Implementación de sistemas de recuperación de información basados en el estadístico $tf - idf$

Victor Manuel Cardentey Fundora
Grupo C511

Karla Olivera Hernández
Grupo C511

Amanda González Borrell
Grupo C511

Tutor(es):
Prof. Marcel E. Sánchez Aguilar

1. Introducción

El estadístico $tf - idf$ permite modelar la importancia de un término dentro de un documento perteneciente a un corpus determinado. Esta medida crece proporcionalmente con el número de veces que aparece el término en el documento y es ajustada de acuerdo a la cantidad de documentos del corpus donde aparece el término, esto permite capturar el hecho de que algunas palabras aparecen con más frecuencia que otras de forma general. Su facilidad de cómputo, la efectividad en la modelación de la importancia de términos y que pueda ser adaptada tanto a modelos vectoriales como probabilísticos hace que sea una de las medidas más utilizadas en la implementación de sistemas de recuperación de información. En este trabajo se implementará una clase base que permitirá implementar distintos modelos que utilicen esta medida, para ello brindaremos como ejemplo la implementación del modelo vectorial y del modelo probabilístico frecuentista Okapi BM25.

2. Desarrollo

2.1 Arquitectura

El sistema desarrollado se compone de dos módulos: un módulo de procesamiento de texto en lenguaje natural en inglés y un módulo de modelos de recuperación. El desarrollo de la solución se realizó en el lenguaje **python**

2.2 Procesamiento de texto

El primer paso en el *pipeline* de recuperación es transformar la representación en string del texto en un conjunto de términos significativos. Para esto se emplearon las bibliotecas **nltk**, **re**, **contractions**, **unicode** las cuales proveen un conjunto de métodos útiles para el procesamiento de lenguaje natural.

Se definió una interfaz para la funcionalidad de un procesador de texto que en este caso normaliza un string y lo convierte en un array de strings donde cada uno representa un término.

```
class TextNormalizer:
  def normalize(self, text, verbose=False):
    raise NotImplementedError()
```

Figura 1: Interfaz de procesador de texto.

La implementación provista en esta solución provee las siguientes funcionalidades:

1. Eliminación de espacios en blanco
2. Conversión de formato unicode a ascii
3. Expansión de contracciones gramaticales
4. Eliminación de símbolos de puntuación
5. Tokenización
6. Eliminación de *stopwords*
7. *Lemmatization*

2.3 Modelo basado en $tf - id$

La clase base implementada en la solución permite el cálculo de medidas relevantes para modelos que utilizan este estadístico.

1. Tabla tf: debido a que las tablas tf generalmente son esparcidas la implmentación se realizó utilizando utilizando un índice invertido el cual asocia a cada término t_i con pares (d_j, f_{ij}) manteniendo un acceso eficiente mediante el uso de diccionarios y reduciendo el costo de memoria de almacenar dicha tabla.
2. Tabla df: esta tabla no se almacena explícitamente debido a que la cantidad de documentos en las que está presente un término es igual a $\textbf{len}(tf[t_i])$.

3. Tabla *idf*: se utiliza un diccionario para implementar esta tabla, sin embargo el cálculo de *idf* se definió como un método abstracto debido a que existen múltiples formas de calcular *idf*.

4. Tabla *doc_len*: asocia cada documento con la cantidad de términos distintos que aparecen en él.

5. Cardinalidad del corpus.

El proceso de *scoring* también se puede implementar de forma genérica, sin embargo, debido a optimizaciones en uso de memoria y precálculo resulta más eficiente la sobrescritura de este método en cada implementación.

A continuación definimos el procedimiento general para el *scoring* de documentos que utilizaremos en las implementaciones de los modelos.

```
for every query term q in Q do
  get indexed documents at tf table
  for each document d indexed
    score(d) = score(d) + weight(q,d)
normalize scores
sort documents
```

Figura 2: Interfaz de procesador de texto.

Esta forma de evaluación permite descartar aquellos documentos en los cuales no aparece ningún término relevante a la consulta y la cantidad de índices accedidos está acotado por la cantidad de términos que aparecen en la consulta. Resultando más eficiente que una evaluación de similitud vector a vector en el corpus.

2.4 Modelo vectorial

El modelo vectorial implementado se realizó de acuerdo a lo definido en las conferencias impartidas durante la asignatura, sin embargo, existen consideraciones especiales que deben de ser comprobadas en trabajo futuro.

1. El cálculo de *idf* se realiza mediante la fórmula $log(\frac{N}{n_i})$ la cual puede presentar problemas numéricos en casos particulares, esta es suavizada para evitar estos problemas obteniendo la expresión $log(\frac{N}{1+n_i}) + 1$.

2. El factor de normalización utilizado en el modelo vectorial implica procesar cada documento del corpus para computar su clausura, además esto es necesario cada vez que se modifica el corpus del sistema, este procesamiento es muy costoso por lo cual se propone en [1] la utilización de la raíz cuadrada de la cantidad de términos de un documento como factor de normalización y se presentaron resultados satisfactorios en su utilización en conjuntos de prueba.

2.5 Modelo Okapi

El modelo Okapi es un modelo probabilístico el cual utiliza la frecuencia de términos en lugar del modelo de representación binaria. Se sugiere este modelo debido a que permite la implementación de esquemas de retroalimentación de ambos modelos, tanto clustering como enfoques de retroalimentación para la optimación de los parámetros del modelo. Por lo que se especula que con esquemas de retroalimentación se alcancen mejores resultados que el modelo vectorial.

El cálculo de *idf* se realiza mediante la fórmula $log(1 + \frac{N-n_i+0.5}{n_i+0.5})$

La función de similitud utilizada es

$$\sum_{t \in q} \log(\frac{N}{n_t}) \frac{(k_1+1)tf_{td}}{k_1((1-b)+b(L_d/L_ave)) + tf_{td}}$$

donde k_1 y b son parámetros de estimación, L_d es la cantidad de términos del documento y L_ave es la media de términos por documento en el corpus.

2.6 Corpus

Para probar el modelo se utilizó el corpus Cranfield, este fue obtenido a través de la API de **ir_datasets** la cual provee acceso a colecciones de documentos utilizados para la evaluación de sistemas de recuperación y otras áreas de NLP.

3. Conclusiones y trabajo futuro

En este trabajo se ha realizado la implementación de un modelo abstracto de recuperación de información que permita implementar de múltiples modelos basados en la utilización de medidas $tf - idf$ para ello se ha provisto una solución en un marco experimental como es Jupyter Notebook con la implementación del modelo vectorial y okapi. En la próxima etapa de desarrollo se propone la creación de un modulo de evaluación que permita realizar experimentos sobre estos modelos utilizando los corpus provistos por la API de **ir_datasets** lo cual permita determinar cuál de estos modelos debe de ser empleado en la solución final debido a que en la etapa actual de desarrollo todavía no se han implementado esquemas de retroalimentación, los cuales pudiesen tener un gran impacto en el rendimiento de ambos modelos.

Referencias

[1] Dik L. Lee, Huei Chuang, Kent Seamons. Document Ranking and the Vector-Space Model. 1997

[2] MacAvaney, Sean and Yates, Andrew and Feldman, Sergey and Downey, Doug and Cohan, Arman and Goharian, Nazli. Simplified Data Wrangling with ir_datasets, 2021, SIGIR.

[3] Gesare Asnath Tinega, Waweru Mwangi, Richar Rimiru. Text Mining in Digital Libraries using OKAPI BM25 Model. 2018