GoSoccer

Ángel Manuel Ferrer Álvarez
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
angferalv@alum.us.es FBG8620

Manuel Blánquez Galobart
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
manblagal@alum.us.es TKH5786

***Resumen*—El objetivo principal de este trabajo es diseñar, implementar y evaluar sistemas de recuperación de información aplicados a un corpus textual compuesto por fichas de jugadores de fútbol. Para ello, se desarrollan dos modelos de recuperación: uno basado en consultas booleanas y otro en consultas de texto libre utilizando la métrica tf-idf y similitud del coseno. Este enfoque permite analizar comparativamente el rendimiento de ambos modelos frente a una colección de necesidades de información previamente definidas y etiquetadas en términos de relevancia, con el fin de observar el impacto de las técnicas de normalización lingüística (como el stemming) en la precisión y sensibilidad de los resultados.**

Los resultados que hemos obtenido evidencian que el uso de técnicas de normalización como el stemming tiene un impacto relevante, particularmente en los modelos de texto libre, y que existe un compromiso inherente entre precisión y sensibilidad en función del tipo de índice empleado.

I. Introducción

Hasta los años 90, la mayoría de las personas prefería obtener información directamente de otras personas en lugar de usar sistemas automáticos de recuperación. Sin embargo, en la última década, la mejora continua de estos sistemas ha llevado a que los motores de búsqueda web sean la fuente preferida de información para la mayoría. La recuperación de información no nació con la web; comenzó facilitando el acceso a publicaciones científicas y registros bibliotecarios, y se extendió a otros contenidos usados por profesionales como periodistas y abogados. Hoy en día, su aplicación abarca también la gestión de información no estructurada en ámbitos corporativos y gubernamentales. El enorme crecimiento de la información en la web habría sido inútil sin sistemas eficientes que permitan encontrar y analizar datos relevantes adaptados a las necesidades de cada usuario [1].

El objetivo principal de la RI es procesar grandes colecciones de documentos no estructurados para ofrecer respuestas precisas y relevantes a consultas formuladas por los usuarios. Para ello, existen diferentes modelos y técnicas que permiten transformar los textos en representaciones computacionales adecuadas, facilitando el proceso de búsqueda. Entre estos, el modelo booleano se basa en la presencia o ausencia de términos en documentos y utiliza operadores lógicos para realizar consultas exactas. Por otro lado, los modelos basados en recuperación por texto libre, como el modelo vectorial con pesos TF-IDF y similitud del coseno, permiten ordenar los documentos por grado de relevancia, lo que es especialmente útil en colecciones grandes y heterogéneas [1].

El proyecto se centra en la aplicación práctica de estos dos modelos a un corpus específico de fichas de jugadores de fútbol [2], explorando distintas estrategias de normalización y procesamiento del texto (como tokenización, eliminación de palabras vacías y stemming) para mejorar la eficacia de la búsqueda. Asimismo, se evaluará la calidad del sistema para valorar la efectividad de un sistema de RI.

Después vienen un par de párrafos para dar un poco más de detalle sobre el trabajo realizado. Hay que indicar la problemática o el objetivo que se marca, y cómo se ha enfocado la solución propuesta en este trabajo. Finalmente, el último párrafo se dedica a comentar la estructura del documento por secciones, como el que sigue.

II. Corpus recopilado

El corpus seleccionado ha sido las fichas de jugadores de fútbol, estas se han obtenido a través de la api de wikipedia. Se ha seleccionado ese corpus dada a su variedad de información perfecta para evaluar un sistema de recuperación de la información.

III. Necesidades de información

Para evaluar los sistemas de recuperación desarrollados, se ha diseñado un conjunto de veinte necesidades de información. Se han elaborado tanto para el modelo booleano como para el modelo de recuperación basado en el esquema tf-idf, con el objetivo de cubrir una amplia gama de características relevantes de los jugadores del corpus.

