Project Name:

Team Members and emails:

- Ankit Gupta: ankitgupta@college.harvard.edu

- Hirsh Jain: hirshjain@college.harvard.edu

- Nishant Kakar: nkakar@college.harvard.edu

- Emily Wang: emilyluwang@college.harvard.edu

Brief Overview:

Our team wants to use sentiment analysis to parse and analyze news articles in order to identify their polarity. In particular, we want to identify the degrees of partisan bias within a given source text. Current work on sentiment analysis attaches either keywords ("happy" "sad" "angry") or a certain number on a scale (-10 to +10) to a given text. We plan to use the numerical scale to measure partisanship, as there are primarily two poles (conservative and liberal) with gradations in between. In addition, we're interested in using the aggregate polarity of a news outlet (for example, nytimes.com or reuters.com) to see if the news outlet on the whole has a tendency to publish articles with a particular partisan bent. Another potential area of interest would involve looking at news coverage of a certain issue (for example, the Obamacare rollout or United States sanctions on Russia) to see if certain issues gain particular traction within liberal or conservative media outlets. An interesting outcome of our project may be a list of politically-charged keywords, ie certain words more frequently used by conservative sources than liberal sources, or vice versa. There currently exists a large body of work studying methods of sentiment analysis, including various courses and studies that have looked at political speeches. Consequently, we're hoping that the initial groundwork should be fairly straightforward, and the bulk of our remaining time can be spent exploring cool potential applications.

Feature List

We have a list of features we would like to implement, and they vary in computational complexity and time to implement. Moreover, certain ones of them are more vital to the core aspects of our project. The first feature we will need to implement is to take a file, most likely one in HTML that a website article would be in, and be able to convert it to plaintext for our parser to read. We expect this to be fairly straightforward, since there exist technologies that can assist in this process. Then, we need to build a means of parsing this data into some sort of a data structure for storing the words, and building a model. According to our currently limited understanding of how to implement these relatively simple machine learning algorithms on sentiment analysis, this will involve some sort of graph-based model, and so we will need to have an appropriate data structure to represent it, such as an adjacency list with linked lists or an adjacency matrix as an array. Next, and perhaps most vitally, we need to implement the probabilistic models that allow us to do Bayesian analysis on the words that we parse, and use that to determine if there is any sort of preference for one party or the other in a news article.

These were the most vital aspects of the project. Now, if we can accomplish these steps in a reasonable amount of time, which we believe is possible but is definitely going to be difficult, we have a few extensions. First, we hope to build a web crawler, similar to one in Moogle, that allows us to crawl through the website of a certain news source, like NYTimes, and determine if throughout the website there exists any tendency towards a certain party. This could lead to some very interesting insights. Similarly (and optimistically), we may also build a web crawler that can crawl through Google search results on a certain news topic to determine if coverage of the topic itself skews liberal or conservative.

Technical Specification

Research on sentiment analysis (including the various subcomponents of it - natural language processing, text analysis, computational linguistics) already exists, with certain algorithms already available. We plan on implementing our own project using the text classification process detailed in Fabrizio Sebastiani's paper "Machine Learning in Automated Text Categorization" with Naive Bayes classification. Then, we intend to use the information available on Stanford's Coursera online class on Natural Language Processing, as the material builds upon text classification and Naive Bayes classification. This project on analyzing Presidential speeches to identify partisanship within the prose also details multiple approaches that may be useful.

We see our project as consisting of a few key modularized parts.

- First will be the tool to parse information from an HTML site and be able to graphically represent it (in terms of an adjacency node relating connections), as we suggested earlier.
- If we choose to build a web crawler that can crawl through a certain news website and find all of the pages to parse, we will need a module that can do this.
- We need to then build our Naive Bayes classifier using the information that we gained, and using the input words that we provide.
- The functions we will need between modules are those that allow us to access the graph, and then conduct the Naive Bayes analysis. So, to better understand exactly which functions we need, we will need to first better understanding how Bayes analysis works.
- Also, we will need to take advantage of the fact that there exists certain libraries that exist for Python and other languages we are interested in using that allow us to easily do web crawling, and we will need to implement methods that use those functions.

Next Steps

We currently have a strong understanding of what we specifically want to accomplish but haven't addressed how exactly we wish to do so. This is the direction we look to move forward in before the next checkpoint comes around. We are fairly certain that we will want to program this project in python due to the language's integrated ML oriented libraries. Using python will also allow us to access its machine learning libraries and use these tools to better implement our project. However, solidifying this decision is key for us to make any further progress on this project, so it will definitely require some more thorough research. Before the next checkpoint, our team will more comprehensively analyze the various languages we could use to implement our idea and surely choose the most efficient and fitting one. Another tool we may want to use for our project is a data-scraping tool to get information off of the web. We haven't decided on a specific library that we could use for this, but plan on doing so before the next checkpoint. Once we have scraped this data, we plan on using a parser tool, such as Beautiful Soup, in order to intelligently iterate over and analyze the information we have gathered. Additionally, our group will have a better idea regarding which interfaces we wish to incorporate in our sentiment analyzer. Once these technical questions are answered, we will also have set up a developing environment that not only works on our individual computers but also allows us to share and update our code with each other in an efficient manner. This sharing will most likely be done through git and the environment setup will be done once we decide on the exact language and framework we plan to use. Overall, we're feeling good about this project and are excited to see what's in store for us to learn!