Software Design Final Report
December 17, 2013

Project Title: News Sentiment Analysis in Financial Markets

Team Members:
Marisa Mahlenkamp
Dara Behjat
Aaron Shapiro

The overall goal of our project is to create a program that takes historical news, measures it's relative sentiment and then tracks the correlation between the sentiment of the news related to a particular equity and the price of the stock. The model will take current news about a company and the most recent stock price, and show the correlation in a graphical output.

Our project can be staged in three parts:
1) Creating the two databases of stock price information and historical news for companies in question using select business sources
2) Filtering the news article through sentiment analysis programs (e.g. NLTK) and finding its correlation with the stock price change from when the article was posted.
3) Portraying the output in a clean and insightful manner. This has us using certain math packages and modules to make our outputs correlate and display in a graphical format (using Matplotlib).

Outlined below are different versions of the project that our group would work towards:

Minimum Deliverable: Finding the Data and being able to use it
Our minimum deliverable will be to download the news data from several different news sources as well as converting stock data into a usable format in Python. Once downloaded the data should be formatted in a way that it can be used in a sentiment analysis program.

Version 2: Storing, Organizing, Utilizing
The deliverable here is having the ability to store data from each article from each news source in a way that makes it easy to distinguish between articles, and to call or pull into a different programs. At this step, we also want to parse our articles of HTML so that we are only storing the text from the actual story.

Version 3: Simple Sentiment

The next version will be to implement simple sentiment analysis programs on our databases. Once modified, we would use the data returned from these programs to input into other simple algorithms. In particular, we would have an algorithm that found the correlation between the sentiment of the news and the company's stock price performance.

Version 4: Learning a Model and Incorporating Probability into the Model

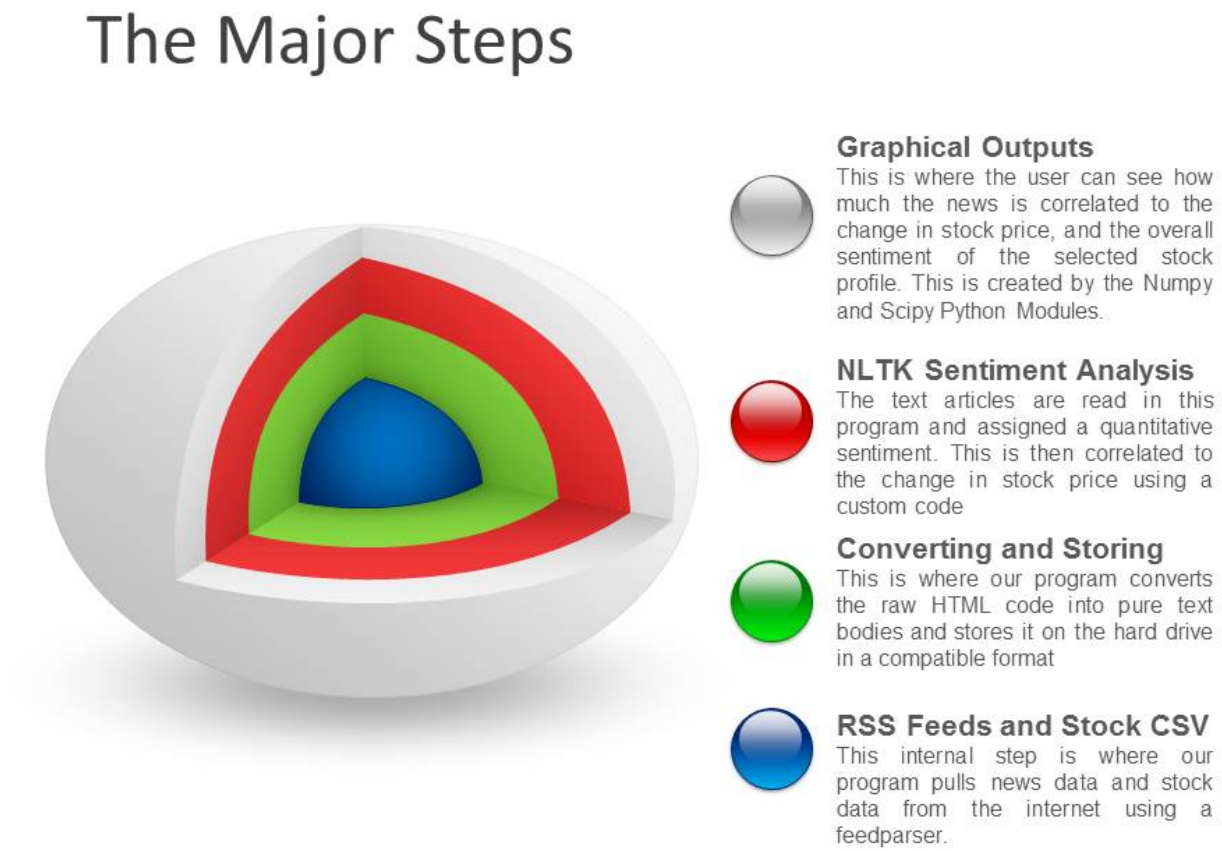
This version would incorporate more complicated sentiment analysis programs (e.g. working with NLTK) to improve our correlation between news and stock price analysis. Essentially, we would create a database of historical correlations between a specific company's stock price movements and public sentiment. As well as providing a clean and clear graphical output to explain the correlation.

