**Adaptive Web: (Content-based) Recommendation**

**Submitted By: Shaishavkumar Jogani (1212392985, sjogani@asu.edu)**

**Content Collection:**

In order to provide user recommendations on a post based on the content of a post, I used a popular technique called content-based recommendation system. To gather a content on Java Programming Language, I used the Wikibooks Java pages, and Oracle Java Tutorials. I first, gather all the links from the home page of both pages and crawl through each of them to the received content. I used the python script for crawling and stored the crawled content in a text file. While crawling I ensure that the content does not repeat, and unnecessary links in the header, footers, menu bar are excluded.

**Content Indexing:**

Indexing is a technique that collects, parse and stores the data to facilitate fast and accurate search result. The overall goal of it is to retrieve the relevant information of search query in timely manners. To suggest user contents on a post, I indexed the crawled content using Apache Lucene Library. Apache Lucene is a free and open-source information retrieval software library, that provides the full-text indexing and searching functionality.

In addition to that, I remove all the stop words and stemmed the result while crawling and provide that result for indexing. Stop words are irrelevant for searching purposes because they occur frequently in the language. Stemming converts each word into its root form. It eliminates the difference between inflected forms of a word.

**Web Application:**

For every 10 posts, I took all contents and gave it as a search query to my Lucene Indexing server. It provides me with the most relevant 10 recommendations based on the post. I used the result to provide a user a content-based recommendation. I treated the code and text in each web-page separately while indexing and providing the result. So, that user can get code examples relevant to the post and text description explaining the key topics in a post. Since I did the indexing on stemmed and stop words removal text, indexing result gives the same output. I link that output with the actual content and display that to the web-page.

In Web UI, the screen is divided into two vertical panes. List of posts are shown in the left pane, and on the right pane post description and its 10 recommendations are shown. Upon clicking on a post from the left pane, right pane’s content changes accordingly.

**Originality:**

* Index the crawled content into the section:
  + On the Java Wikibooks page, the web-page on each subject contains very details description of the topic. However, the relevant information can be found in a subsection of the page and rest are unnecessary. Keeping that in mind, while gathering the data, I took all the subsection of the page and only store them. So, each sub-section is indexed separately. This resulted in the improvement of a system’s information retrieval, as we can ignore the un-relevant information and keep the correct one.
* Separate Text and Code:
  + Programming help pages are written in a way that contains both text and code. Text which explains the fundamentals of language, and code which illustrates them. I treated them considering that. I also stored the text and code content of any sub-section of the page in different files. I processed stop words removal and stemming only on text files, as in code each word has meaning.
* Navigation Style UI for ease:
  + Because, each post and its recommendations contain a very large amount of data, displaying them on a single page can be lost the user while s/he reads the information. To resolve this, I provided a navigation bar with a link to 10 posts on the left pane. By this, a user can easily navigate between posts and only finds the relevant information regarding the post.