ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

Sistema de Recuperación de Información basado en Reuters-21578

Autor:

Bruce Soto

Quito, Ecuador - 19 de junio de 2024

Contenido

1. INTRODUCCIÓN 2
1.1. Objetivo 2
2. MARCO TEÓRICO 2
2.1. Reuters-21578 2
2.2. Eliminación de caracteres no deseados 2
2.3. Normalización de texto 2
2.4. Tokenización 3
2.5. Stemming 3
2.6. Bag of Words (BoW) 4
2.7. TF-IDF 4
2.8. Índice invertido 4
2.9. Similitud coseno 5
2.10. Precisión, recall, F1-score 5
3. DESARROLLO 6
3.1. El preprocesamiento de documentos 6
3.2. Representación de Datos en Espacio Vectorial 7
3.3. Indexación 7
3.4. Diseño del Motor de Búsqueda 8
4. EVALUACIÓN DEL SISTEMA 8
5. CONCLUSIONES Y RECOMENDACIONES 12
6. BIBLIOGRAFÍA 12

1. INTRODUCCIÓN

1.1. Objetivo

El objetivo de este proyecto es diseñar, construir, programar y desplegar un Sistema de Recuperación de Información (SRI) utilizando el corpus Reuters-21578. El proyecto se dividirá en varias fases, que se describen a continuación.

2. MARCO TEÓRICO

2.1. Reuters-21578

Reuters-21578: Un conjunto de datos de noticias financieras

Reuters-21578 es una colección de documentos de noticias que aparecieron en el teletipo de Reuters en 1987. Es uno de los conjuntos de datos más utilizados para la investigación en categorización de texto.

2.2. Eliminación de caracteres no deseados

La eliminación de caracteres no deseados y la normalización de texto son dos tareas importantes en el procesamiento del lenguaje natural (PLN). Estas tareas son necesarias para preparar el texto para su posterior análisis y procesamiento.

La eliminación de caracteres no deseados implica eliminar del texto caracteres que no son relevantes para el análisis o procesamiento posterior. Estos caracteres pueden incluir:

- Caracteres de control, los cuales no se imprimen y se utilizan para controlar la presentación del texto, como los saltos de línea y las tabulaciones.
- Caracteres especiales que tienen un significado especial en el lenguaje de programación o el formato de archivo que se utiliza, como comillas o barras diagonales.
- Caracteres de puntuación como puntos, comas y signos de interrogación.
- Espacios en blanco

2.3. Normalización de texto

La normalización de texto puede hacerse usando expresiones regulares o funciones de cadena, lo cual implica convertir el texto a una forma consistente. Esto puede incluir:

- Convertir todo el texto a minúsculas o mayúsculas.
- Eliminar los acentos de las letras.
- Convertir caracteres especiales a representaciones equivalentes, como convertir comillas dobles en comillas simples.
- Eliminar espacios en blanco adicionales y normalizar el espaciado entre palabras.

La normalización de texto puede hacerse usando expresiones regulares o funciones de cadena.

2.4. Tokenización

La tokenización es el proceso de dividir una cadena de texto en unidades más pequeñas llamadas tokens. Estos tokens pueden ser palabras, caracteres o cualquier otro tipo de unidad que sea útil para el procesamiento del texto. La tokenización se utiliza en una amplia variedad de aplicaciones de procesamiento del lenguaje natural (PLN), como:

- Identificar las palabras y frases clave en un texto y luego analizar su sentimiento general.
- Crear un índice de palabras que permita a los usuarios buscar rápidamente documentos relevantes.
- Dividir un texto en unidades más pequeñas que luego pueden ser traducidas a otro idioma.
- Identificar las palabras y frases clave en un texto y luego generar un resumen que capture los puntos más importantes.
- Generar nuevos textos a partir de fragmentos de texto existentes.

2.5. Stemming

El stemming es un proceso de normalización de texto en el procesamiento del lenguaje natural (PLN) que consiste en reducir las palabras a su raíz o forma base, también conocida como stem.

