|  |
| --- |
|  |
| Project 3: Information Retrieval |
| Leader Report |

|  |
| --- |
| Zachary Edens  4/9/2009 |

Table of Contents

[Version Control 3](#_Toc227083050)

[Members of Team 3 3](#_Toc227083051)

[Team Organization 3](#_Toc227083052)

[Development 3](#_Toc227083053)

[Tentative Schedule 4](#_Toc227083054)

[Application Overview 4](#_Toc227083055)

[Application Description 4](#_Toc227083056)

# Version Control

For version control, I decided on an SVN repository on code.google.com. Google Code allows a person to upkeep a repository, set an end-user license agreement, and set project members who share rights to modification of the code. This was also a desirable option because all members had claimed a Gmail account, could readily access the repository, and were familiar with Google Code’s use from prior experience.

# Members of Team 3

Zachary Edens - Leader

Justin Kern - Developer

Andrew Johnson - Developer

## Team Organization

For this project, I decided that I will work on the primary user interface, the stop list, and the nuts and bolts of the program. I kept the main user interface and the framework code for myself primarily so the other team members would not have to spend considerable amounts of time learning how the objects interact. I would like them to specialize in their areas while I remain the project architect. I will also find a stop list to integrate into the application for part in assisting with text processing and the inverted index algorithm.

I assigned Andrew two major components: the ranking algorithm and the stemming algorithm. Although I had done some initial research for stemming, Andrew quickly delved into his tasks. I gave these tasks to him also because of his technical knowledge. Hi s work will pertain mostly to the StemFilter class and the

I gave Justin the tasks of developing an information visualization interface to accompany the main UI and creating a tokenizer. I gave Justin the data visualization task because of his artistic skills. I only gave him the tokenizer on top of visualization because of the emphasis of visualization and user interface in this project. He will be using the latest version of the Prefuse visualization library.

These assignments are not in stone. As the project progresses, I will reassign work as necessary depending rate of completion.

# Development

For this project, I decided to use the NetBeans IDE to take advantage of GUI building capabilities and the Prefuse library for its powerful visualization capabilities. We will be using a variation of the Waterfall development cycle because the project requirements are set with a logical progression in developing parts of the program.

# Tentative Schedule

April 9 – 23: Primary project development. Group members will work on the their tasks, and unit testing will also be done.

April 22 – 28: Wrap up primary development and unit testing for whole-project tests.

April 29 – 30: Study for finals

# Application Overview

For this application I wanted to provide a user experience that is familiar and easy to use. The program should present a basic UI that is similar to applications users have used, but the concept of a search engine dictates the behavior of a web page like Google or Yahoo Search.

The application should provide typical functionality such as setting the working directory and changing views from a simple menu interface. The search function should be presented as the main page, divided into two sections: a section for entering and executing a query and a section for listing results for users to browse. The user should also be able to switch at will between visualization and traditional search layouts.

## Application Description

For realizing the application, I used the model-view-presenter architecture. This allows the UI, data processing, and data layers to largely remain separate in their implementations.

At the UI layer is the ui package containing

* MainFrame - JFrame for the program
* QueryPanel - for entering and starting queries
* ListPanel - displays search results in a list format
* VisualPanel - displays search results in a custom format
* ResultsPanel - visual component of a search result

The MainFrame set to the QueryPanel and ListPanel will be what the user sees first; this combination of UI elements represents the typical search engine interface. A menu will be devoted to switching views which will tell the UIManager to swap the ListPanel with the Visual panel. The user should still be able to make queries in visual mode.

In the controller package is:

* UIController - The UIController handles the logic behind user interface events like switching from a list view to visual view and providing a way to choose a working directory
* DataController - The DataController provides most of the glue code for the program. Its only UI event is the initialization of a search. In response, it will feed the QueryProcessor class the original query to generate a list of index terms. It will feed those index terms to the DocumentSearcher class which returns a list of ranked Documents.

The textProcessing package contains all objects necessary for evaluating a query from the user:

* QueryProcessor - Class that owns the Tokenizer, StemFilter, and StopWordFilter classes. The processor transitions data amongst each section of processing to return a list of index stems.
* Tokenizer - Breaks the query into individual words and removes erroneous punctuation.
* StopWordFilter - Takes input from the Tokenizer and removes any token considered too common to yield good search results
* StemFilter - The final pass of the QueryProcessor, which applies a stemming algorithm to the remaining search terms.

After text processing is completed the list of index terms from the QueryProcessor is handed off to the DocumentSearch class in the documentSearching package. The DocumentSearch owns list of indexes to documents and encapsulates the search and page ranking algorithm. During the search, the DocumentSearcher will populate a container with documentSearching.Document objects. These objects implement the DocumentInfo interface of the project3 package, and will be sent off to the UI for constructing ResultPanels.

The above process outlines a typical search, but application start-up includes a few other objects of note. The infoRetrieval package has two classes:

* InfoRetrieval - This class represents the application independent of the main method.
* Main - This is the entry point of the project where the InfoRetrieval.Run() method is called to begin execution.

After the call to Run() is made, the IndexBuilder is launced and will populate the IndexMap structure with inverted indexes to all text documents in a directory, which is indicated by the user before the IndexBuilder is launched but after the call to Run().