A. Consultas para el modelo booleano

Las siguientes consultas utilizan operadores lógicos para delimitar con precisión el conjunto de documentos relevantes:

- **Porteros:** *portero NOT delantero NOT defensa NOT centrocampista*
- **Defensas:** *defensa NOT delantero NOT portero NOT centrocampista*
- **Centrocampistas:** *centrocampista NOT delantero NOT defensa NOT portero*
- **Delanteros:** *delantero NOT defensa NOT portero NOT centrocampista*
- **Argentinos:** *argentino*
- **Españoles:** *español*
- **Italianos:** *italiano*
- **Franceses:** *francés*

- **Brasileños:** *brasileño*
- **Holandeses:** *holandés*
- **Portugueses:** *portugués*
- **Capitanes:** *capitán*
- **Ganadores del Balón de Oro:** *ganar AND balón AND oro*
- **Ganadores de la Champions:** *ganar AND champions*
- **Ganadores del Mundial:** *ganar AND mundial*
- **Internacionales:** *internacional*
- **Entrenadores:** *entrenador*
- **Exfutbolistas:** *exfutbolista OR retiro OR retirado*
- **Ligados al Inter de Miami:** *inter AND miami*
- **De Fuentealbilla:** *fuentealbilla*

B. Consultas para el modelo tf-idf

Las consultas en texto libre permiten evaluar el sistema en condiciones más cercanas a un entorno de búsqueda natural, con menor control sobre la estructura de la consulta:

- **Porteros:** *portero*
- **Defensas:** *defensa*
- **Centrocampistas:** *centrocampista*
- **Delanteros:** *delantero*
- **Argentinos:** *argentino*
- **Españoles:** *español*
- **Italianos:** *italiano*
- **Franceses:** *francés*
- **Brasileños:** *brasileño*
- **Holandeses:** *holandés*
- **Portugueses:** *portugués*
- **Capitanes:** *capitán*
- **Ganadores del Balón de Oro:** *ganador de balón de oro*
- **Ganadores de la Champions:** *ganador de champions*
- **Ganadores del Mundial:** *ganador de mundial*
- **Internacionales:** *internacional*
- **Entrenadores:** *entrenador*
- **Exfutbolistas:** *exfutbolista*
- **Ligados al Inter de Miami:** *inter de miami*
- **De Fuentealbilla:** *fuentealbilla*

C. Justificación

La selección de estas necesidades de información responde a la intención de abarcar diferentes niveles de complejidad semántica y dificultad de recuperación. Algunas consultas son directas y específicas, mientras que otras requieren una interpretación más compleja de los documentos. Esta variedad es clave para realizar una evaluación más robusta del desempeño de ambos sistemas.

IV. Preliminares

En esta sección se hace una breve introducción de las técnicas empleadas y también trabajos relacionados.

A. Métodos empleados

En este proyecto se han utilizado distintas técnicas para construir y evaluar sistemas de recuperación de información sobre un corpus de fichas de jugadores de fútbol. A continuación, se describen las principales metodologías empleadas.

1) Técnicas de normalización: Para mejorar la calidad del procesamiento de texto y la recuperación, se aplicaron las siguientes técnicas de normalización, a través de la librería NTLK [3], además se uso la librería os para el manejo de carpetas y archivos [6]:

- **Tokenización:** proceso de segmentar el texto en unidades básicas llamadas tokens, generalmente palabras o términos, separándolos mediante espacios y signos de puntuación.
- **Eliminación de palabras vacías (stop words):** consiste en eliminar palabras comunes y poco informativas (como “el”, “la”, “y”, “de”) que no aportan valor significativo para la búsqueda, reduciendo el ruido en el índice.
- **Stemming:** técnica que reduce las palabras a su raíz o forma base, eliminando sufijos o variaciones morfológicas para agrupar diferentes formas de una palabra bajo un mismo término, facilitando la correspondencia entre consulta y documento.

2) Índices invertidos: Para la implementación del modelo booleano, se utilizaron índices invertidos, una estructura que asocia cada término del vocabulario con la lista de documentos donde aparece. Esto permite recuperar rápidamente todos los documentos que contienen uno o varios términos especificados en la consulta mediante operadores lógicos (AND, OR, NOT). La construcción y manejo de estos índices se realizó utilizando la librería Whoosh en Python [4], que facilita la creación, actualización y consulta eficiente de índices invertidos, además se uso la librería os para el manejo de carpetas y archivos [6].

