HW #4 Report

Patrick Sheehan

Web API Mashup Reflection - SharpShopper

Looking back over the duration of this project, I reached many milestones while encountering my fair share of stumbling blocks. I had a faint idea for the idea of this project early in the semester but it lacked a definite direction and I lacked the motivation to move forward with a tutorial-level shopping list app. However, when I heard that our big assignment for the semester involved combining APIs in a unique way, I was enlightened.

This assignment unexpectedly turned out to be rather inspirational for the new path that SharpShopper would take. Other similar projects in the past that I've worked on have been a bit over ambitious in that they required a large user base in order to be relevant. Contrarily, with SharpShopper, the use of public APIs simply for information allowed it to be a self-functioning stand-alone application. As I learned, functionality is not everything, especially in a highly competitive market such as Apple's App Store. While my software works as I anticipated it, a much more user-friendly experience will be needed before I publish it.

I was quite happy with the overall result of this project. The expectations that I set for myself were to create a fully functional, integrated system as an iPhone app, no matter how bare-bones it turned out to be. I was proud of the amount that I accomplished, and I am eager to continue development on the app and pursue its initial release.

A great idea that I saw from other projects that I would like to incorporate into my own in the future is the use of public APIs not only for raw data pertaining to a topic, but also image resources such as Flickr and Google Images. These could definitely be a sufficient substitute to the Supermarket API I used because these more common ones are free and are more developed.

Bloom Filter Results

The error rate was incredibly high for such a low number of bits; 200 bits, 10 hashes, false positive rate = 98.41%. The effect of using a number this small was that when adding training data to the bit array, there were several duplicate hash results, and the array quickly approached its capacity as more and more training words were added.

Using many more bits made a significant difference; 1000 bits, 5 hashes, false positive rate = 83.35%. Although the error rate was still very high, the number of words that were filtered out decreased and the paragraph was still at least readable. The results can be found in the test files entitled: “bloom-output-<#>bits-<#>hashes”