INSTITUTO TECNOLÓGICO DE COSTA RICA **II SEMESTRE 2016**
INGENIERÍA EN COMPUTACIÓN
IC-8020 RECUPERACIÓN DE INFORMACIÓN TEXTUAL

TAREAS PROGRAMADAS 1

Esta tarea programada consta de tres partes:

- Implementar la creación de archivos invertidos.
- Implementar las búsquedas vectoriales usando el archivo invertido.
- Implementar búsquedas estructuradas para refinar el resultado de las búsquedas vectoriales.

Se usará la colección ***Flora-2016-01-22*** que se distribuye con este documento. Dicha colección consta de 711 archivos XML con un total de 4,8MB de texto. Los archivos describen en forma muy estructurada las características de diferentes árboles de Costa Rica.

Indización

A partir de la colección distribuida, se debe crear un archivo invertido que permita el procesamiento eficiente de consultas vectoriales. El procesamiento de los índices y de las consultas debe ser razonablemente eficiente y debe poder manejar la totalidad de la colección que se proveerá.

Cada archivo de la colección contiene un único documento que se quiere indexar. Solo se debe indexar la siguiente información:

- el texto del atributo **`taxon_description`** del elemento **`<description>`**
- el texto de los atributos **`taxon_name`** y **`rank`** del elemento **`<taxon_identification>`**.

Los términos que serán usados en las búsquedas deben ser obtenidos aplicando las siguientes consideraciones:

- **palabras**: secuencias consecutivas de una o más letras; deberán simplificarse eliminando tildes y convirtiéndolas a minúsculas; la eñe debe preservarse
- **números**: secuencias consecutivas de uno o más dígitos; opcionalmente pueden tener una única coma decimal en medio; si la coma aparece, debe haber por lo menos un dígito antes y uno después de la misma
- **stopwords**: usar la lista especificada en https://github.com/6/stopwords-json/blob/master/dist/es.json
- **frases**: si el texto del atributo **`taxon_name`** contiene más de una palabra, se debe incluir como término la frase formada por las primeras dos palabras; por ejemplo, el caso:
  `<taxon_identification taxon_name="Quercus cortesii Liebm." ...>`
  generaría el término = “quercus cortesii”; asegurarse de dejar exactamente un espacio en blanco entre ambas palabras de la frase

El archivo invertido consistirá de tres partes:

documentos
diccionario
postings

La localización usada será (doc-id).

En otras palabras, la información que se lee y escribe de disco debe ser una variante de la siguiente estructura:

```
typedef struct {
   int  docid;
   char doc[MAXRUTA];
} documento_t;
```

```
typedef struct {
   char term[MAXTERM];
   int inicio;
   int numdocs;
} diccionario_t;
```

```
typedef struct {
   int   docid;
   int   frecuencia;
   float peso}
posting_t;
```

El archivo de postings puede ser cargado en su totalidad en memoria, **pero sus registros deben ser de tamaño fijo**. Esto es para permitir la posibilidad de acceder directamente en disco a la lista invertida de un término a partir de su información en el diccionario.

Para la creación del archivo invertido puede suponer que toda la información cabe en memoria. **LA INDEXACIÓN DEBE GENERAR ARCHIVOS QUE PUEDAN SER USADOS PARA CONSULTAS SIN TENER QUE REINDEXAR LA COLECCIÓN.** No debe darse el caso de que la información del índice solo está en memoria y que se pierda cuando termina el programa.

Los pesos serán calculados usando la fórmula:

$$w_{ij} = \left(1 + log_2(freq_{ij})\right) \cdot log_2(N/n_i)$$

Búsqueda vectorial

Las consultas consistirán en búsquedas vectoriales que aprovecharán el archivo invertido para obtener eficientemente la respuesta.

Las búsquedas tendrán el formato:

operando$_1$ ... operando$_n$

donde los operandos pueden ser de los siguientes tipos:

- ***termino***
  termino es alguno de los términos simples indexados (palabras o números)
- ***prefijo****
  prefijo es el inicio de términos; por ejemplo gat* ==> gata, gato, gatas, gatos.
- "***termino1 termino2***
  se usan ambos términos juntos como una frase

Antes de cada operando hay un caracter +/- opcional, el cual le permite al usuario especificar una mayor o menor importancia de los términos en la consulta. Si se omite, la importancia por omisión es 2. Si se incluye un "+" la importancia es 4 y si se incluye un "-" la importancia es 1.

Dichos valores de importancia se usan como frecuencia al calcular los pesos de los términos de las consultas.

Ejemplos de consultas:

- **árbol pequeño**
- **flores frutos amarill***
- **hojas simples "Lecointea amazonica margen aserrado**
- **+árbol mediano +grande**
- **flores frutos -amarill***
- **+hojas simples +"Lecointea amazonica +margen -aserrado**.

Los documentos que cumplan con la consulta deberán ranquearse en orden inverso a su similitud.

