Proyecto Final de Sistemas de Recuperación de Información

Leismael Sosa Hernández C-312
Alejandro Yero Valdes C-311

Resumen

En este documento se describe todo el proceso de diseño, implementación y evaluación de un Sistema de Recuperación de Información que permite elegir entre el Modelo Vectorial Clásico, el Modelo Probabilístico y el Modelo de Semántica Latente para realizar consultas. Se analiza primeramente el problema de la recuperación de información, se explica brevemente cada uno de los modelos implementados y se finaliza con la evaluación sobre una colección de pruebas, la colección de documentos Cranfield.

1. Introducción

Este trabajo describe el proceso realizado por parte de nuestro equipo para la implementación de un Sistema de Recuperación de Información que permite de una forma sencilla tanto la implementación de nuevos modelos de recuperación de información, como la creación de nuevos parsers que transformen los documentos de colecciones arbitrarias a una representación estándar que reconoce el sistema y también permite la evaluación de estos modelos con colecciones estándares como la popular colección de Cranfield.
Nuestro equipo seleccionó el Modelo Vectorial Clásico, el Modelo Probabilístico y el de Semántica Latente como los modelos a implementar en nuestro sistema.

1.1. Colecciones

Las colecciones de documentos de prueba que se usan en el sistema para realizar consultas sobre ellas son la popular colección de Cranfield y la de Reuters.

1.2. Herramientas utilizadas

Para la implementación de nuestro sistema se utilizó el lenguaje de Python debido a su facilidad de uso y a la gran cantidad de librerías de las que dispone, como es el caso de la librería de procesamiento natural del lenguaje NLTK.

1.3. La Recuperación de Información

La recuperación de información es un campo de la ciencia que nace con el surgimiento de las bibliotecas y tiene un auge con la aparición de las computadoras y el internet, esto es debido a que en ambos casos se requiere de un sistema que permita organizar la información de una forma que permita su rápido acceso y satisfaga una necesidad de información.
Explicado más formalmente, el objetivo de un Sistema de Recuperación de Información es, dada una necesidad de información y un conjunto de documentos, devolver un orden de los mismos de acuerdo al nivel con el que satisfaga la necesidad de información.
El proceso completo que sigue un sistema de este tipo, dada una colección de documentos, se puede dividir en las siguientes etapas:

- **Preprocesamiento**: consiste en la transformación de los documentos de la colección de tal forma que el sistema pueda entenderlos. Este proceso se puede dividir en dos partes:
  - **Tokenización**: consiste en dividir los documentos en unidades de información que el sistema pueda entender, como por ejemplo palabras.
  - **Normalización**: consiste en transformar las unidades de información en un formato estándar, como por ejemplo pasar todas las palabras a minúsculas, aplicar stemming, lemmatization, etc.
- **Indexación**: consiste en el almacenamiento de los documentos de la colección en un formato que permita su rápida recuperación

- **Consulta**: consiste en la transformación de la consulta del usuario en un formato estándar que el sistema pueda entender.
- **Recuperación**: consiste en la recuperación de los documentos que satisfacen la necesidad de información del usuario.
- **Evaluación**: consiste en la evaluación de la calidad de los resultados obtenidos por el sistema.

A lo largo de este documento se explicará cada una de estas etapas con cada uno de los modelos que se implementaron. Pero antes de esto explicaremos la arquitectura general de nuestro proyecto.

2. Arquitectura General

Como se mencionó en la introducción, nuestra arquitectura está pensada para permitir una sencilla implementación de nuevos modelos y nuevos parsers de colecciones de documentos. Para esto implementamos una serie de clases abstractas que permite que el sistema tenga el siguiente flujo:

- El usuario selecciona un archivo(s) que represente la colección deseada para indexar junto con el Parser que se quiere que las procese. Ejemplo: Colección Cran con el Parser de CranParser.
- El Parser devuelve una colección de documentos en un formato estándar en nuestro sistema (clase Document). Estos documentos una vez creados procesan el texto usando la librería nltk para poder remover stopwords, realizar el proceso de Stemming, transformar las palabras a minúsculas, entre otras operaciones.
- Estos documentos son enviados al Modelo de Recuperación que el usuario desee para que puedan ser indexados por el mismo.
- Luego de indexados es que el usuario puede comenzar a escribir queries.
- Una vez el usuario escribe una query, el sistema se encarga de enviársela al Modelo seleccionado para que procese la query y devuelva la colección de documentos más relevantes, ordenados de mayor a menor. Este orden está dado por la función de Ranking asociada a cada Modelo

El conjunto de clases bases que permite esta flexibilidad es la siguiente:

