Taller 1 Procesamiento de Lenguaje Natural

Integrantes:

Santiago Martínez

Camilo Castañeda

Punto 1.

Se implementan diversas métricas para evaluar sistemas de recuperación de información (IR) con relevancia binaria y numérica.

Precision (Relevancia Binaria):

La función *precision* calcula la precisión de un conjunto de resultados donde la relevancia es binaria (0 o 1). Devuelve el porcentaje de documentos relevantes recuperados en los resultados.

Precision en K (Relevancia Binaria).

La función *precision_at_k* calcula la precisión en los primeros K resultados, donde la relevancia es binaria (0 o 1).

Recall en K (Relevancia Binaria).

La función *recall_at_k* calcula el *recall* en los primeros K resultados, donde la relevancia es binaria (0 o 1).

Average Precision (Relevancia Binaria).

La función *average_precision* calcula la precisión promedio de un conjunto de resultados donde la relevancia es binaria. Se supone que el vector binario de entrada contiene todos los documentos relevantes.

Mean Average Precision (MAP) (Relevancia Binaria).

La función *mean_average_precision* calcula el promedio de la precisión promedio para una lista de vectores binarios, cada uno representando un conjunto de resultados de consulta.

DCG at K (Relevancia como Números Naturales).

La función *dcg_at_k* calcula el Descuento Acumulado en los primeros K resultados, donde la relevancia es un número natural.

NDCG at K (Relevancia como Números Naturales).

La función *ndcg_at_k* calcula el *Normalized Discounted* Cumulative Gain (**NDCG**) en los primeros K resultados, donde la relevancia es un número natural.

Punto 2:

En la implementación se realizó lo siguiente:

Preprocesamiento de Texto.

Se descargan recursos de *stopwords* del paquete **NLTK** y se configura **NLTK** para el preprocesamiento de texto. Se define una función llamada *preprocess_text* que normaliza el texto (minúsculas y eliminación de espacios innecesarios), tokeniza el texto, aplica el proceso de *stemming* y elimina palabras vacías.

Extracción de Texto Sin Procesar.

Se define una función llamada *extract_raw_text* que extrae el texto sin procesar de archivos XML (.naf). Opcionalmente, puede incluir el título del documento.

Creación de un Índice Invertido.

Se crea un índice invertido que asocia términos con la lista de documentos que los contienen. Se calcula la frecuencia de términos por documento.

Consultas Booleanas.

Se define una función llamada *boolean_query* para realizar consultas booleanas en el índice invertido. Esta función permite operaciones de AND y NOT entre términos. Se realizan ejemplos de consultas booleanas como "*confused*," "*Anne*," "*confused AND Anne*" "*confused NOT Anne*" y "*confused OR Anne AND Language NOT William*."

Consultas de Búsqueda Binaria.

Se realizan consultas de búsqueda binaria utilizando un conjunto de consultas almacenadas en archivos *XML*. Estas consultas se procesan y se realiza una búsqueda de documentos relevantes utilizando el índice invertido. Los resultados de estas consultas se escriben en un archivo de resultados llamado "***BSII-AND-queries_results.tsv***."

Punto 3:

La representación tf-idf es eficiente para la recuperación de información en colecciones de documentos moderadamente grandes. Con 331 documentos el cálculo de vectores **tf-idf** no debería ser excesivamente costoso. Respecto al cálculo de la similitud coseno entre la consulta y cada uno de los 331 documentos también puede llevar tiempo, ya que implica operaciones matriciales. Sin embargo, este cálculo generalmente es rápido en comparación con el cálculo de los vectores TF-IDF. Para grandes volúmenes de corpus el cálculo del índice invertid tiende a ser más ineficiente.

En la implementación se realizó lo siguiente:

Extracción de datos de archivos comprimidos.

Se especifican los nombres de archivos comprimidos ('docs-raw-texts.zip' y 'queries-raw-texts.zip') que contienen datos. Se extraen los archivos comprimidos y se almacenan en carpetas con los mismos nombres que los archivos comprimidos, eliminando la extensión ".zip".

Preprocesamiento de texto.

Se descargan recursos de *stopwords* del paquete NLTK (Natural Language Toolkit).

Se define una función *preprocess_text* para preprocesar texto, que incluye normalización a minúsculas, tokenización, eliminación de palabras vacías y aplicar stemming.

Extracción de texto sin procesar.

Se define una función *extract_raw_text* para extraer el texto sin procesar de archivos XML (.naf), opcionalmente incluyendo el título del documento.

Creación de un índice invertido.

Se crea un índice invertido que asocia términos con la lista de documentos que los contienen. Se calcula la frecuencia de términos por documento.

Cálculo de vectores TF-IDF.

Se define una función *calculate_tfidf_vector* para calcular el vector TF-IDF de una consulta. Se calcula el vector TF-IDF para una consulta de ejemplo.

Cálculo de similitud coseno.

Se define una función *calculate_cosine_similarity* para calcular la similitud coseno entre dos vectores. Se define una función *retrieve_and_rank_documents* que recupera y clasifica documentos basados en sus puntajes de similitud coseno con una consulta.

Escritura de resultados a un archivo.

Los resultados de las consultas se escriben en un archivo llamado "**RRDV-consultas_resultados.tsv**".

Cálculo de métricas de evaluación.

Se cargan los juicios de relevancia desde un archivo ("relevance-judgments.tsv") y los resultados de las consultas desde el archivo anterior.

Se calculan métricas de evaluación como Precisión en M (P@M), Recall en M (R@M), y NDCG en M (NDCG@M) para cada consulta.

Se calcula el Promedio del Puntaje Promedio (MAP) para evaluar la calidad general de las recuperaciones.

Implementación con GENSIM

En la implementación se realizó lo siguiente:

Se utiliza Gensim para realizar el preprocesamiento de texto, incluyendo tokenización, eliminación de stopwords y stemming (Porter Stemmer).

Creación del Índice Invertido con Gensim:

Se utiliza Gensim para crear un diccionario de corpora y un modelo TF-IDF.

La tokenización y el procesamiento se hacen utilizando Gensim en una sola línea.

Cálculo de TF-IDF con Gensim:

Gensim se utiliza para calcular automáticamente los vectores TF-IDF para las consultas y documentos.

Cálculo de Similitud Coseno con Gensim:

Gensim se utiliza para calcular la similitud coseno entre vectores TF-IDF.

Escritura de Resultados a un Archivo con Gensim:

Los resultados de las consultas se escriben en un archivo TSV utilizando Gensim.

Cálculo de Métricas de Evaluación con Scikit-Learn:

Se utilizan funciones de Scikit-Learn para calcular métricas como Precisión en K (precision_score), Recall en K (recall_score) y NDCG en K (calculado manualmente).

Con GENSIM simplifica y automatiza tareas relacionadas con el procesamiento de texto, el cálculo de vectores TF-IDF y la similitud coseno.