El objetivo principal del stemming es agrupar palabras con morfologías similares para mejorar la eficiencia de tareas como:

- Recuperación de información
- Análisis de sentimientos
- Reducción de la dimensionalidad

2.6. Bag of Words (BoW)

El modelo Bolsa de Palabras (Bag of Words, BoW en inglés) es una técnica de representación de texto ampliamente utilizada en el procesamiento del lenguaje natural (PLN) y la minería de texto. A grandes rasgos, funciona de la siguiente manera:

1. Preprocesamiento
    - Se limpia el texto de elementos irrelevantes como puntuación, símbolos y caracteres especiales.
    - Se convierten todas las palabras a minúsculas.
    - Se elimina la segmentación de palabras, si la hay.
2. Creación del vocabulario
    - Se extraen todas las palabras únicas del conjunto de documentos que se está analizando.
    - Este conjunto de palabras únicas conforma el vocabulario.
3. Representación vectorial
    - Cada documento se representa como un vector de características.
    - Cada posición en el vector corresponde a una palabra del vocabulario.
    - El valor en cada posición indica la frecuencia de aparición de la palabra correspondiente en el documento.

2.7. TF-IDF

TF-IDF significa Frecuencia de Término - Frecuencia Inversa de Documento. Es una medida estadística que se utiliza para evaluar la relevancia de una palabra clave para un documento específico dentro de un conjunto de documentos. En otras palabras, TF-IDF nos dice qué tan importante es una palabra para un documento en particular, en comparación con su importancia en todos los demás documentos.

2.8. Índice invertido

Un índice invertido es una estructura de datos fundamental para la recuperación de información, especialmente en motores de búsqueda. Su función principal es agilizar la búsqueda de palabras clave dentro de un gran conjunto de documentos textuales.

Componentes clave de un índice invertido:

- Lista de todos los términos únicos que aparecen en la colección de documentos.
- Cada documento se identifica con un número único.
- Para cada término del vocabulario, se registra una lista de documentos donde aparece, junto con la posición o frecuencia de aparición dentro de cada documento.

2.9. Similitud coseno

La similitud del coseno, también conocida como similitud coseno o cosine similarity, es una medida que se utiliza para calcular la similitud entre dos vectores. Se basa en el coseno del ángulo entre los vectores en un espacio vectorial. El coseno del ángulo representa la dirección relativa de los vectores.

Se debe tener dos vectores como flechas en un espacio. La similitud del coseno dice qué tan alineados están esos vectores. Si los vectores apuntan en la misma dirección, el ángulo entre ellos es 0 grados y la similitud del coseno es 1, lo que significa que son muy similares. Si los vectores son perpendiculares entre sí, el ángulo entre ellos es de 90 grados y la similitud del coseno es 0, lo que significa que son muy diferentes. Los valores intermedios entre 0 y 1 indican grados variables de similitud.

2.10. Precisión, recall, F1-score

La precisión, el recall y el F1-score son métricas comunes utilizadas para evaluar el rendimiento de modelos de clasificación en problemas de aprendizaje automático y procesamiento del lenguaje natural.

- Precisión: Se define como la proporción de casos positivos que el modelo identifica correctamente como positivos. Se calcula de la siguiente manera:
  (Verdaderos Positivos) / (Verdaderos Positivos + Falsos Positivos)
- Recall: Se define como la proporción de casos positivos reales que el modelo identifica correctamente. Se calcula de la siguiente manera:
  (Verdaderos Positivos) / (Verdaderos Positivos + Falsos Negativos)
- F1-score: Es una medida que combina la precisión y el recall en una única métrica. Se calcula de la siguiente manera:
  2 * (Precisión * Recall) / (Precisión + Recall)

3. DESARROLLO

3.1. El preprocesamiento de documentos

Este proceso involucra varias etapas, cada una con un propósito específico para limpiar y normalizar los datos. A continuación, se detallan los pasos necesarios que se aplican a cada documento:

Eliminación de caracteres no deseados y normalización del texto

- Se eliminan todos los caracteres que no son alfabéticos. Esto incluye números, signos de puntuación y caracteres especiales.
- Se convierte todo el texto a minúsculas para asegurar que las comparaciones entre palabras sean insensibles a mayúsculas y minúsculas.

Tokenización del texto

El texto se divide en palabras individuales utilizando la función word_tokenize de la biblioteca NLTK.

