BestBET: Final Report

Group Members

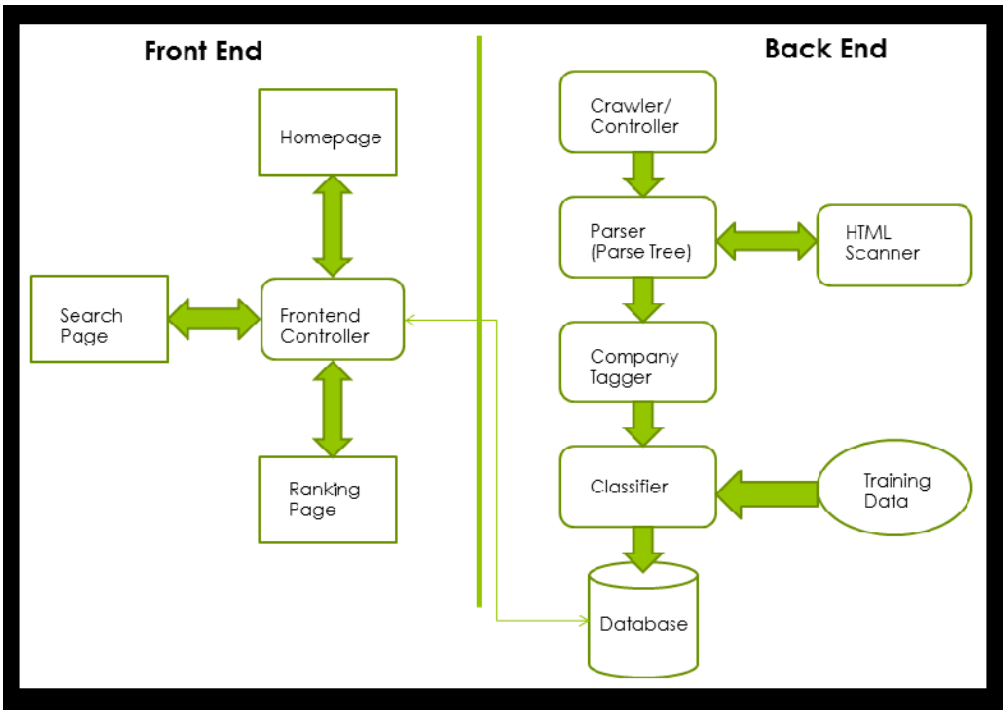
- Saptarshi Bhattacharya
- Neha Gaur
- Yamir Godil
- Isamu Mar
- Abdul Salama

Project Goals

The objective of this project was to make it easier for stock investors to interpret the immense amount of data available online. News is a major driving factor of the stock market. The focus of our project was to utilize this data in order to understand the bigger picture of the stock market and its trends. We wanted to build a tool that would classify news articles from various web sources, with respect to companies, and present the results in a coherent manner. The crux of this project lied in applying machine learning and sentiment analysis techniques to a difficult problem and then presenting the results to the end user in a simple yet effective way.

System Architecture

The following is a highly accurate design diagram of our core features:



Following is a brief description of each of the major components in our system:

- **Crawler/Controller:** A crawler is utilized to fetch relevant web articles from various different web sources. It also acts as a controller for the entire back-end.
- **Parser/HTMLScanner:** A parser is used to create the Document Object Tree of the HTML page we retrieve from web sources. This is an essential component of the system, since it is necessary to extract the relevant information out of the HTML content. A typical web page containing an article, includes a lot of garbage data including but not limited to ads, menu items, user comments and links to other articles.
- **Company Tagger:** Once we have the article text along with the parse tree of the HTML page, it is necessary to find the name of the companies that are most relevant to the article. This component tags an article with one or multiple company names, which is later used in the classification.
- **Classifier:** It classifies an article into the three categories(positive, neutral or negative) with respect to a company. More on this later(Algorithmic choices).
- **Database:** It stores all associations between articles and companies, and the results from the classifier. It also stores data related to current stocks, which is being updated periodically. It serves as the only piece of connection between the front-end and the back-end.
- **Front-end Controller:** This serves as a host to respond to all requests, fetches the relevant data from the database, and then re-routes to the correct web-page.

