Gianella Belén Nardi

diciembre 2024

Trabajo Práctico 2

PROCESAMIENTO DEL LENGUAJE NATURAL

TECNICATURA UNIVERSITARIA
EN INTELIGENCIA ARTIFICIAL

UNR Universidad Nacional de Rosario

FCEIA
FACULTAD DE CIENCIAS EXACTAS, INGENIERIA Y AGRIMENSURA

Índice:

Introducción ……. 2/20

Metodologías ……. 3/20

Ejercicio 1: resumen ……. 4/20

Ejercicio 1: desarrollo …… 5-15/20

Ejercicio 1: conclusiones …… 16, 17/20

Ejercicio 2: conclusiones ……. 18/20

Enlaces a modelos y librerías utilizadas…….. 19/20

1 INTRODUCCIÓN:

Este trabajo presenta la implementación de un chatbot experto en el juego de mesa 'The White Castle', desarrollado como parte del Trabajo Práctico Final de la materia Procesamiento del Lenguaje Natural. The White Castle es un juego de estilo Eurogame que desafía a los jugadores a construir y gestionar su propio castillo medieval, combinando elementos de gestión de recursos y planificación estratégica.

El objetivo principal fue desarrollar un sistema capaz de mantener conversaciones en idioma inglés coherentes e informativas con usuarios sobre las reglas, estrategias y mecánicas del juego, basándose en una variedad de fuentes de conocimiento estructuradas y no estructuradas.

La primera implementación se basa en la técnica RAG (Retrieval Augmented Generation), que combina modelos de lenguaje de gran tamaño con una base de conocimiento construida a partir de tres tipos principales de fuentes:

- Documentos de texto que incluyen reglas oficiales, guías estratégicas y reseñas del juego.
- Datos tabulares que contienen información numérica, puntuaciones y estadísticas del juego.
- Una base de datos de grafos que modela las relaciones entre los diferentes creadores y distribuidores del juego.

El sistema implementa dos aproximaciones diferentes para la clasificación de consultas: un modelo entrenado con ejemplos y embeddings y otra basada en LLM, permitiendo comparar su efectividad en la tarea de direccionar las consultas a las fuentes de información más apropiadas. Además, se utiliza una búsqueda híbrida que combina similitud semántica y comparación por palabras clave para optimizar la recuperación de información relevante.

Este informe documenta el proceso completo de desarrollo, desde la recopilación y preparación de los datos hasta la evaluación del sistema, detallando:

- La metodología de procesamiento.
- La implementación de consultas dinámicas para la recuperación eficiente de información.
- Las estrategias de clasificación de consultas y su evaluación comparativa.
- Los desafíos técnicos encontrados y las soluciones implementadas.
- Los resultados obtenidos y su análisis.

La segunda implementación se basa en la primera e incorpora el concepto de Agente.

Finalmente, se discuten las limitaciones actuales del sistema y se proponen posibles mejoras para expandir sus capacidades.

2 METODOLOGÍAS:

El desarrollo del proyecto se realizó íntegramente en Google Colab, un entorno de notebook basado en la nube que permite la ejecución de código Python y proporciona acceso a recursos de computación incluyendo GPUs. Para establecer el entorno de trabajo, fue necesario instalar y configurar múltiples dependencias, que se pueden categorizar en los siguientes grupos:

- Herramientas de Web Scraping y Procesamiento de Documentos.
- Bases de Datos y Procesamiento de Vectores.
- Procesamiento de Lenguaje Natural y Modelos.

Para la recopilación automatizada de datos, se implementó un sistema de web scraping utilizando Selenium WebDriver con Chrome en modo headless.

Además, se conecta con Google Drive para la carga y descarga de los archivos que se van generando.

Para ejecutar correctamente el código presentado en este proyecto, es necesario proporcionar una API key válida de Hugging Face en cada sección donde se requiera. Estas claves son credenciales confidenciales y personales, esenciales para autenticar el acceso a los servicios de Hugging Face, como modelos de lenguaje, bases de datos u otras herramientas. Por razones de seguridad y privacidad, las API keys no se incluyen directamente en el código compartido. Por favor, genera tu propia API key desde tu cuenta de Hugging Face.

