Overview

Purpose of This Report

This report serves to update the reader on the progress our team has made on TL;DR. In particular, the report will go over what we have finished, what we have left to do, and our future plans for our project. In addition, the report will also detail major changes we have made to our project.

Purpose of Our Project

The purpose of our project is create a web application that attempts to answer questions asked by users on different articles on the internet. By having our web application read the document and try to answer it, we save the user valuable time. For example, suppose a user doesn't have time to read Facebook's privacy policy but wants to know if Facebook receives data from devices used to access Facebook. He or she would go to our website and type in the question "Does Facebook receive data from devices used to access Facebook?" and our website will attempt to answer that question. If the answer is what the user is looking for, then we have just saved them from skimming through Facebook's privacy policy.

Literature Review

After the completion of the natural language processing module, we performed some evaluations on different articles such as Facebook's privacy policy and Wikipedia articles. We realized that how the articles are organized is important in trying to parse it. For example, a lot of crucial information is conveyed in the privacy policy through bullet points, something which our system does not pick up on. Thus, we conducted additional research on privacy policy and EULA agreements of Facebook, Google, Apple, and Blizzard. This helped us refine our NLP system by revealing common patterns with these documents such as using structure to convey relationships between terms. The most important additional aspect of NLP we've discovered is semantics. For our NLP system, we need to find out in what sense a word is used. For example, "play" can mean "to manipulate" or "to participate in a game". We are attempting to solve problems with semantics within our project with Princeton University's WordNet semantic database. Thus, our system is limited by the scope of WordNet's API, which does not have any good libraries for disambiguating word senses. This influenced us to write our own word sense disambiguation code.

Progress

We have finished two of the three systems we outlined: web parser and NLP backend. The web front end is mostly done.

Front End

We are currently in the process of developing a front end interface using Ruby on Rails that integrates the other two core components of the project into an interactive, polished web application. The front-end development is currently in its integration and development stage--we are experimenting with the rubypython gem (a library in Ruby) in an attempt to integrate the parser and NLP component that have been built in Python. Our application will first take in the name of a company to investigate and a question regarding the privacy policy from the user in an HTML form and pass the input information along to the Python web scraper to get the necessary information from the web for the particular company in question. Next, once the application has gathered the appropriate information from the web, it will pass along the information from the web pages to the NLP component, along with the question from the user from the HTML form, so the NLP component can then attempt to process the query. Then, the application will display the information returned from the NLP component in an HTML page that contains Embedded Ruby tags (ERB tags). These Embedded Ruby tags will show the result of evaluated Ruby code in an HTML page. Despite the fact that our components are built in Python, we are hoping to use the rubypython library to be able to call Python functions in Ruby so that the ERB tags will be able to display the results generated from the core components of our application. The aesthetic portion of the web application is currently being developed using Foundation, a front-end framework that integrates CSS and Javascript into an easy-to-use framework to build appealing applications.

Parser

We have implemented a simple version of the web scraper and parser in Python. Using the BeautifulSoup library, we parse an HTML privacy policy document into a tree which we need to traverse. Out of this tree, we try to extract each individual bullet point or sentence. However, in some cases, a single bullet point may not give enough context to be a useful answer to the user. In this case, we try to find a header or sentence preceding a bulleted list which puts the bullet point in context. Even if we find a complete sentence, having context such as the header the sentence belongs to can help the question answering sentence return a more relevant result. We can find a header which uses an HTML header tag, but there is enough variability between different HTML documents that we may miss some headers.

Question Answering

We have built a working NLP question answering system which when given a text article and a question about the article will try to find the sentence in the article which best answers the question. We use an algorithm based on the term frequency-inverse document frequency (TF-IDF) statistic to solve this problem. The system ranks sentences which have the most number of words overlapping with words in the question. We assign a weight to each sentence in the article. For every word in the sentence that also appears in the question, we add to its weight. The sentence with the highest weight is considered the answer. TF-IDF gives lesser weight to frequent words like "the", "of", etc. This approach works relatively well if the question is very specific and the answer can be found within the passage without logical deduction or worldly knowledge (i.e. knowing information outside of the article). However, when presented with a question which doesn't contain any unique keywords such as "Did John agree?", the system sometimes has difficulty pinpointing the section of the article where the answer would be located. Additionally, the system fails for questions that have the answer in the passage but use different wording (such as synonyms). The question answering system also assumes that any question inputted by the user will have an answer in the article. Thus, the system will return nonsensical results when presented with a question which doesn't pertain to the article at all.

Status

As of right now, we have a working parser, NLP system, and a almost working webpage. However, these components are currently not connected. Additionally, the question answering system still has room for improvement.