Algorithmic Choices

Investors have an enormous amount of financial news available to them, but perhaps the most interesting are the articles that display a positive or negative sentiment about a particular company. We wanted our system to preform sentiment analysis to accurately pick these articles out and classify them as either positive or negative. This gives us three categories for classification in total: Positive, Negative, and Neutral.

For the sentiment analysis, we wanted to use an algorithm that could be implemented quickly and could easily be extended. Naive Bayes classification fit perfectly.

The most intuitive extension of Naive Bayes was removing common words that appear in all types of articles. Proper nouns and numbers do not have any polar context either, so those were removed from training and classification as well.

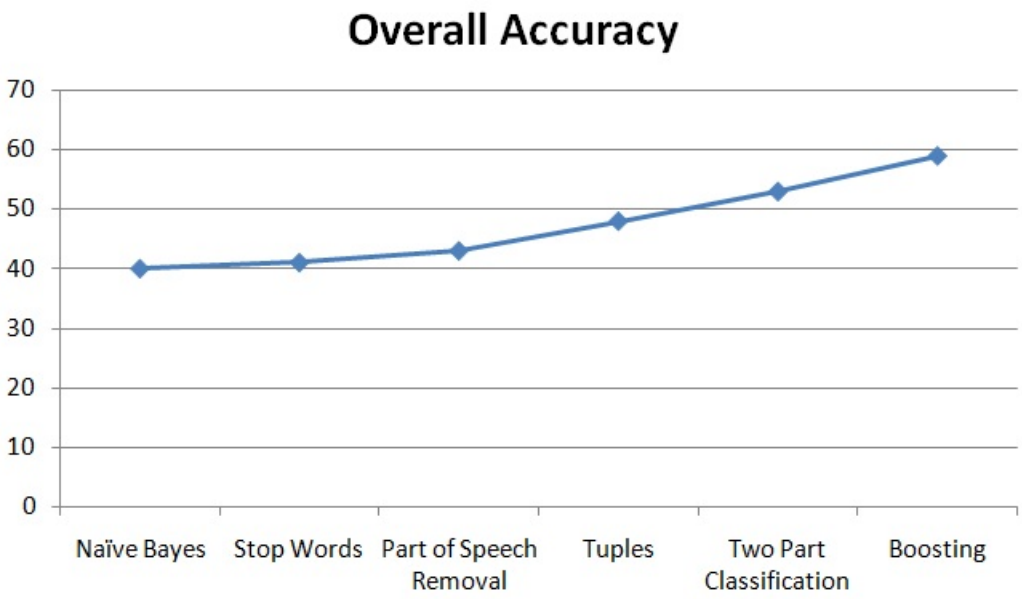
Sentiment is difficult to extract from independent words, so we introduced the notion of Tuples. Instead of looking at each word individually, we looked at Tuples of adjacent words, to extract more context out of the article.

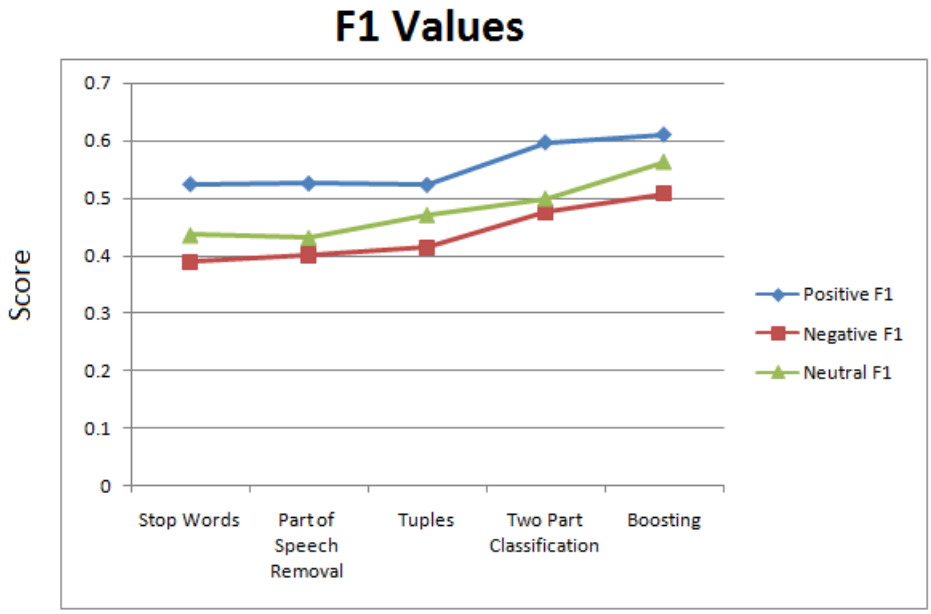
In reading “Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis” (Wilson, Wiebe, Hoffman), the writers encountered a similar problem that we were having; large amounts of non-neutral articles being classified as neutral. Their solution was to use two way classification, first classify between neutral and non-neutral, then classify the non-neutral articles between positive and negative. This technique of Two Way Classification allowed us to build a more precise classifier for neutral article removal, which could be piped in to a stronger positive vs negative classifier.