Eliminación de stop words y aplicación de stemming

- Se eliminan las stop words, que son palabras comunes que no aportan un significado sustancial al análisis del texto.
- Se aplica stemming o el proceso de reducir las palabras a su raíz o forma base, eliminando sufijos y otros afijos.

La función de la Ilustración 1 me permite evidenciar todo el proceso de preprocesamiento de datos.

```
def preprocess_document(content):
    text = re.sub(r'\s+', ' ', content)  #remover espacios extra
    text = re.sub(r'[^a-zA-Z]', ' ', text)  #mantener solo caracteres alfabeticos
    text = text.lower()  #convertir a minusculas

    tokens = word_tokenize(text, language='english') #tokenizacion

    tokens = [ps.stem(word) for word in tokens if word not in stop_words and len(word) > 1] #eliminar stop words y aplicar stemming

    return tokens
```

Ilustración 1 Document preprocess

3.2. Representación de Datos en Espacio Vectorial

Bag of Words (BoW)

La técnica de Bag of Words (BoW) convierte los documentos en vectores de frecuencia de términos. La Ilustración 2 me muestra cómo aplicar BoW utilizando CountVectorizer de scikit-learn.

```
def apply_bow(documents):
    vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(documents)
    return X, vectorizer
```

Ilustración 2 BoW

TF-IDF

La técnica TF-IDF (Term Frequency-Inverse Document Frequency) ajusta la representación de BoW para dar más peso a las palabras significativas. La Ilustración 3 me muestra cómo aplicar TF-IDF utilizando TfidfVectorizer de scikit-learn.

```
def apply_tfidf(documents):
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(documents)
    return X, vectorizer
```

Ilustración 3 TF-IDF

3.3. Indexación

Construcción del Índice Invertido

El índice invertido es la estructura de datos que me va a permitir realizar búsquedas más eficientes en el conjunto de documentos *Reuters*. La Ilustración 4 me muestra cómo construir un índice invertido utilizando defaultdict de la biblioteca collections.

```
def build_inverted_index(X, vectorizer, filenames):
    inverted_index = defaultdict(list)
    terms = vectorizer.get_feature_names_out()
    for doc_id, doc in enumerate(X.toarray()):
        for term_id, term_freq in enumerate(doc):
            if term_freq > 0:
                term = terms[term_id]
                inverted_index[term].append(filenames[doc_id])
    return inverted_index
```

Ilustración 4 Build inverted index

3.4. Diseño del Motor de Búsqueda

Para la creación del motor de búsqueda se realizaron varios pasos dentro de la función con el fin de tener mejores resultados, para esto se preprocesa la consulta, encuentra documentos relevantes, calcula similitudes de coseno, y devuelve los documentos más relevantes ordenados por similitud, limitándose a aquellos con una similitud mayor a 0.35 (más del 35% de similitud) tal y como lo muestra la Ilustración 5.

```
def search(query, inverted_index, vectorizer, X, filenames, top_n=None):
    query = preprocess_query(query)
    query_terms = query.split()

    filename_to_index = {filename: idx for idx, filename in enumerate(filenames)} #mapear nombres de archivos a sus indices
    relevant_docs = set()
    for term in query_terms:
        if term in inverted_index:
            relevant_docs.update(inverted_index[term])
    relevant_docs = [filename_to_index[doc] for doc in relevant_docs if doc in filename_to_index] #convertir nombres de archivos
     relevantes a indices
    if not relevant_docs:
        return [], []
    X_relevant = X[relevant_docs]
    filenames_relevant = [filenames[i] for i in relevant_docs]

    #procesar la consulta y calcular similitudes
    query_vector = vectorizer.transform([query])
    similarities = cosine_similarity(query_vector, X_relevant).flatten()

    #ordenar documentos por similitud y limitar a los que tengan una siminitud de mas del 35%
    ranked_indices = np.argsort(similarities)[::-1][:]
    ranked_filenames = [filenames_relevant[i] for i in ranked_indices if similarities[i] > 0.20]

    return ranked_filenames, similarities[ranked_indices]
```

Ilustración 5 Search

4. EVALUACIÓN DEL SISTEMA

Con la finalidad de obtener mejores resultados, en el desarrollo del sistema se indico que solo se muestran los archivos que tienen una similitud mayor al 35%, lo que me permite tener solo los mejores resultados para la búsqueda de una consulta. Para la evaluación del sistema primero se realizó pruebas con algunas consultas, todas esas consultas son parte de *cats.txt,* el cual es el archivo que contiene las categorías de cada uno de los archivos. A continuación, se muestran los resultados para cada técnica, con su archivo, similitud y la categoría real a la que pertenece.