2.1. Class Model

Model es la clase base para todos los Modelos de Recuperación de Información que se quieran implementar en el sistema. Todos deben implementar la siguiente interfaz

```
class Model:
def get_model_name(self) -> str:

def add_document(self, document: Document)

def get_ranking(self, query:str,
                first_n_results:int,
                lang:str = 'english') -> list[Document]
```

2.2. Class Document

La clase Document representa el formato estándar de un documento que todos los modelos del sistema deben saber como indexar. Esta clase expone propiedades básicas que debe tener todo documento:

- Nombre del documento
- Cuerpo del documento

2.3. Class Parser

La clase Parser presenta unos métodos básicos que deben presentar todos los Parsers que se quieran agregar al sistema. Dentro de estos el más importante es el método para parsear un archivo. Este archivo puede ser tanto un documento como una colección de estos.
Quien se encarga de enviarle los archivos correctos al Parser es la clase IRS, de la cual

2.4. Class IRS

La clase IRS es la clase que se encarga de manejar la interacción entre el usuario y el sistema. Esta clase contiene funcionalidad para registrar nuevos modelos al sistema, para guardar en el almacenamiento físico los modelos

que se crean (y para cargarlos de ahí también), para registrar nuevos parsers y usarlos, etc. Por último, esta clase permite realizar una query al sistema, seleccionando un modelo y devuelve un ranking de los documentos según el modelo seleccionado.

3. Modelos de Recuperación de Información

Los modelos de recuperación de información que implementamos en este sistema fueron:

- Modelo Vectorial Generalizado
- Modelo Probabilístico
- Modelo de Semántica Latente

En las siguientes secciones explicaré las decisiones de diseño tomadas para la implementación de cada uno de los modelos.

3.1. Modelo Vectorial Clásico

Nuestra implementación del Modelo Vectorial Clásico es casi idéntica a como se describe en la literatura, es decir, representando los documentos y consultas como un vector n-dimensional, donde n es la cardinalidad del vocabulario.
Vale la pena hacer notar que a la hora de representar los vectores de documentos y consultas no llega a ser necesario crearlos como un array de longitud n. Esto es por el hecho de que casi todo documento contiene un porcentaje muy pequeño de términos del vocabulario y por consecuencia, su vector n-dimensional tendrá muchos ceros que a la hora de calcular la similitud no aportarán valor alguno y si se remueven reduciría bastante el consumo de la RAM del sistema. La forma que seleccionamos para reducir la cantidad de elementos del vector en el programa fue crear un diccionario en vez de un arreglo, donde las llaves fueran los términos del sistema presentes en el documento y el valor fuera el peso del término en el documento.
Las ventajas de hacer esto fue que se redujo el tiempo de cálculo de similitud de las consultas con respecto a la version del arreglo y además se redujo grandemente el consumo de RAM.

3.2. Modelo Probabilístico

En la implementación del Modelo Probabilístico no hicimos nada novedoso. Y en cuanto a su retroalimentación usamos Pseudo-Retroalimentación.

3.3. Modelo de Semántica Latente

En nuestra implementación del Modelo de Semántica Latente solo hay que hacer notar que la forma en la que calculamos los pesos de la matriz término-documento fue usando Tf-Idf.

4. Evaluación de los Modelos

A la hora de evaluar cada uno de los modelos implementados con la colección de prueba Cranfield se obtienen los siguientes resultados.

	Recall	Precision	F1
Modelo Vectorial Clásico	0.103	0.230	0.142
Modelo Probabilístico	0.103	0.230	0.142
Modelo de Semántica Latente	0.172	0.333	0.227

5. Ventajas y Desventajas

Las ventajas que tiene el sistema implementado son las siguientes:

- Es sencillo implementar y agregar nuevos modelos al sistema
- Al proveer de una representación estándar de los documentos que todos los modelos reconocen, es fácil implementar un Parser, agregárselo al sistema y agregarle las colecciones de documentos que se deseen a cualquier modelo implementado.

Como desventajas podemos citar la siguiente:

- La forma de almacenar físicamente los documentos indexados no es la mejor, puesto que se almacena el modelo completamente con todos los documentos indexados y esto en colecciones muy grandes puede ser un problema.

6. Recomendaciones

Las recomendaciones para futuros proyectos sería que se implemente un sistema de almacenamiento mejor que el actual, que permita guardar por separados los documentos de su forma de indexarse. Esto lo haría mucho más eficiente y escalable puesto que se podrían adicionar mejoras en modelos específicos para que a la hora del ranking solo tengan que cargar la parte de la colección que necesiten en cada momento.