Finally we used boosting to strengthen our mildly effective classifier into a much stronger classifier.

Experiments & Results

All results are from 8 way crossvalidation testing, where each subsequent test contained the previous’ techniques. i.e. The Tuples score also includes part of speech and stop word removal.



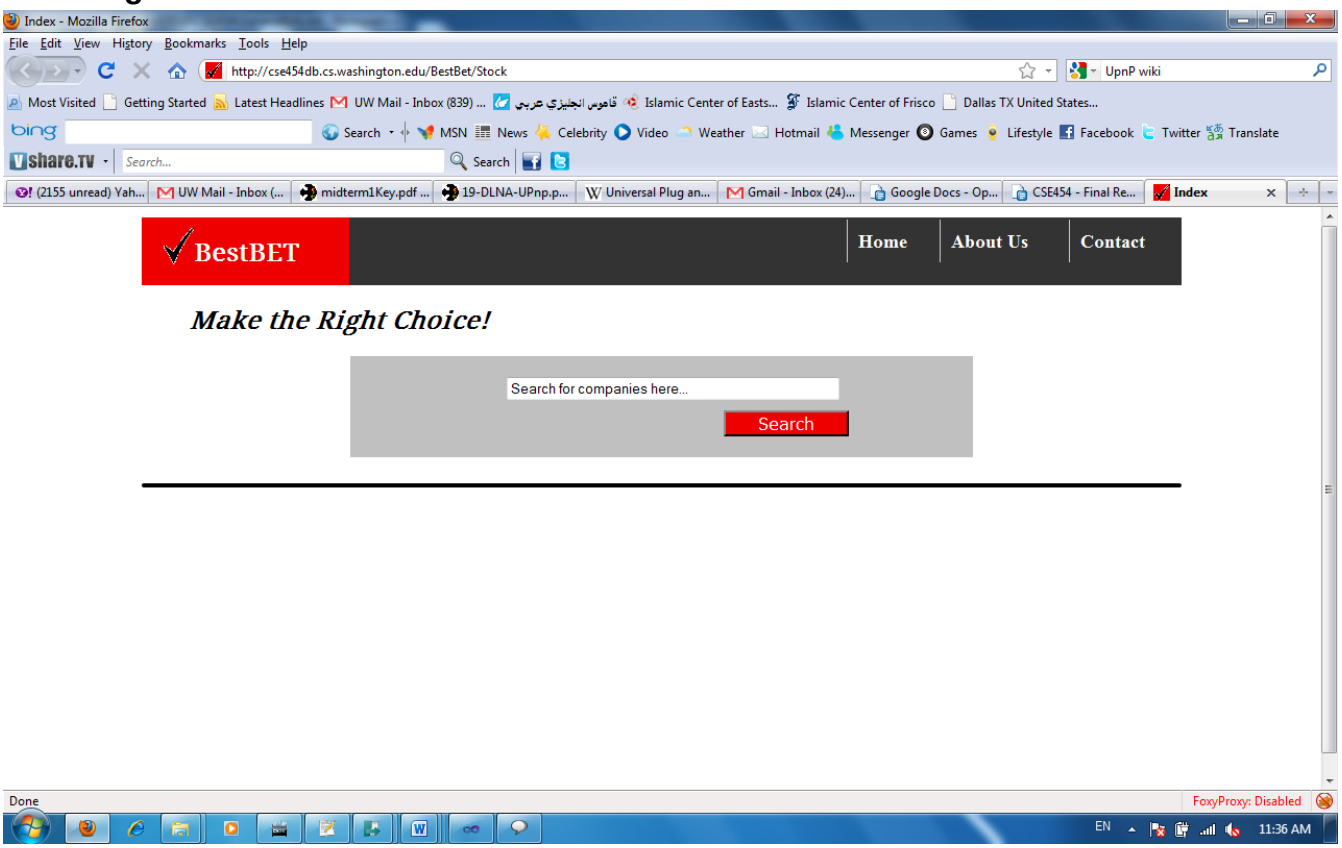


It is clear from the F1 graph that classifying negative articles correctly was