Actividad Dirigida
Implementación Básica del Modelo Vectorial

Prerrequisitos de la práctica

- Programación (la práctica se podrá realizar en el lenguaje que el alumno desee)
- Base de datos relacionales (opcional, aunque recomendable) y saber conectar el programa con la base de datos

Enunciado

Para realizar esta actividad, se proporciona la misma consulta del ejercicio en Excel sobre el espacio vectorial. En la implementación deberán tenerse todas las decisiones de diseño indicadas en el Excel.

- Se considera la misma palabra en mayúsculas y minúsculas, se puede normalizar siempre a minúsculas y sumar las ocurrencias

- Se puede agrupar singulares y plurales, con acento y sin acento

- Consideramos palabras sencillas siempre o puede haber términos compuestos

- Para este ejercicio se pueden eliminar las palabras vacías (en concreto tener en cuenta coche rojo blanca marca Citroen blanco Madrid ocasión auto)

- Se ha considerado palabra sencilla y no tenido en cuenta la categoría gramatical

- Se ha tenido en cuenta el mismo cálculo de pesos en documentos y consulta (no es habitual) , en concreto $W_{ij}=tf*IDF=f_{i,j}*\log (N/n_i)$

Estas decisiones podrían no ser pertinentes en un caso práctico

La solución correcta se podrá validar tan solo comparando con los datos de la hoja Excel:

Consulta	**Coche rojo de Blanca**
Documento 1	Marca Citroën color rojo
Documento 2	Coches rojo coches blancos Madrid-ocasión
Documento 3	Ocasión Madrid autos rojos de marcas blancas

CC BY NC SA

J.Morato, V.Palacios, S.Sanchez-Cuadrado, M.Marrero, J.Urbano

Cada uno de estos documentos deberá estar en un HTML independiente, que deberá parsear para extraer el texto.

Se pide implementar un programa Java que tiene como objetivo principal calcular el Modelo Vectorial de recuperación para la similitud entre los documentos y las consultas. Para ello este programa debería primero crear un diccionario en RAM con los valores de TF e IDF para cada término y documento. Una vez creado el diccionario el programa procede a calcular los valores de similitud según las siguientes funcionalidades:

- Producto Escalar TF: Función de similitud del producto escalar con pesos según TF.
- Producto Escalar TF IDF: Función de similitud del producto escalar con TFxIDF.
- Coseno TF: Función de similitud del coseno con pesos TF.
- Coseno TF IDF: Función de similitud del coseno con pesos TFxIDF.

Uso de Librerías y Recursos Externos

- Se permite y aconseja el uso de librerías u otros recursos para realizar procesos de tokenización, filtrado por palabras vacías (si se considera oportuno), normalización y stemming o lematización.
- No deben utilizarse librerías externas para la creación directa del índice ni para el cálculo de la similitud entre documentos y consultas.
- Los recursos externos utilizados deberán estar debidamente documentados. No debe utilizarse el código parcial o total de otros recursos.

Recomendación sobre la implementación:

- Crear un proyecto Java llamado ModeloVectorial_N, donde N es el número del grupo. La clase principal llamada Modelos contiene el método main(). Está permitido implementar y utilizar cualquier clase, método auxiliar, y estructuras propias de Java, con el fin que el programa funcione correctamente. Por ejemplo, para crear el diccionario, se puede utilizar un método llamado CrearDiccionario().
- Se tendrá en cuenta la eficiencia de las implementaciones en tiempo de consulta.
- Los resultados de la similitud se deberán mostrar por la pantalla en función de la relevancia entre el documento y la consulta según cada función, y como se muestra a continuación:

```
* RELEVANCIA: ProductoEscalarTF
Nombre del doc     Q1
Documento1 valores……
……                 ……

* RELEVANCIA: ProductoEscalarTFIDF
Nombre del doc     Q1
Documento2         valores……
……                 ……

* RELEVANCIA: CosenoTF
Nombre del doc     Q1
Documento3         valores……
……                 ……
```

Posibles Mejoras

- **Índice en base de datos.** En el enunciado de la práctica el índice inverso se ha almacenado mediante diccionarios en memoria RAM por sencillez. Sin embargo, los índices realmente se guardan en bases de datos de diversos tipos. Para la práctica del motor de búsqueda se implementará el índice inverso con una base de datos MySQL por ejemplo.
- **Tokenizer.** En este caso el tokenizer puede ser muy básico. Sin embargo, podría mejorarse tanto para limpiar las etiquetas HTML como para separar los términos. Para la práctica del motor de búsqueda deberán incorporarse estas mejoras.
- **Normalización de Términos.** Los **términos** se podrían normalizar, por ejemplo pasándolos a minúsculas y reduciéndolos a la raíz, para que casos como *Teclado* y *teclados* se consideren el mismo término. Para la práctica del motor de búsqueda se deberán implementar este tipo de mejoras.
- **Separar el cálculo de pesos de la función de similitud.** Tanto la función de similitud del producto escalar como la del **coseno** están definidas independientemente de los pesos que se den para cada término. En vez de implementar cada modelo con una asignación de pesos concreta (e.g. ProductoEscalarTF, ProductoEscalarTFIDF, CosenoTF y CosenoTFIDF), se podría separar en una clase para la función de similitud (e.g. ProductoEscalar y Coseno) y otra clase para el cálculo de pesos (e.g. TF y TFIDF).

Corrección de la práctica

Comparar los resultados con los del ejercicio Excel, tiene los mismos datos que en esta ocasión.

CC BY NC SA