Projections

Our project is on schedule. However, the accuracy of our NLP system is not up to our expectations. We have a few ideas to deal with this problem, namely relying on semantic word similarity instead of exact word matching. We hypothesize that word similarities may solve the system's inability to answer questions requiring worldly knowledge or logic. This will mean that we will devote more time to improving the NLP system. In addition, the web interface is extremely simple consisting of only input forms and textual output, so we will allocate time from working on web interface to working on the NLP backend. The last portion of our project is to integrate all of these parts together and make sure they interact with each other seamlessly. We estimate implementing word similarities will take about 1 week and integration 2 weeks. Thus, we estimate that our project will be finished within the next two weeks. Our revised Gantt chart shows these changes accordingly.

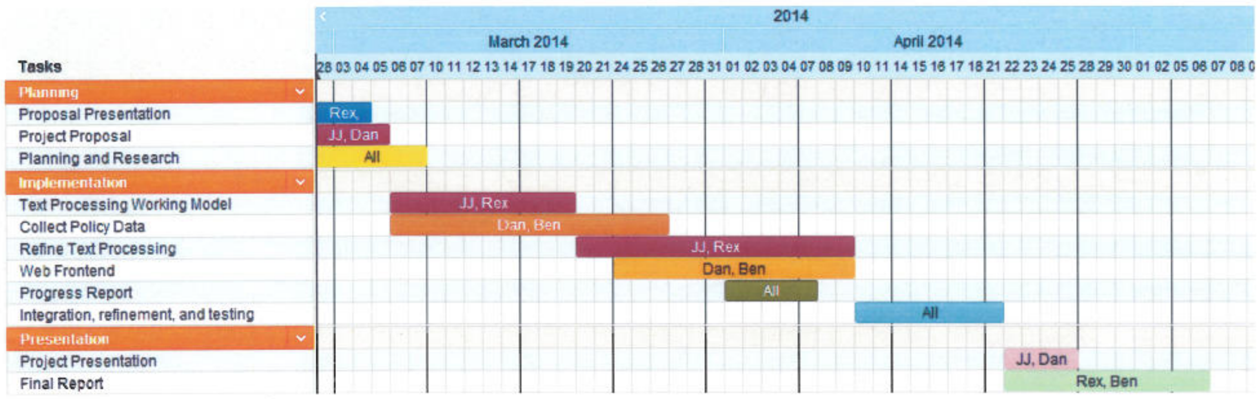

Figure 1: Old Gantt Chart

Recommendations

As noted in the progress section, the natural language processing component has trouble dealing with questions that are vague, require worldly knowledge, or logic. Thus, we decided to devote some more time into improving the accuracy of the NLP component. Since the web interface will just have an input form and textual output, we realized that it will not take much time. Thus, we decided it was beneficial to allocate more people more time to work on the backend and less on the front end.

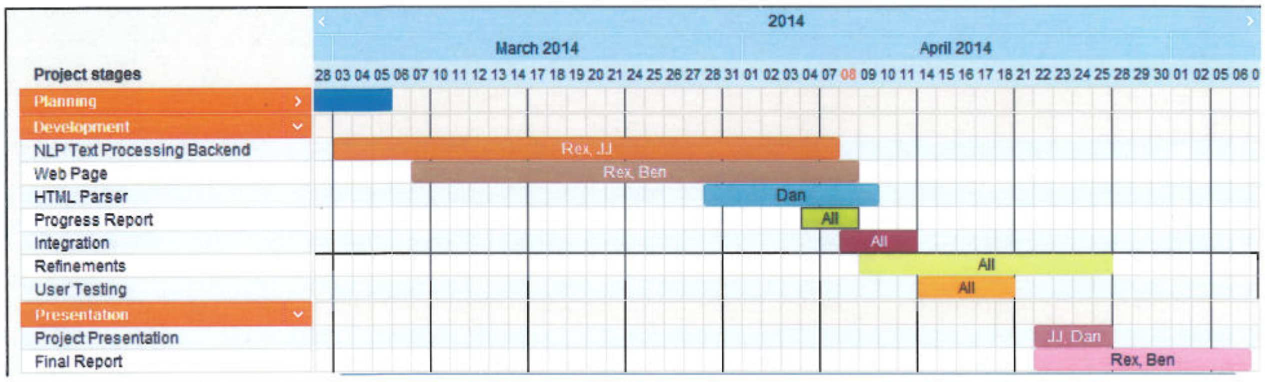

Figure 2: Revised Gantt Chart

Discussions

The changes in allocation will have no impact on the finished product. However, from initial evaluation of our NLP system, we are concerned that the question answering system may not live up to expectations of what we originally had in our proposal. This will only affect the effectiveness of our product. As long as the user asks a question for which the answer can be found within the article, then TL;DR should give a reasonable answer.