3) Modelo vectorial con TF-IDF y similitud del coseno: Para la recuperación basada en consultas en texto libre, se emplea el modelo vectorial, que representa tanto los documentos como las consultas como vectores en un espacio multidimensional, donde cada dimensión corresponde a un término del vocabulario [1].

La ponderación de los términos se realiza mediante TF-IDF, una técnica que asigna un peso a cada término según su frecuencia en un documento y su rareza en la colección completa, lo que permite destacar términos discriminativos [1].

La similitud entre la consulta y cada documento se calcula mediante la similitud del coseno, que mide el ángulo entre sus vectores correspondientes. Un valor más alto indica mayor similitud y, por tanto, mayor relevancia del documento para la consulta [1].

La construcción de la matriz TF-IDF y el cálculo de la similitud del coseno se implementaron utilizando la librería scikit-learn [5] de Python, además se uso la librería os para el manejo de carpetas y archivos [6].

4) Evaluación del sistema de recuperación: La evaluación de los sistemas de recuperación implementados se llevó a cabo mediante un conjunto de 20 necesidades de información representativas del dominio futbolístico. Estas necesidades se mapearon a consultas específicas (por ejemplo, “portero”, “argentino”, “entrenador”, etc.), y se definió manualmente qué documentos del corpus eran relevantes para cada una de ellas.

- **Sistema booleano:** se calcularon las métricas de precisión y sensibilidad para cada consulta, utilizando las

herramientas de evaluación de la librería nltk.metrics [1]. Se compararon los documentos recuperados frente a los relevantes y se promedió cada métrica para obtener una visión global del rendimiento del sistema.
- **Sistema vectorial (TF-IDF):** se evaluó mediante la precisión promedio (Average Precision, AP) sobre los 20 primeros documentos recuperados según similitud, utilizando sklearn.metrics [1]. La metodología sigue las directrices descritas en el documento de evaluación [1]. Finalmente, se calculó el Mean Average Precision (MAP) como media de las AP individuales, proporcionando una medida global del sistema basado en ranking.

Las necesidades de información evaluadas incluyeron categorías como nacionalidades, posiciones en el campo, logros deportivos y relaciones con clubes específicos. Cada consulta se asoció a una palabra clave y se creó un diccionario con los documentos relevantes para poder calcular las métricas adecuadamente.

B. Trabajo Relacionado

En este trabajo no se ha seguido directamente ningún proyecto previo, pero sí se han utilizado fundamentos teóricos y herramientas ampliamente reconocidas en el ámbito de la Recuperación de Información.

A nivel teórico, se ha tomado como referencia principal el libro de *An Introduction to Information Retrieval* [1], en el que se describen con detalle conceptos fundamentales como el modelo booleano, el modelo vectorial (TF-IDF) y métricas de evaluación. Estos conceptos han guiado el desarrollo y evaluación del sistema propuesto.

En cuanto a herramientas prácticas, se han empleado diversas librerías de uso común en el campo del procesamiento del lenguaje natural y la recuperación de información:

- **NLTK** [3], para el tratamiento de texto, eliminación de stopwords, tokenización y evaluación de métricas básicas.
- **Whoosh** [4], como motor de búsqueda de texto ligero, para implementar el sistema booleano.
- **Scikit-learn** [5], utilizada para representar los documentos en el espacio vectorial mediante TF-IDF y aplicar técnicas de evaluación.

V. Metodología

Esta sección describe detalladamente el proceso metodológico desarrollado en el trabajo. Como se presentó en la sección de Preliminares, el sistema se basa en cuatro pilares fundamentales: el preprocesamiento del texto mediante técnicas de normalización, la construcción de índices invertidos para el modelo booleano, la implementación del modelo vectorial con TF-IDF y similitud del coseno, y, finalmente, la evaluación sistemática del rendimiento de ambos modelos. Cada una de estas etapas se implementó con herramientas específicas y siguiendo buenas prácticas del campo de la Recuperación de Información. A continuación, se detallan estos componentes en orden secuencial.