Categoría *housing (hous)*

Resultados para BoW		
Archivo	**Similitud**	**Categoría**
11665	0.589506	hous
3898	0.503953	hous
9374	0.384900	suggar

Tabla 1 Consulta "housing" con BoW

Resultados para TF-IDF		
Archivo	**Similitud**	**Categoría**
11665	0.556163	hous
3898	0.461386	hous
11170	0.386236	hous
9374	0.378478	sugar

Tabla 2 Consulta "housing" con TF-IDF

Categoría *soybean oilseed (oilse, soybean)*

Resultados para BoW		
Archivo	**Similitud**	**Categoría**
5702	0.466942	oilse, soybean
9617	0.441894	soybean, oilse
7356	0.392604	soybean, oilse
6906	0.359856	soybean, oilse

Tabla 3 Consulta " soybean oilseed " con BoW

Resultados para TF-IDF		
Archivo	**Similitud**	**Categoría**
3540	0.461336	coconut oil, palm oil, veg oil, soybean, oilse
5702	0.458457	oilse, soybean
9617	0.420977	soybean, oilse
7356	0.363090	soybean, oilse
3888	0.361270	oilse

Tabla 4 Consulta "soybean oilseed " con TF-IDF

Precisión, recall y F1-score

La precisión es la fracción de documentos relevantes recuperados con respecto al total de documentos recuperados. En este caso, la precisión usando BoW es 0.6482 y usando TF-IDF es 0.6538. Esto significa que, en promedio, el 64.82% de los documentos recuperados usando BoW y el 65.38% usando TF-IDF eran relevantes. La precisión es ligeramente mejor con TF-IDF, lo que indica que esta técnica es algo mejor en filtrar documentos irrelevantes en comparación con BoW.

El recall es la fracción de documentos relevantes recuperados con respecto al total de documentos relevantes disponibles. Ambos métodos, BoW y TF-IDF, tienen un recall similar a sus respectivas precisiones (0.6482 para BoW y 0.6538 para TF-IDF). Esto sugiere que ambas técnicas son igualmente efectivas en encontrar la mayoría de los documentos relevantes disponibles, aunque TF-IDF tiene una ligera ventaja.

El F1-score es la media armónica de la precisión y el recall. Es una medida combinada de precisión y recall que busca un equilibrio entre ambas. En este caso, los F1-scores son 0.6482 para BoW y 0.6538 para TF-IDF. Esto refuerza la observación de que TF-IDF tiene un rendimiento ligeramente mejor en general, debido a su capacidad de mantener un buen equilibrio entre precisión y recall.

La Ilustración 6 me muestra todos los resultados mencionados anteriormente.

```
Resultados de evaluacion usando BoW:
Precision: 0.6482
Recall: 0.6482
F1-score: 0.6482

Resultados de evaluacion usando TF-IDF:
Precision: 0.6538
Recall: 0.6538
F1-score: 0.6538
```

Ilustración 6 Evaluation results

Para la interfaz de búsqueda se hizo una interfaz básica la cual acepta consultas y funciona enviando la consulta al backend (servidor realizado con flask) y esta consulta es preprocesada y como en las tablas 2 y 4, es decir, se usa solo la técnica TF-IDF ya que es la que mejores resultados me dio al momento de evaluar el sistema. Posterior a esto el servidor me devuelve los archivos ordenados (de mayor a menor similitud) que tengan una similitud mayor al 35% tal y como lo muestra el ejemplo de la Ilustración 7.