Maximum Deliverable

The maximum deliverable will be a program that can evaluate the sentiment of news for any publically traded equity on a U.S exchange and use this data to predict the probability of price movement in the company's stock in either direction on the current day. Additionally, we would develop a user friendly platform (i.e. website) built from html and css to enhance the user experience.

First Step

Our first step was finding the best way to pull news and stock data in the most efficient and reusable way. We ended up choosing RSS feeds as a way to target relevant news from our sources and modifying an existing feed parser to get an output of URL's into our python program. We also modified a program that would pull stock data into an excel file and convert the data from XML to CSV which is a format that python can use. We then began researching existing natural language processing "black box" programs and finding one that is suitable to build the foundation of our project around. Specifically, we will be looking at existing sentiment analysis programs and finding programs that can be catered to our project. Once we have chosen an adequate foundation, we will have to organize our data so that it can be input into these programs.



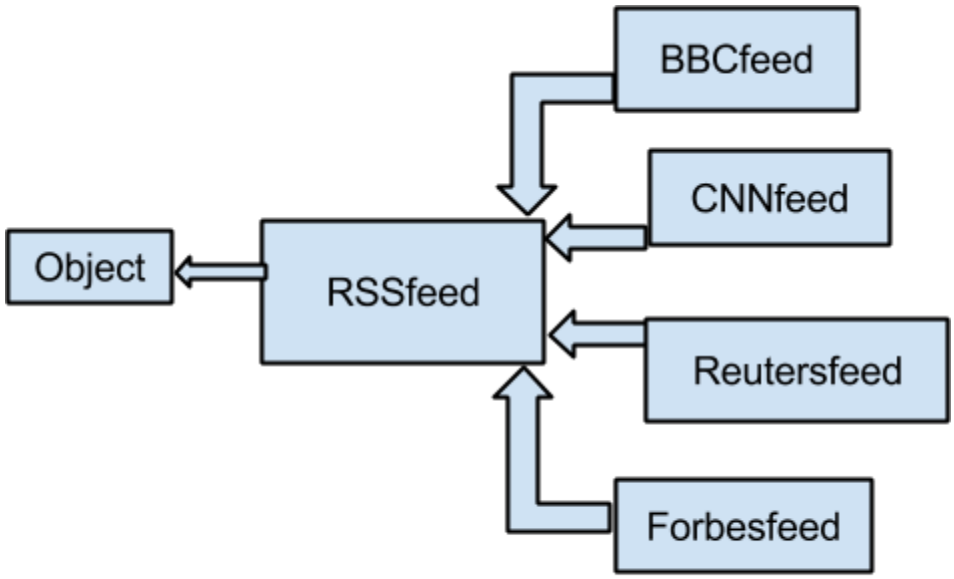
Biggest Problem

The biggest problem our team will face will be organizing the data into a usable format. Most likely we will input the specific websites where we would like our news data to be drawn from but then we will have to create a data structure that holds and synthesizes this news and that will also tracks its correlation with stock performance at time t. Once downloaded, the information will have to be parsed and organized with all the necessary variables.

