Universidad de Costa Rica

Escuela de Computación e Informática

**Tarea Programada I**

Elaborado por:

Daniel Muñoz B24645

Jeffry Venegas B37495

16 de mayo de 2016

**Resumen**

La tarea consiste en la implementación de tres módulos, una araña que navega por la WWW, un procesamiento lingüístico y el uso de los algoritmos de indexación como BSBI o SPIMI.

Se buscó un diseño usando Java como lenguaje de programación, se encontró una biblioteca llamado JSoup que logra conectarse con las páginas web para extraer información. JSoup permite “parsear” de forma simultánea la página que se acceda por lo que a partir de la conexión se crean archivos txt con base en el archivo denominado “Urls.txt” en la carpeta “Araña” que contiene las direcciones a buscar. (Estas páginas web que se acceden no deben contener una seguridad muy avanzada ya que reconocen que la araña está solicitando permiso a través del servicio y lo niega).

El procesamiento lingüístico consiste en separar los archivos de texto por término para que sea más sencillo su lectura, por lo que se prosiguió a crear los documentos de forma que cada término estuviese en un renglón diferente. Esto permite a la hora de indexar una lectura más eficiente.

La indexación consistió en el análisis de cada término encontrado por documento, si no existía se agregaba a la estructura de datos, caso contrario se buscada el dato que contuviese dicho término, se analizaba si era de un mismo documento o diferente y se aplicaba la misma metodología para que finalmente se contabilizara la frecuencia con que aparecía cada término por documento leído.

Se tomó el tiempo que dura el programa en indexar los documentos, y se guardaron los resultados en un archivo de texto. Se corrió el programa 200 veces para cada configuración.

Estos resultados junto con el tamaño del índice para cada uno de los experimentos, se ubican en la carpeta Resultados dentro de Promedio, esta última contiene un programa que lee esos resultados de los experimentos y calcula un promedio de esas mediciones de tiempo.

**Experimentos**

Para los experimentos de cada módulo se analizó sus propiedades, el experimento de la “Araña” mide los tiempos de conexión y extracción de datos usando diferentes agentes o navegadores. El procesamiento brinda opciones de quitar los signos de puntuación y usar *stemming* para medir tiempos de duración en uno y otro caso con el fin de ver su velocidad de procesamiento lingüístico.

Finalmente, para el módulo tres se construyeron los índices con diferentes estructuras de datos para medir su tiempo de duración y ver las diferencias, junto con otro experimento de usar la opción de quitar *stop words* para su indexación.

Los resultados de los experimentos se encuentran en la carpeta Promedio.

En promedios.txt se encuentra la información de la duración promedio para cada configuración de construcción del índice, así como el tamaño del índice.

En la carpeta Resultados se encuentran las mediciones de tiempo de cada configuración:

* Configuración de la araña con el agente Firefox 25 (Firefox.txt).
* Configuración de la araña con el agente Chrome 41 (Chrome.txt).
* Indexado sin alterar los términos usando HashMap (HashMap.txt).
* Indexado sin alterar los términos usando TreeMap (TreeMap.txt).

Usando TreeMap:

* Aplicación de *stemming* (Stemming.txt).
* Eliminación de signos de puntuación y otros símbolos (Simb.txt).
* Eliminación de *stop words* (StopWords.txt).
* Eliminación de signos de puntuación y otros símbolos y eliminación de *stop words* (StopWordsSimb.txt).
* Aplicación de *stemming* y elminación de *stop words* (StemmingSW.txt).

De los experimentos se puede decir:

1. Utilizar un HashMap como estructura de datos es más rápido que utilizar un TreeMap, esto debido a que el TreeMap mantiene los datos ordenados al insertar. Para la implementación del HashMap se ordena hasta el final, solo los términos, no sus respectivas listas de postings.
2. Es más rápido el procesamiento de la araña si se utiliza el agente Chrome 41, en lugar de Firefox 25.
3. Eliminar signos de puntuación no significa mucha diferencia en el tiempo de construcción del índice.
4. El eliminar *stop words* aumenta considerablemente el tiempo de ejecución.
5. Aplicar *stemming* duplica el tiempo, pero reduce el tamaño del índice cerca de un 90%.
6. Si se aplica *stemming* y se eliminan *stop words* se cuadruplica el tiempo de ejecución.

**Sobre la escalabilidad del sistema.**

Las estructuras de datos utilizadas son dinámicas, por lo que permiten que la información almacenada crezca y cambie con el tiempo, por lo que no debería ser un problema procesar cantidades grandes de documentos, esto dependería más que todo del *hardware* donde se ejecute el programa.

**Manual de Usuario**

Para el uso del programa en la carpeta “Araña” debe agregar la biblioteca jsoup.jar a las preferencias del IDE (se usó BlueJ por simpleza) y ver que el archivo llamado “Urls.txt” tenga direcciones web correctas y con seguridad no muy alta, por ejemplo: amazon.com no permite la extracción.

Los archivos generados se deben mover al directorio “documents” ya sea en el directorio “IndexarHashMap” o “IndexarTreeMap”

Los programas dentro de “IndexarHashMap” y “IndexarTreeMap” usan el mismo diseño, pero su objetivo es plantear una diferenciación o comparación de tiempo entre las dos estructuras. Puede seleccionar la opción de indexar y luego escoger algunas de las casillas para ver su comportamiento en tiempo. Estos tiempos se guardarán en un archivo de texto, “TreeMap.txt” o “HashMap.txt” dependiendo de la estructura utilizada. Además, el procesamiento de los archivos extraídos de los URL, *parsea* el contenido y lo almacena de manera que sea fácil de procesar por parte del sistema.