A. Normalización del texto

El proceso de normalización textual es el primer paso clave para preparar el corpus antes de construir el índice y aplicar modelos de recuperación. Se desarrollaron dos versiones del proceso: una que incluye stemming y otra que no lo aplica, permitiendo comparar el impacto del uso de raíces léxicas en los resultados.

Las funciones desarrolladas para este propósito fueron las siguientes:

normalizar_texto_con_stem / normalizar_texto_sin_stem: realiza la conversión del texto a minúsculas, lo tokeniza, elimina signos de puntuación y palabras vacías (stopwords). En su versión con stemming, además aplica la reducción de palabras a su raíz mediante SnowballStemmer de NLTK [3].

normalizar_corpus: recorre todos los archivos .txt en una carpeta de entrada, aplica cada normalizador de texto sobre su contenido y guarda los documentos procesados en unas carpetas de salidas, manteniendo el identificador del archivo.

normalizar_consulta_con_stem / normalizar_consulta_sin_stem: estas funciones están diseñadas para procesar consultas escritas por el usuario, manteniendo los operadores booleanos (AND, OR, NOT) intactos y normalizando solo los términos no lógicos, para garantizar la coherencia entre consulta y corpus.

descargar_recursos: función auxiliar que asegura que se descargan los recursos necesarios de NLTK como el tokenizador y las listas de stopwords en español.

Este sistema modular permite aplicar fácilmente cualquiera de las dos variantes de normalización, tanto al corpus como a las consultas, lo que facilita la experimentación y evaluación posterior.

normalizar_texto_con_stem($texto$)
Entrada: una cadena de texto crudo
Salida: texto normalizado con stemming aplicado
$texto \leftarrow$ convertir a minúsculas
$tokens \leftarrow$ tokenizar(texto)
$tokens \leftarrow$ filtrar alfabéticos y quitar stopwords
$stems \leftarrow$ aplicar stemmer a cada token
Devolver concatenación de stems

normalizar_texto_sin_stem($texto$)
Entrada: una cadena de texto crudo
Salida: texto normalizado sin stemming
$texto \leftarrow$ convertir a minúsculas
$tokens \leftarrow$ tokenizar(texto)
$tokens \leftarrow$ filtrar alfabéticos y quitar stopwords
Devolverconcatenación de tokens

normalizar_corpus($carpeta\_entrada, salida\_sin, salida\_con$)
Entrada: ruta de entrada y rutas de salida
Salida: diccionarios con documentos normalizados
para cada archivo en $carpeta\_entrada$ hacer
    $texto \leftarrow$ leer archivo
    $texto\_sin \leftarrow$ *normalizar_texto_sin_stem*($texto$)
    $texto\_con \leftarrow$ *normalizar_texto_con_stem*($texto$)
    guardar $texto\_sin$ en $salida\_sin$
    guardar $texto\_con$ en $salida\_con$
Devolver diccionarios de documentos procesados

Fig. 1. Pseudocódigo de la clase `Normalizador_textos` para procesamiento con y sin *stemming*

crear_indice($corpus\_dir, indice\_dir$)
Entrada: ruta de documentos normalizados y destino del índice
Salida: índice invertido persistente
$schema \leftarrow$ definir campos doc_id (ID) y content (TEXT)
crear o abrir índice en **indice_dir**
$writer \leftarrow$ abrir escritor del índice
para cada archivo .txt en **corpus_dir hacer**
    $doc\_id \leftarrow$ nombre del archivo sin extensión
    $texto \leftarrow$ leer contenido del archivo
    añadir documento al índice con writer.update_document
commit del índice

ejecutar_consulta_boolean($consulta\_norm, indice$)
Entrada: consulta booleana normalizada y ruta del índice
Salida: conjunto de documentos relevantes
$ix \leftarrow$ abrir índice en $indice$
$parser \leftarrow$ QueryParser sobre campo content
$query \leftarrow$ parser.parse(consulta_norm)
ejecutar búsqueda con searcher.search(query)
recoger doc_ids de resultados en un conjunto
Devolver conjunto de documentos relevantes