the most difficult. This may be attributed to a deficiency of negative articles in our training data. A greater amount of robust training data is a simple improvement point. A more complicated improvement could involve natural language processing to more effectively extract the context of each sentence.

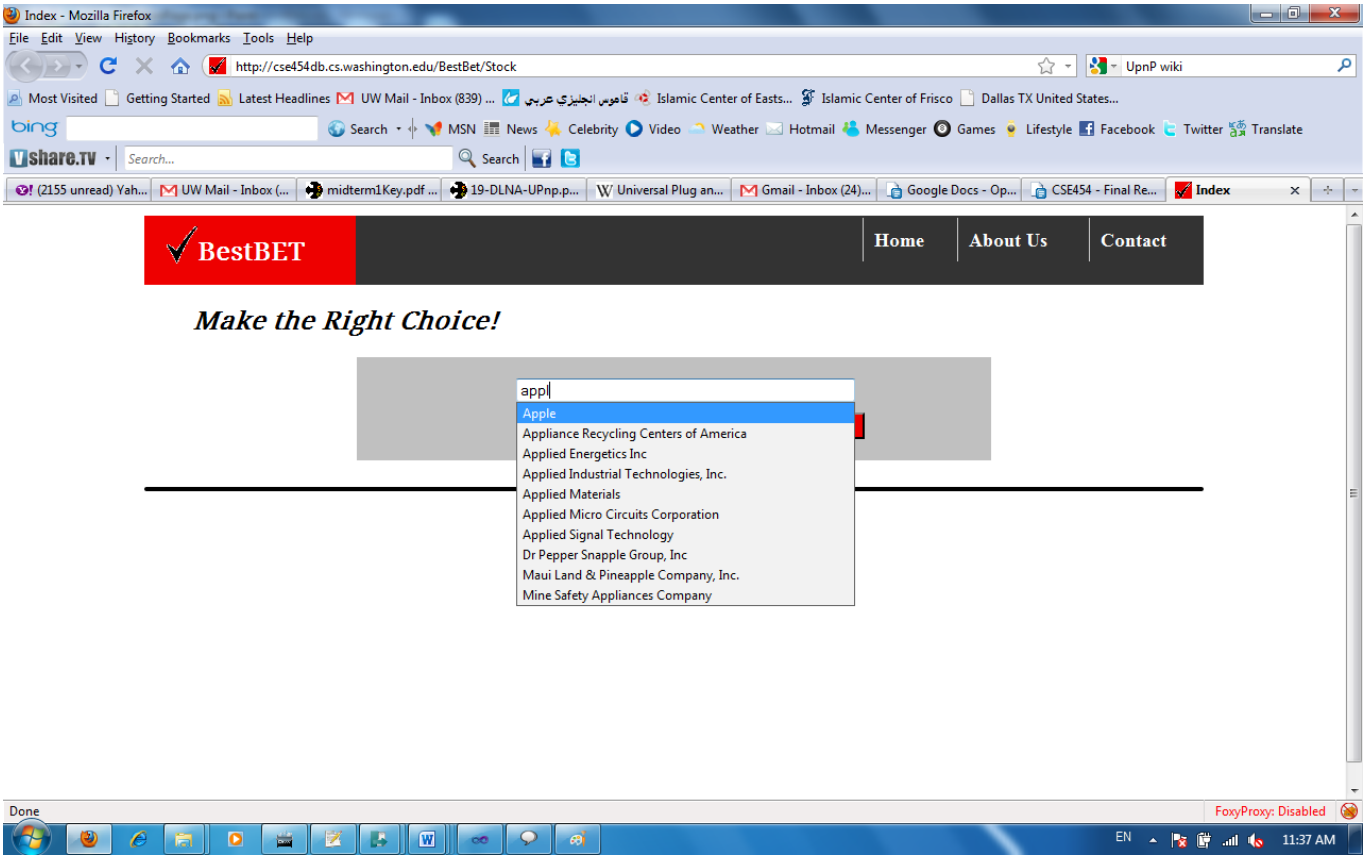
Usage Scenarios

The following snapshots represent a typical usage scenario for our product:

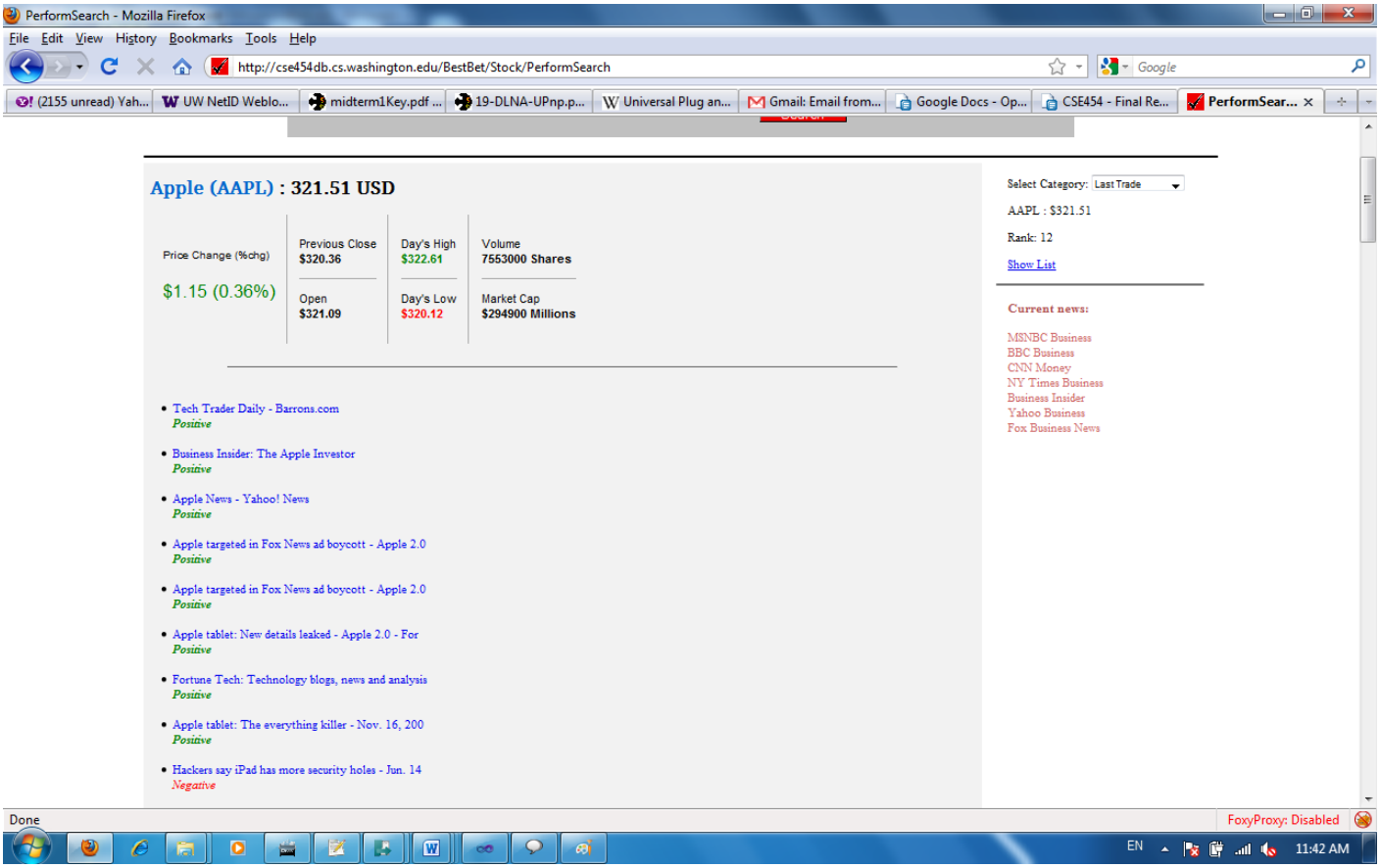
Main Page



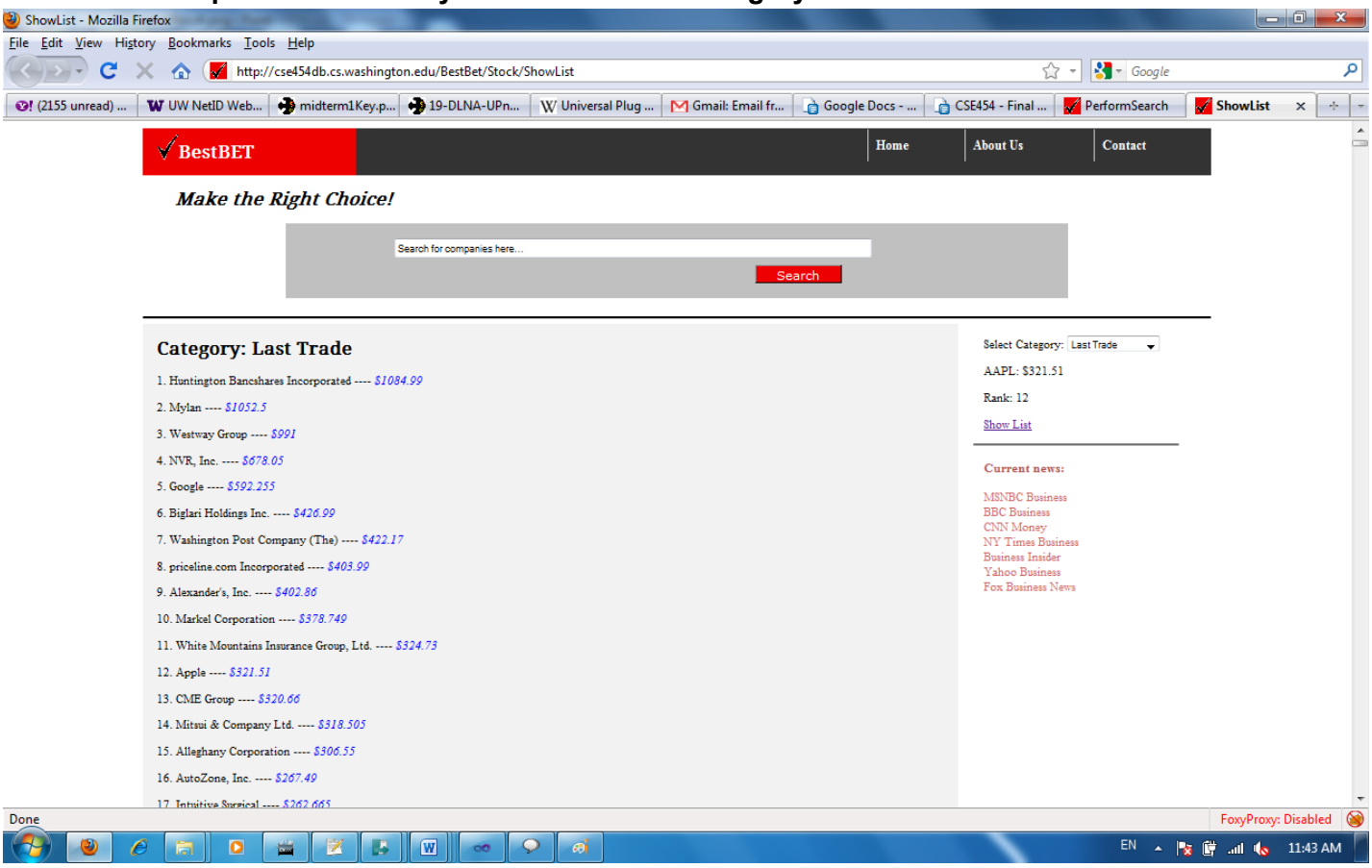
The auto-complete feature as the user searches for the company “Apple”.



The result page for the search query “Apple”. It shows some dynamically extracted financial stock data for “Apple”. Below that are crawled and processed news articles related to the company “Apple”. The panel on the right shows the ranking of this company based on the “Last Trade” category.



A list of all companies ordered by the “Last Trade” category.



User Study

Our user study was on a very small sample of a very biased population. We tried to incorporate feedback of our colleagues within the CSE department. Below are the results that we received.