The most current biggest problem, is finding complex correlation between our stock data and our news. This is the type of correlation we need to actually be able to predict stock prices for the user. All we are doing right now is simple math by assigning a quantitative value to the sentiment of our article and dividing it by the change in stock price. Ideally we would be able to use more advanced NLTK modules and linear regression to create an accurate prediction.

Implementation

For creating our two databases, we are using a number of classes to simplify and clarify our design. In our news database, there will be a class called RSSfeed(object) containing the code to pull the list of URL links from the RSS feed. Then depending on the news source, the list of URL links will be individually parsed and stored. This code will be contained in subclasses that are specific to each news source.



UML Class Diagram for creation of News Database

In our stock database, up until now there is no need to use classes because there is no need to differentiate our processes. The stock price data is downloaded from an external source (Bloomberg), converted into CSV format and then opened in Python. Ideally, in a more advanced version, we would have this information pulled from the web, giving us unlimited and instantaneous information but due to time constraints, our project did not allow for that.

Once we have started running our correlations between the two databases, there may be need for definition of additional classes but this will depend on the sentiment analysis program we use.

Vocabulary that is specific to our project includes: RSS feeds, stock price movements, html parsing, sentiment analysis, NLTK.

GitHub Repository link: https://github.com/DaraBehjat/SentimentAnalysis

Project Update, November 8, 2013:

Right now, we are working on creating our databases for our program. We will have two databases, one for our news data and the other for our stock price data. For our news data, we are working with RSS feeds and downloaded the module feedparser to read that code. As of right now, we have working code that pulls the RSS feed from the internet and returns a list of links to different headlines. We have a list of additional feedparser commands but will continue reading up on RSS feeds to figure out how to sort, read and store the information.

For our stock price data, we have created a file of 10 high-publicity stocks with its daily price information for the past month. We were able to have python read the original data from an excel file by creating a CSV reader in python. Our next step is to organize the data into a usable data structure (our current thinking is a dictionary) where the day (key) maps to the change in price of the stock (value).

Final Update, November 17, 2013:

We were able to achieve our minimum deliverable in this project. We wanted to download the news data from several different news sources as well as converting stock data into a usable format in Python. We wanted to have our data in a compatible format for whatever sentiment analysis we were looking to do. We were able to write a code that pulled RSS feeds for relevant news data from the internet in HTML format. We parsed the articles to make them large bodies of texts with no filler so that our sentiment analysis program in NLTK could import the article and output the relative sentiment. We also were able to pull stock data and convert it to CSV which is the compatible format in Python.

We were not able to reach our maximum deliverable which could evaluate the sentiment of news for any publically traded equity on a U.S exchange and use the data to predict the probability of price movement in the company's stock in either direction on the current day. We also said the program would have a user friendly platform (i.e. website) built from html and css to enhance the user experience. We were not able to accomplish the prediction portion of our program which is explained above, and we aren't able to evaluate the sentiment for *any* chosen publicly traded equity either. This would require some resources we don't have such as a powerful server computers and an automated code that would enable us to pull articles for every single publicly traded stock and a place to store the info beyond our hard drives. We also didn't have the time to build a website or a GUI interface that the users could use. That required us to learn many more styles and formats which we didn't have the time to do. Instead we used scipy and numpy to provide graphical outputs.

Overall, we accomplished Version 4 of our project proposal. Our program successfully incorporated NLTK into our sentiment analysis and used this generated sentiment to look at its correlation with stock prices for that specific day. Also, we included a simple interface for the user so that when they input their company, statistics are printed and visuals are generated.