Fig. 2. Pseudocódigo para creación del índice y ejecución de consultas booleanas

B. Índices invertidos y consultas booleanas

Una vez normalizado el corpus, se procede a indexarlo para habilitar la recuperación eficiente de documentos a partir de consultas booleanas. Para ello, se implementó un índice invertido utilizando la librería **Whoosh** [4], el cual permite almacenar, acceder y consultar documentos en función de los términos que contienen.

Las funciones desarrolladas para este propósito fueron las siguientes:

crear_indice: define un esquema con los campos `doc_id` y `content`, y recorre todos los archivos de una carpeta de entrada. Para cada uno, extrae el contenido textual y lo incorpora al índice mediante la función `update_document`. Este índice se almacena en disco para permitir consultas posteriores de forma eficiente.

ejecutar_consulta_boolean: esta función carga un índice previamente generado, interpreta la consulta booleana normalizada mediante el `QueryParser` de Whoosh y ejecuta la búsqueda sobre el campo `content`. Devuelve un conjunto de identificadores de documentos relevantes.

C. Modelo vectorial: creación y consulta de la matriz TF-IDF

Para implementar el modelo vectorial, se representó cada documento como un vector numérico utilizando la técnica de TF-IDF (*Term Frequency-Inverse Document Frequency*), lo que permite ponderar los términos más representativos del contenido y reducir el peso de los términos comunes. Esta representación posibilita calcular similitudes entre documentos y consultas mediante el coseno del ángulo entre sus vectores.

Las funciones desarrolladas para este propósito fueron las siguientes:

crear_matriz_tfidf: definida en la clase `tf_idf`, esta función recorre todos los documentos normalizados de una carpeta, construye la lista de textos y aplica el vectorizador `TfidfVectorizer` de **scikit-learn** para obtener la matriz TF-IDF y la lista de identificadores de documentos [5].

buscar_documentos: en la clase `consulta_tf_idf`, esta función toma una consulta en texto libre, la transforma usando el mismo vectorizador TF-IDF que el corpus, calcula la similitud del coseno entre la consulta y todos los documentos, y retorna los documentos ordenados por relevancia junto a sus puntuaciones de similitud.

crear_matriz_tfidf($carpeta\_corpus$)
Entrada: carpeta con documentos normalizados
Salida: vectorizador, matriz TF-IDF y lista de identificadores
1 para cada archivo en $carpeta\_corpus$ hacer
2 $doc\_id \leftarrow$ nombre del archivo sin extensión
3 $texto \leftarrow$ leer contenido del archivo
4 guardar texto en lista corpus y su id
5 $vectorizer \leftarrow$ TfidfVectorizer()
6 $tfidf\_matrix \leftarrow$ vectorizer.fit_transform(corpus)
7 **Devolver** $vectorizer, tfidf\_matrix, doc\_ids$

buscar_documentos($consulta, vectorizer, tfidf\_matrix, doc\_ids$)
Entrada: consulta de texto libre y datos del corpus
Salida: lista de documentos relevantes con puntuación
1 $consulta\_tfidf \leftarrow$ vectorizer.transform([consulta])
2 $similitudes \leftarrow$ cosine_similarity entre consulta y matriz
3 $indices \leftarrow$ ordenar índices por similitud descendente
4 $resultados \leftarrow$ filtrar y emparejar con doc_ids
5 **Devolver** $resultados$

Fig. 3. Pseudocódigo para creación y búsqueda con matriz TF-IDF

D. Evaluación de los sistemas

Para valorar la efectividad de los sistemas de recuperación implementados, se definieron un conjunto de necesidades de información agrupadas en 20 categorías relacionadas con futbolistas (nacionalidades, posiciones, logros, etc.), y se estableció manualmente un conjunto de documentos relevantes para cada una.

