-Driven-FinModel>
* <https://github.com/WayneDW/Sentiment-Analysis-in-Event-Driven-Stock-Price-Movement-Prediction>
* <https://github.com/llSourcell/How-to-Predict-Stock-Prices-Easily-Demo>
* <http://textminingonline.com/getting-started-with-word2vec-and-glove-in-python>
* <https://medium.com/@thoszymkowiak/how-to-implement-sentiment-analysis-using-word-embedding-and-convolutional-neural-networks-on-keras-163197aef623>

**Before submitting your proposal, ask yourself. . .**

* Does the proposal you have written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Solution Statement** and **Project Design**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your proposal?
* Have you properly proofread your proposal to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?