Jake Muir
Karl Bell
Toshihiro Kuboi
CSC 580

Lab 3

Parsing a Query

Our first approach to parsing a query from the text received from a user was regular expressions. We wrote a number of them based off of the examples provided in the Lab 3 instructions. We tested these out for a while and found that it caught most cases when we accounted for tense and some variations of words in our regular expressions. After all, there are only so many ways to ask the two questions we're supposed to answer.

However, we also wanted to be able to handle forms we hadn't thought of, in the case all of our regular expressions failed. To do this, we first took the message and determined whether it was a birthday query by looking for particular words. If it was, we attempted to pull out the last thing mentioned before these birthday words (if they were at the end of the sentence), or the thing after the birthday words (if they weren't at the end of the sentence). If the query contained no birthday words, we picked the subject by pulling out all of the words at the end of a sentence. In both of these cases, we picked only words which were capitalized -- That is, we assumed proper noun subjects.

Fetching a Description

Our approach to getting the first two sentences on a given subject from Wikipedia consisted of these steps: retrieve content from wikipedia page, clean up raw content, find the first 2 sentences, and return the result. Getting the content from a subject's Wikipedia page was pretty straightforward because the URL for a subject's Wikipedia page just contains the subject name with any spaces replaced by underscores. Once we had the raw content for the subject we had to clean up it so it could be analyzed for sentence boundaries correctly. Although the content of the Wikipedia pages showed some common formatting and consistency, this was probably the hardest part for fetching the description about the subject. In addition to simply cleaning up the HTML and finding the first paragraph, the raw content needed to be scanned for features like references and citations because they appeared in a special format. Here is an example of some raw content from the John Adams Wikipedia page: "An American [[Founding Fathers of the United States|Founding Father]],<ref>{{cite web|title=John Adams (1735–1826)|url=http://www.bbc.co.uk/history/historic_figures/adams_john.shtml}}</ref> he was a statesman, diplomat, and a leader of American independence from [[Kingdom of Great Britain|Great Britain]]." Using a sequence of regular expression statements we cleaned up the raw content to remove citations and references, and replace them with the main word/phrase that appears on your browser when viewing the page. As an example, we used regular expressions to replace instances like "[[Kingdom of Great Britain|Great Britain]]" with "Great Britain." After completing the sequence of regular expression statements, the above raw content sentence is changed to something much more readable: "An American Founding Father, he was a statesman, diplomat,

and a leader of American independence from Great Britain."

After cleaning up the raw content, we spliced it to only include the first two sentences. The sentence boundaries were determined by finding punctuation marks, and then looking at the immediately preceding character and the following two characters (possibly more if more than just one space after punctuation mark). The hardest part about finding the sentence boundaries was trying to distinguish acronyms from sentences that were actually ending. Basically we were looking for punctuation marks where the preceding character is not uppercase and the following characters consists of spaces and an uppercase character. After determining the sentence boundaries we returned the first sentences for the given subject.

Finding a Birthday

To get a birthday, we look at the source for the Wikipedia page (as we do for a subject), and look for the phrase "birth date". This is found in any Wikipedia article which has a sidebar. When we find this key phrase, we know that what follows is the birthday of the subject. We then attempt to parse out this date with a regular expression. Once we're able to get it's components, we can simply format the date how we wish and then respond with it.