evaluar_sist_bool_necesidades_info($indice$)
Entrada: nombre del índice a usar
Salida: precisión, sensibilidad y sus medias
1 para cada necesidad de información hacer
2 normalizar la consulta según el índice
3 ejecutar consulta booleana y obtener recuperados
4 comparar con documentos relevantes
5 calcular precisión y sensibilidad
6 calcular media de todas las precisiones y sensibilidades
7 **Devolver** métricas por consulta y sus medias

evaluar_sist_tf_idf($vectorizadores, matrices, ids$)
Entrada: modelos vectorizadores y matrices tf-idf con/sin stemming
Salida: MAP con y sin stemming
1 para cada necesidad de información hacer
2 normalizar consultas con y sin stem
3 buscar documentos en modelo tf-idf correspondiente
4 calcular average precision sobre top-20 resultados
5 calcular la media de las APs (MAP)
6 **Devolver** MAP con y sin stemming

Fig. 4. Evaluación del sistema booleano y TF-IDF

evaluar_sist_bool_necesidades_info: Esta función, implementada en el archivo `evaluar_sist_bool_necesidades_info`, lanza las consultas booleanas sobre cada índice generado (sin normalizar, normalizado sin stemming, y con stemming), calcula precisión y sensibilidad usando **nltk.metrics** [3], y devuelve la media de ambas métricas.

evaluar_sist_tf_idf: Definida en el archivo `evaluar_sist_tf_idf`, esta función lanza cada consulta sobre ambos modelos TF-IDF (sin stemming y con stemming), calcula la *Average Precision* [1] de los 20 primeros resultados recuperados, ponderando por su similitud, mediante **sklearn.metrics.average_precision_score** [5]. Finalmente, se promedia el resultado para obtener el *Mean Average Precision* (MAP) [1].

E. Interfaz y ejecución del sistema

La ejecución de todo el sistema se centraliza en el archivo `main`, que actúa como punto de entrada al programa. Al iniciarse, este script realiza automáticamente varias tareas clave para preparar el entorno:

- Descarga de los recursos necesarios de NLTK (tokenizadores y listas de stopwords).
- Normalización del corpus en sus tres variantes (sin normalizar, con stemming y sin stemming).
- Construcción de los índices invertidos mediante Whoosh para cada versión del corpus.
- Creación de las matrices TF-IDF con y sin stemming.

Tras esta inicialización, el sistema presenta un menú interactivo que permite al usuario elegir entre varias acciones disponibles:

- **Realizar una consulta booleana:** Lanza la interfaz contenida en `Interfaz_consultas_booleanas`, desde donde se puede elegir el índice deseado y realizar consultas usando operadores lógicos (AND, OR, NOT).
- **Realizar una consulta vectorial (TF-IDF):** Accede a la interfaz de `Interfaz_consulta_tf_idf`, que permite introducir consultas en lenguaje natural y obtener documentos ordenados por similitud, usando modelos TF-IDF con y sin stemming.
- **Evaluar el sistema booleano:** Ejecuta el módulo correspondiente para medir precisión y sensibilidad sobre las necesidades de información predefinidas.
- **Evaluar el sistema vectorial:** Lanza la evaluación del modelo TF-IDF mediante el cálculo de Average Precision y Mean Average Precision (MAP).

Este diseño modular e interactivo permite al usuario explorar, consultar y evaluar el sistema de forma clara y eficiente.

VI. Resultados

TABLA I
Medidas promedio de sensibilidad y precisión para las búsquedas booleanas

Método	Sensibilidad promedio	Precisión promedio
Sin normalizar	0.7585	0.6552
Normalizado	0.7585	0.6552
Normalizado + stemming	0.8009	0.5913

Los resultados obtenidos en el sistema booleano (Tabla I) han sido evaluados en tres configuraciones: sin normalización, tokenizando y eliminando las palabras vacías, y con stemming. En el caso del índice sin normalizar, se obtuvo una precisión promedio de 0.6552 y una sensibilidad de 0.7585. Estos mismos valores se mantuvieron al eliminar únicamente las palabras vacías, lo que indica que este tipo de preprocesamiento no afecta a la calidad de los resultados en términos de precisión y sensibilidad.