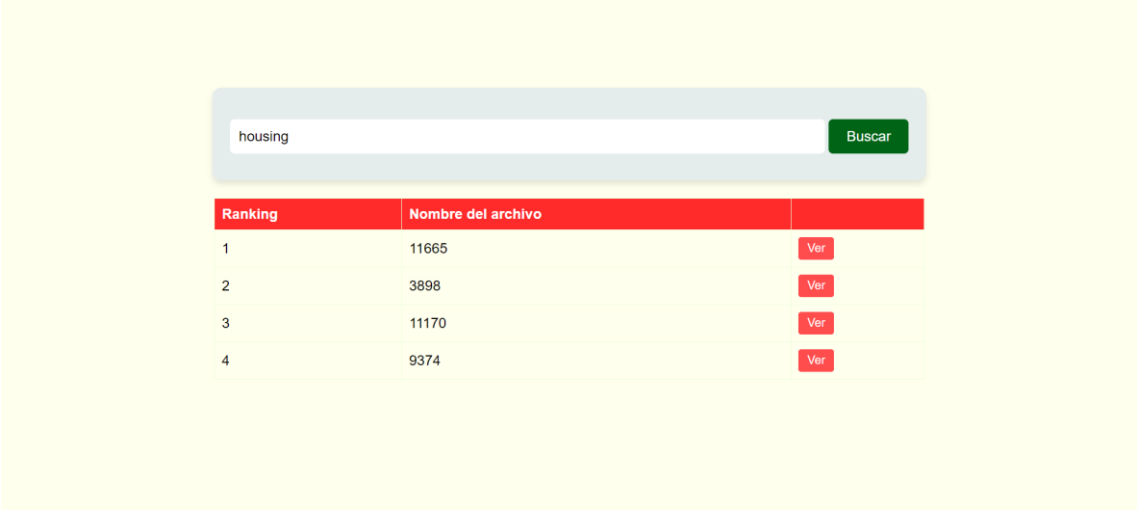

Ilustración 7 Results of "housing"

5. CONCLUSIONES Y RECOMENDACIONES

- Se ha demostrado que la aplicación de técnicas de preprocesamiento como la limpieza de datos, la tokenización, la eliminación de stop words y el stemming/lematización mejora la precisión de los resultados de búsqueda, así como la evaluación de diferentes técnicas de vectorización, como Bag of Words (BoW) y TF-IDF, ha permitido identificar la más adecuada para el corpus de datos utilizado.
- La construcción de un índice invertido que mapee términos a documentos permite realizar búsquedas rápidas y eficientes ya que esto ha mejorado significativamente el rendimiento del motor de búsqueda.
- El uso de algoritmos de similitud como la "similitud coseno" ha permitido identificar los documentos más similares a la consulta del usuario y la implementación de un algoritmo de ranking ha permitido ordenar los resultados de búsqueda por relevancia, priorizando los documentos más relevantes para el usuario.
- Se recomienda ajustar el umbral de similitud mínimo para mostrar resultados a un valor superior al 35%, esto permitirá filtrar resultados menos relevantes y mejorar la experiencia del usuario.

6. BIBLIOGRAFÍA

[1] Dialnet, 2022. Aprendizaje automático y la colección reuters-21578 en la clasificación de documentos. [En línea] Available at: https://dialnet.unirioja.es/servlet/articulo?codigo=9377913

[2] Transformación Digital, 2024. Tokenización: ¿qué es y cómo utilizarla?. [En línea] Available at: https://www.sydle.com/es/blog/tokenizacion-65e8cd225d245147cf6e412e

[3] Seobility Wiki, 2024. Stemming. [En línea] Available at: https://www.seobility.net/es/wiki/Stemming

[4] Keytrends, 2023. Bag of Words en la inteligencia artificial. [En línea] Available at: https://keytrends.ai/es/academy/glosario/inteligencia-artificial/bag-of-words#:~:text=El%20Bag%20of%20Words%20(BoW,las%20palabras%20en%20el%20texto.

[5] Casarotto, C., 2021. TF-IDF: el abordaje de optimización on page que tu blog necesita. [En línea] Available at: https://rockcontent.com/es/blog/que-es-tf-idf/

[6] Optimize360, 2024. Entender el índice invertido en SEO: la clave para optimizar su posicionamiento en buscadores. [En línea] Available at: https://www.optimize360.fr/es/definicion/indice-inverso/

[7] Scikit Learn, 2024. precision_recall_fscore_support. [En línea] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

[8] GRAPH everywhere, 2024. Algoritmo de similitud de coseno. [En línea] Available at: https://www.grapheverywhere.com/algoritmo-de-similitud-de-coseno/