Design Reflection

Although our project was quite complex, we chose to opt for a relatively simple user interface. Our code contained a lot of information but we didn't want to confuse our user and make the output too confusing. We chose to use some of the visualization tools contained in the matplotlib package to accomplish this. In the end of our project time, we chose to focus less on complicating the "back end" and pulling the different parts together in a working program.

Design of code:

Modules written:
rss_reader_classes.py: generates our news article database (and stores information about each article in read_articles.csv)
stock_reader.py: creates database from csv file of stocks/stock_data_12_14.csv
article_indices.py: creates indices to pull together and hold information about news database
analyzer1.py: contains the NLTK model and functions to evaluate the sentiment of each article
(**sentiment_vocabulary.py**: creates the extensive vocab lists for NLTK)
pull_together.py: program that pulls together all modules and generates output for specific company that the user inputs

Labor Reflection

At the beginning of the project, we were able to divide the work for the program in such a way that each team member had a specialty. One team member focused on news data, one on stock data and another on natural language sentiment. Yet, as the project progressed , each segment of the code had to be pieced together. As such , some team members would be faced with an increasing amount of work, while the others could not move forward without that work before that work could be done. At large, we chose to divide the labor for the project in the most efficient manner possible. While we may have been able to learn more if we had divided the work in a different manner, the way we chose to divide the work ultimately lead to the best possible outcome.

Bugs

Given the complexity of our program and our relative inexperience coding, we certainly encountered quite a few bugs. However, of all the bugs we encountered, the biggest bug was not even in the code. More specifically, the "bug" was simply not having all the proper packages on all of the team member's computers. As we began to compile each segment of code, we came to realize that in order to run the code we would need to download the packages that the team member used in their development iterations.

In retrospect, before we began coding we should have done more research to generate a list of all the packages we would need to complete the project. After completing the list, each team member should have made sure that they had each package on the list properly installed on their computer. Any additional packages being used should have been communicated to the rest of the team members.

Our team also would have saved a lot of time by choosing to explore some of the available packages that can be used for natural language sentiment prior to beginning to write our code. At times, we spent an incredible amount of time to develop code that solved problems that could have been solved by some of the available python packages including NLTK.

Other limitations of our code include the accuracy of the correlations we were able to find. We were not able to extract specific correlations because we did not have time or the necessary knowledge to incorporate regressions into our model and graphical outputs. In terms of pulling data, many of our articles were published on the weekend and since the market isn't open on the weekend, we couldn't directly correlate a specific stock change to news over the weekend. To alleviate this issue, we chose to have it be represented by a 0% change and made sure to show the whole range of dates in our output. However, we did not have time for our code to incorporate mathematical modules (StatPy for example) that would give us exact regressions.

In terms of NLTK, our sentiment vocabulary was very adequate but could have been extended to increase accuracy. One limitation however, was that in order to have more accurate sentiment analysis, we needed to import the entire NLTK vocabulary list which made the program very slow and hard to demonstrate. For future reference, we would want to find another way to be able to import our vocab into the NLTK program such as a dictionary as opposed to a list.

We were also limited to using news sources that published their articles in the same format so it could be easily parsed in an HTML parser loop. We had an issue with searching companies that had more than one word because articles were indexed by single word count. Lastly, our program cannot be run in real-time since our stock data is all historical. Our news was pulled instantaneously but the stock needed to be downloaded from an external source and input into the program. With more time, we could have written an additional module that pulled real stock data, and looked at its correlation with news articles published on that day to generate a prediction.

At large, developing our financial news sentiment analyzer proved to be a valuable learning experience. The complexity of the program required us to master some of the foundational skills of software design that we learned in the class as well as learn completely new things. Although there were certainly times of despair, we were able to build a program that far surpasses the minimum deliverable we established for ourselves. Thus, this program is not only something we are excited to see “come to life”, it is also something we are all quite proud of.