We asked these questions to about 20 students about the usability of our website.

1. Does this classification help you?
Yes - 40%, No - 5%, Can't Tell : 55%(not enough experience in investments)
2. Does our stock information with news classification help you invest?
Yes - 45%, No - 55%
3. Does the news classification influence your decision in any way?
Yes - 90%, No - 10%
4. Our web-interface that can be improved(Improvements mentioned below)?
Yes - 95%, Maybe - 5%

User Suggestions on Improvement:
- We should allow for searching for companies who are having high percentages of positive articles, and negative articles.
- We should allow to search by sector/industry.
- We should try to make recommendations based on recent news.
- Stock information just makes it easy to understand if you have the budget to invest.
- If classifications are not accurate, there should be an option to have user input and feedback.
- Have visual(graphical) representations of stock information.
- Have a comparison tool, that compares graphs of 2 companies and compares how they are being portrayed recently.
- Have a section that shows all of the top performing companies in every sector.
- Website does help to try to get the overall picture about a company which helps in investment.
- Articles should have dates, so that we can tell how old the article is.
- Try to display the articles on our website.

If we had time, we could have interacted with economics professors and students and get their opinions. Since the user studies were done late in the quarter, we did not get a chance to implement any of the suggested features.

Surprises

During the whole quarter there were a lot of things that we did not expect to happen. Following is a brief summary of these surprises :

- Google API was being suddenly deprecated during the middle of the quarter.
- Web-pages containing news articles have a lot of garbage content that was surprisingly hard to get rid off.

What would we do differently

If we were to do the whole project again, we would several things differently :

- Try to narrow down minor/major features early in the project, and have a well designed idea.
- Prepare a more formal design document; for example, define clearly the interaction of the database with a particular component. Currently, we just added functionality on the go, which perhaps could have been optimized if we had thought of the big picture earlier.
- Use a different programming platform. We used C#, but majority of us had trouble adapting to the new environment in such a short time for a big project, perhaps we should have used Java.
- Spending more time gathering training data would have been useful.

Possible Future Extensions

Given a very constrained set of resources, we could not implement all the features we wanted to. The following is a brief overview of some of these possible extensions:

- Use natural language processing to improve the recall/precision of our classifier.
- Use information extraction(or other machine learning techniques) to more accurately identify & get rid of the ads/comments/garbage.
- Provide the user with a confidence level for each of our predictions.
- Add the ability for users to provide us with an URL to a web article or some article text to instantly view the result of sentiment analysis for the given article.
- Ability for the user to enter if the classification was correct or not and based on the feedback, change the classification.
- In the UI, for each article analyzed for a company, present a preview screenshot of the article as a user hovers a mouse over the article url.

Conclusion

Although this project gave us the opportunity to gain a lot of technical knowledge, we strongly believe that the soft skills acquired from this project are indispensable. Working in a group, to transform a simple idea into a wonderful product, is what we have learned from this project.

Appendices

- **Division of Labor(Technical components only)**
  Saptarshi: Crawler, Parser/Scanner, Pipe-lining
  Neha: Front-end/Pipe-lining
  Yamir: Database Implementation, Company Tagger
  Isamu: Major responsibilities of Sentiment Analysis and Classification algorithms.
  Abdul: Server Setup/Maintainence, Front-end, Crawler

- **Group Dynamics**
  The team worked quite well on the project. We strongly believe that responsibilities were evenly distributed among all members, and everyone met each other's expectations. All differences in opinions were soon resolved through group meetings and debates.

- **Externally-written code used in the project:**
  1) Part of Speech Tagger
  2) Bing Live Search API

- **Instructions for setup:**
  Our product is hosted live on the cse454db.cs.washington.edu server. To visit our product, use the URL "http://cse454db.cs.washington.edu/BestBet/Stock". Supported browser is FireFox.
  To build and run our code internally, you need Visual Studio 2010 (VS2010) installed on the machine along with required setup for running IIS processes (usually installing VS2010 will take care of setting up the machine to run IIS processes).
  Unzip the submitted BestBetCode.rar folder. Double click on BestBet.sln. It should open the project in VS2010. Building and running the project in the VS2010 environment

should be trivial. Just click “Build Solution” from the “Build” menu, that should build the whole project. To run, choose either “Start Debugging” or “Start Without Debugging” options from the Debug menu.