3 EJERCICIO 1:

3.1 RESUMEN:

Este ejercicio presenta el desarrollo e implementación de dos clasificadores de consultas para un sistema de chatbot especializado en el juego de mesa "The White Castle". El sistema, basado en la arquitectura RAG (Retrieval Augmented Generation), utiliza tres bases de datos diferentes para almacenar y recuperar información: una base de datos vectorial para información textual y descriptiva, una base de datos tabular para datos numéricos y estadísticos, y una base de datos de grafos para información relacional.

Se implementaron y compararon dos aproximaciones distintas para la clasificación de consultas: un clasificador supervisado que combina embeddings de SentenceTransformer con regresión logística y un clasificador basado en un modelo de lenguaje grande (LLM) utilizando 'microsoft/Phi-3-mini-4k-instruct'. Para la recuperación de información, se desarrolló un sistema de búsqueda híbrida que integra similitud semántica (embeddings) y coincidencia de palabras clave (BM25), complementado con un sistema de re-ranking para optimizar la relevancia de los resultados.

El proyecto incluyó la recopilación automatizada de datos mediante web scraping, el procesamiento y estructuración de la información en las diferentes bases de datos, y la implementación de consultas dinámicas para acceder eficientemente a cada fuente de datos. Los resultados demuestran la viabilidad de ambos enfoques de clasificación, destacando las ventajas y limitaciones de cada aproximación.

3.2 DESARROLLO:

3.2.1 CREACIÓN DE FUNCIONES PRINCIPALES

Se implementaron funciones fundamentales para la extracción, procesamiento y manipulación de datos:

download_file

Facilita la obtención de recursos almacenados en Google Drive, manteniendo el proyecto portable y reproducible en cualquier instancia de Google Colab. Descarga archivos desde Google Drive con el id del mismo y una ruta para guardar el mismo. Retorna el contenido del archivo.

clean_text

Realiza las siguientes operaciones: conversión a minúsculas, eliminación de espacios múltiples, filtrado por longitud mínima y detección y filtrado de textos en alemán.

Limitaciones identificadas:

- la detección de idioma basada en caracteres específicos es poco robusta.
- No se implementa traducción automática.
- Posible pérdida de información valiosa en otros idiomas.

Mejoras propuestas:

- Implementar detección de idioma usando bibliotecas especializadas (langdetect, spacy)
- Añadir traducción automática para textos relevantes.
- Considerar el uso de modelos multilingües.

split_chunks

Fragmenta un texto en chunks. Tiene predefinidos los parámetros de tamaño de los chunks (500) y superposición entre los mismos (40). Considero que este proceso puede ser mejorado.

generate_embeddings

Genera embeddings para cada chunk del texto. Retorna una lista de embeddings. El modelo elegido es de SentenceTransformer, 'all-MiniLM-L6-v2'. Ventajas consideradas en la elección del modelo:

- Eficiencia computacional: modelo ligero que mantiene un buen rendimiento.
- Versatilidad multilingüe: capacidad de procesar texto en múltiples idiomas.
- Dimensionalidad reducida: vectores de 384 dimensiones que permiten búsquedas eficientes.
- Relación rendimiento/recursos: equilibrio entre calidad de embeddings y requisitos computacionales.

3.2.2 EXTRACCIÓN DE INFORMACIÓN PARA LAS BASES DE DATOS

Se implementó un sistema de web scraping para recopilar información de múltiples fuentes especializadas. Fuentes principales:

- Board Game Geek (BGG): Foros y reseñas oficiales.
- The Opinionated Gamers: https://opinionatedgamers.com/2023/09/25/dale-yu-review-of-the-white-castle/
- Gaming Trend: https://gamingtrend.com/feature/reviews/the-white-castle-review-a-tiny-box-with-a-ton-of-game/
- Meeple Mpuntain: https://www.meeplemountain.com/reviews/the-white-castle/
- Punchboard: https://www.punchboard.co.uk/the-white-castle-review/

La extracción de información mediante Web Scrapping varía según la inspección de cada página web, por lo cual se realizó de forma manual.

En la página Board Game Geek se accedió a la sección 'Forums', foro de donde se extrajeron preguntas y respuestas de usuarios del juego. Tener en cuenta que este proceso puede demorarse entre 20 y 30 minutos en realizarse, aproximadamente.

La información extraída (en crudo) se consolidó en un archivo llamado 'parrafos.txt'. Luego se extrae y se lee el Documento y se concatena como texto para facilitar su manipulación.