No obstante, la eliminación de palabras vacías sí aporta ventajas en cuanto a eficiencia, ya que reduce significativamente el tamaño del índice invertido. Al eliminar términos muy frecuentes y poco informativos como “el”, “de” o “en”, el sistema necesita comparar menos términos durante las búsquedas, lo que se traduce en tiempos de respuesta más rápidos y menor consumo de recursos.

Al aplicar stemming, se observó que la sensibilidad aumentó a 0.8009, lo que indica una mejora en la recuperación de documentos relevantes. Sin embargo, la precisión disminuyó a 0.5913, lo que implica que, aunque se recuperaron más documentos relevantes, también se incluyeron más documentos irrelevantes en los resultados.

Este comportamiento pone de manifiesto una limitación del modelo booleano: al aplicar técnicas de normalización agresivas como el stemming, se tiende a recuperar más información pero a costa de una menor precisión. Esto se debe a que el stemming reduce las palabras a su raíz común, lo que puede provocar falsos positivos. Por ejemplo, en un corpus sobre fútbol, los términos “portero” y “portería” podrían reducirse a la raíz “port”, haciendo que un documento sobre porterías de fútbol se recupere al buscar portero, aunque su contenido no sea relevante para el jugador en cuestión.

En resumen, mientras que la eliminación de palabras vacías mejora la eficiencia sin afectar a la calidad, el uso de stemming incrementa la recuperación de información pero con un riesgo mayor de pérdida de precisión, debido a asociaciones semánticas forzadas por la reducción de palabras a raíces comunes.

TABLA II
MEDIDAS PROMEDIO DE MAP PARA EL MODELO TF-IDF

Método	MAP promedio
Con stemming	0.8059
Sin stemming	0.7933

Por otro lado, los experimentos con el modelo tf-idf (TABLA II) mostraron un rendimiento superior en términos de MAP. El sistema basado en tf-idf con stemming alcanzó una precisión promedio más alta, con un MAP de 0.8059, frente a 0.7933 sin stemming. Estos resultados indican que el modelo tf-idf con stemming es más eficaz a la hora de ordenar los documentos relevantes en los primeros lugares del ranking, lo que permite una mejor satisfacción de las necesidades de información del usuario. El uso de stemming en el modelo tf-idf mejoró el rendimiento general, lo que demuestra que la normalización de términos tiene un impacto positivo en la efectividad de la recuperación. Estos experimentos destacan la ventaja del modelo tf-idf para manejar consultas más complejas y generar rankings más precisos en comparación con el sistema booleano.

- Los experimentos realizados, indicando razonadamente la configuración empleada, qué se quiere determinar, y como se ha medido.
- Los resultados obtenidos en cada experimento, explicando en cada caso lo que se ha conseguido.
- Análisis de los resultados, haciendo comparativas y obteniendo conclusiones.

VII. CONCLUSIONES

Los experimentos realizados indican que el modelo basado en TF-IDF combinado con stemming ofrece los mejores resultados en términos de MAP, lo que sugiere que la normalización léxica mediante técnicas como el stemming puede mejorar significativamente el rendimiento en tareas de recuperación de información. Este modelo supera al enfoque booleano al incorporar el análisis de la frecuencia de los términos tanto en el documento como en el corpus completo, lo que permite valorar la relevancia de una palabra no solo por su presencia, sino por su importancia relativa dentro del conjunto de documentos.

En contraste, el modelo booleano sin stemming mantiene una sensibilidad aceptable, pero presenta una precisión relativamente baja. Esto pone de manifiesto las limitaciones del enfoque booleano para gestionar consultas complejas, especialmente en contextos donde el ranking de relevancia es fundamental. Por el contrario, el modelo TF-IDF ha demostrado una mayor capacidad para ordenar correctamente los documentos relevantes, lo cual es esencial en escenarios de búsqueda donde el usuario espera resultados relevantes en las primeras posiciones.