Se usará la similitud coseno en las búsquedas vectoriales:

$$sim(d_i, q) = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

Como resultado de una búsqueda se deben producir dos archivos. El primero es el escalafón, el cual contiene un encabezado con la siguiente información:

- fecha y hora en que fue corrida la consulta
- ruta de la colección de documentos
- texto de la consulta

y luego lista en orden decreciente de similitud la siguiente información sobre todos los documentos que cumplen con la consulta:

- posición en el escalafón
- valor de la similitud obtenida
- doc-id
- **taxon_name** y **rank** del documento
- ruta del documento

El segundo archivo es un documento HTML para mostrar al usuario. Este archivo incluye la siguiente información:

- fecha y hora en que fue corrida la consulta.
- ruta de la colección de documentos.
- texto de la consulta.
- para los primeros 30 documentos recuperados incluir su posición en el escalafón, similitud obtenida, docid, **taxon_name**, **rank** y ruta
- incluir los 200 primeros caracteres del atributo **taxon_description**. Se debe consolidar el espacio en blanco en un solo espacio en blanco y cambiar los cambios de línea por dos espacios en blanco.

Consultas estructuradas

Los resultados de las consultas vectoriales pueden ser refinados por medio de condiciones que toman en cuenta la estructura de los documentos recuperados. Esto requiere procesamiento secuencial de dichos documentos para verificar si ciertos elementos XML contienen la información que se les pide explícitamente.

Las consultas estructuradas toman la forma de una o más cláusulas con el siguiente formato:

<name of biological_entity> <name of character> <value>

Por ejemplo, la siguiente consulta:

ramitas pubescence glabras

sería satisfecha por un documento que incluya lo siguiente:

```
<biological_entity id="T2L3S1-180650" name="ramitas" type="structure">
    <character name="pubescence" value="glabras"> </character>
</biological_entity>
```

Si una consulta tiene más de una cláusula, se supone que hay una operación AND entre ellas y los documentos seleccionados deben cumplir con todas ellas.

Para realizar una consulta estructurada se debe especificar el archivo escalafón de una consulta vectorial previa y el conjunto de cláusulas que se quieren satisfacer.

Como en el caso vectorial, una búsqueda estructurada debe producir dos archivos. El primero es el escalafón refinado, el cual contiene la misma información que el escalafón de la consulta estructural, con las siguientes diferencias:

- el encabezado contiene ahora la siguiente información:
  - fecha y hora en que fue corrida la consulta estructurada
  - ruta del escalafón de la consulta vectorial
  - lista de cláusulas de la consulta estructurada
- la lista de documentos contiene la misma información que en el escalafón de la consulta vectorial, pero solo para aquellos documentos que cumplen con la consulta estructurada; se mantiene el mismo orden decreciente por similitud

El segundo es un documento HTML para mostrar al usuario que incluye la siguiente información:

- fecha y hora en que fue corrida la consulta vectorial
- fecha y hora en que fue corrida la consulta estructurada
- ruta de la colección de documentos.
- texto de la consulta vectorial
- lista de cláusulas de la consulta estructurada
- para los primeros 30 documentos recuperados incluir su posición en el escalafón de la consulta estructurada, similitud obtenida, docid, **`taxon_name`**, **`rank`** y ruta
- incluir los 200 primeros caracteres del atributo **`taxon_description`** archivo. Se debe consolidar el espacio en blanco en un solo espacio en blanco y cambiar los cambios de línea por dos espacios en blanco.

Interfaz

A modo de guía se listan a continuación tres comandos para realizar las operaciones anteriores. No es obligatorio seguir el formato mostrado, pero la funcionalidad indicada sí debe ser implementada de alguna manera.

`indexar` *RutaColección RutaArchivoInvertido Prefijo*

Prefijo es usado para iniciar los nombres de los archivos generados; por ejemplo con prefijo "PR1-" se crearían archivos como:

PR1-documentos
PR1-diccionario
PR1-postings

`consultar` *RutaArchivoInvertido Prefijo Consulta*

Prefijo es usado para iniciar los nombres de dos archivos generados por una consulta; por ejemplo con prefijo "Q1-" se crearían archivos como:

Q1-rank
Q1-resultado.HTML

`buscar` *RutaEscalafon Prefijo Cláusulas*

Prefijo es usado para iniciar los nombres de dos archivos generados por una búsqueda estructurada; por ejemplo con prefijo "Q1e-" se crearían archivos como:

Q1e-rank
Q1e-resultado.HTML

Cláusulas son las cláusulas que forman la búsqueda estructurada, separadas por comas; por ejemplo:

`ramitas pubescence glabras , hojas architecture simples`

Consideraciones finales

La tarea puede ser programada en cualquier lenguaje de programación que esté disponible en los laboratorios de la Escuela. La fecha de entrega del proyecto es el lunes 26 de setiembre al mediodía. El proyecto puede ser realizado por grupos de dos personas.