No obstante, los resultados también evidencian la necesidad de considerar tanto la precisión como la sensibilidad para evaluar adecuadamente la efectividad de un sistema de recuperación. Una alta sensibilidad sin una precisión proporcional puede llevar a una sobrecarga de información irrelevante, disminuyendo la utilidad real del sistema para el usuario final.

Como posibles mejoras, se propone la incorporación de técnicas más avanzadas de procesamiento del lenguaje natural (NLP), con el objetivo de interpretar mejor el contexto de búsqueda. Además, la ampliación del corpus para incluir una mayor diversidad de jugadores, equipos y competiciones permitiría realizar evaluaciones más representativas y generalizables.

VIII. USO DE INTELIGENCIA ARTIFICIAL GENERATIVA

En el presente proyecto se ha hecho uso de sistemas de inteligencia artificial generativa con fines de apoyo a la programación y análisis, cumpliendo con las condiciones establecidas por la asignatura:

- Se utilizó la extensión **GitHub Copilot** en Visual Studio Code para autocompletar y sugerir fragmentos de código.
- Se realizaron consultas al modelo **ChatGPT** para:
  - Refactorizar el código en Python para hacerlo más modular y legible.
  - Aprender a manejar archivos y carpetas utilizando la librería `os`.
  - Generar la base de varios pseudocódigos de la memoria.

- Clasificar a los jugadores del corpus según diferentes necesidades de información.
- Obtener un ejemplo funcional para extraer fichas de jugadores utilizando la API de Wikipedia.
- Desarrollar un ejemplo inicial de cómo estructurar una interfaz textual para realizar consultas.

A continuación, se detallan algunas de las entradas (*prompts*) proporcionadas a ChatGPT, junto con el contexto de uso:

- **Prompt 1:** *"¿Cómo puedo obtener fichas de jugadores de fútbol a través de Wikipedia?"*
  Uso: Comprender el uso de la API de Wikipedia para construir el corpus documental del proyecto.
- **Prompt 2:** *"¿Qué cosas podría refactorizar de este archivo? (archivo adjuntado)"*
  Uso: Simplificación y mejora del código Python, eliminando redundancias y mejorando la estructura modular.
- **Prompt 3:** *"Tengo estas funciones de normalización de texto (`normalizar_texto_con_stem` y `normalizar_texto_sin_stem`). ¿Cómo puedo normalizar todas las fichas de mi corpus y guardarlas en una carpeta?"*
  Uso: Aplicación de técnicas de procesamiento de texto y gestión de archivos con la librería `os`.
- **Prompt 4:** *"Dadas estas necesidades de información (mis necesidades de información), ¿puedes clasificarme estos jugadores (jugadores del corpus)?"*
  Uso: Identificación y agrupación de documentos relevantes según las necesidades planteadas.
- **Prompt 5:** *"Dada esta función (función), ¿cómo la pondrías en pseudocódigo?"*
  Uso: Generar una base clara para los pseudocódigos requeridos en la documentación del proyecto.
- **Prompt 6:** *"En un proyecto se me pide tener una interfaz textual para interactuar con sistemas de recuperación de información. Quiero un ejemplo de cómo construir esta interfaz para elegir el tipo de normalización antes de la creación de los índices sobre los que realizaré consultas."*
  Uso: Obtener una plantilla funcional para diseñar una interfaz textual interactiva.

REFERENCIAS

[1] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, An Introduction to Information Retrieval. Cambridge University Press: Cambridge, England, 2009, capítulo 1 pp. 38–55, capítulo 2.2 pp. 59–73, capítulos 6.2, 6.3 pp. 154–163, capítulos 8.2, 8.3 pp. 190–195

[2] Wikipedia

[3] Librería Natural Language Toolkit — NLTK. Documentación oficial. https://www.nltk.org/.

[4] Librería Whoosh. Documentación oficial. https://whoosh.readthedocs.io/en/latest/.

[5] Librería Scikit-learn. https://scikit-learn.org/.

[6] Python Software Foundation. Librería estándar os. https://docs.python.org/3/library/os.html.