3.2.3 FRAGMENTACIÓN DE TEXTOS

Se implementó un sistema de fragmentación basado en 'RecursiveCharacterTextSplitter'. Los parámetros tamaños de fragmento y superposición fueron definidos manualmente. Oportunidades de mejora identificadas: optimizar parámetros de fragmentación según el tipo de contenido, añadir metadatos a los chunks para mejorar la recuperación.

3.2.4 CREACIÓN Y ACCESO A LAS BASES DE DATOS

Base de Datos Vectorial:

Se crea el cliente y la colección en ChromaDB en caso de que la misma no exista. En la sección de documentos se agrega cada chunk, en metadatos el ID y una breve descripción del mismo y se agregan los embeddings (representación numérica del texto) obtenidos mediante la función 'generate_embeddings'. Esto permite realizar búsquedas semánticas eficientes basadas en la similitud entre los embeddings.

```
1386:
  Contenido: don't really take into account if the bot can score bonus points wit
  Metadatos: {'description': 'Chunk 1386', 'id': 'doc_1385', 'source': 'unknown'}
  Embedding: [-0.03466159  0.0290182  -0.07539575  0.01196197 -0.00144152]
1387:
  Contenido: the background? thank you so much for this variant; i can't wait to
  Metadatos: {'description': 'Chunk 1387', 'id': 'doc_1386', 'source': 'unknown'}
  Embedding: [-0.10140175  0.01976737 -0.02337748 -0.03087726  0.00249805]
1388:
  Contenido: that's what unbalances the bot: the fact that at the end of the roun
  Metadatos: {'description': 'Chunk 1388', 'id': 'doc_1387', 'source': 'unknown'}
  Embedding: [-0.07300729  0.05269133 -0.04669584 -0.01250752  0.04018908]
1389:
  Contenido: i, therefore, thought of introducing this simple rule: the bot pays
  Metadatos: {'description': 'Chunk 1389', 'id': 'doc_1388', 'source': 'unknown'}
  Embedding: [-0.06529893  0.06605622 -0.03218251 -0.05629517 -0.06586684]
1390:
  Contenido: turn order), but this way, i found the cash flow to be more balanced
  Metadatos: {'description': 'Chunk 1390', 'id': 'doc_1389', 'source': 'unknown'}
  Embedding: [-0.04378618  0.04714585 -0.00971359 -0.04725428 -0.03597262]
1391:
  Contenido: meaning, in order: - a position in the card zone with 2 tiles matchi
  Metadatos: {'description': 'Chunk 1391', 'id': 'doc_1390', 'source': 'unknown'}
  Embedding: [ 0.00030404  0.06107379 -0.03692844 -0.04095976  0.00143273]
```

(estructura de la base de datos vectorial)

Base de Datos Tabular

	A	B	C
1	Category	Data	
2	Game title	The White Castle	
3	Number of playe	1–4 Players	
4	Recommended r	Age: 12+	
5	Avg. Rating	7.98	
6	No. of Ratings	11,879	
7	Std. Deviation	1.18	
8	Weight	3.02 / 5	
9	Comments	1,630	
10	Fans	1,863	
11	Page Views	1,382,795	
12	Overall Rank	109 Historical Rank	
13	Strategy Rank	87 Historical Rank	
14	All Time Plays	53,730	
15	This Month	2,670	
16	Own	22,884	
17	Prev. Owned	725	
18	For Trade	79 Find For-Trade Matches	
19	Want In Trade	656 Find Want-in-Trade Matches	
20	Wishlist	5,527	
21	Has Parts	5	
22	Want Parts	3	

Anteriormente, se extrajo el número de jugadores y la edad mínima recomendada en la página principal de ‘The White Castle’ en Board Game Geek.

Para la mejoría de la Base de Datos y un mejor funcionamiento de la misma, se decidió que sería de utilidad agregarle información contenida en la sección de ‘stats’ de Board Game Geek, que contiene información estadística del juego. Esta información se añadió a la lista creada anteriormente.

La lista contiene la información almacenada como texto, separando la descripción de la columna del contenido numérico por una coma. Es importante reconocer esto para luego descargar el archivo como CSV correctamente.

Mejoras propuestas: limpieza de los datos previa a la creación del archivo CSV para que en la columna ‘Data’ únicamente contenga información numérica (de tipo ‘int’).

Base de Datos de Grafos

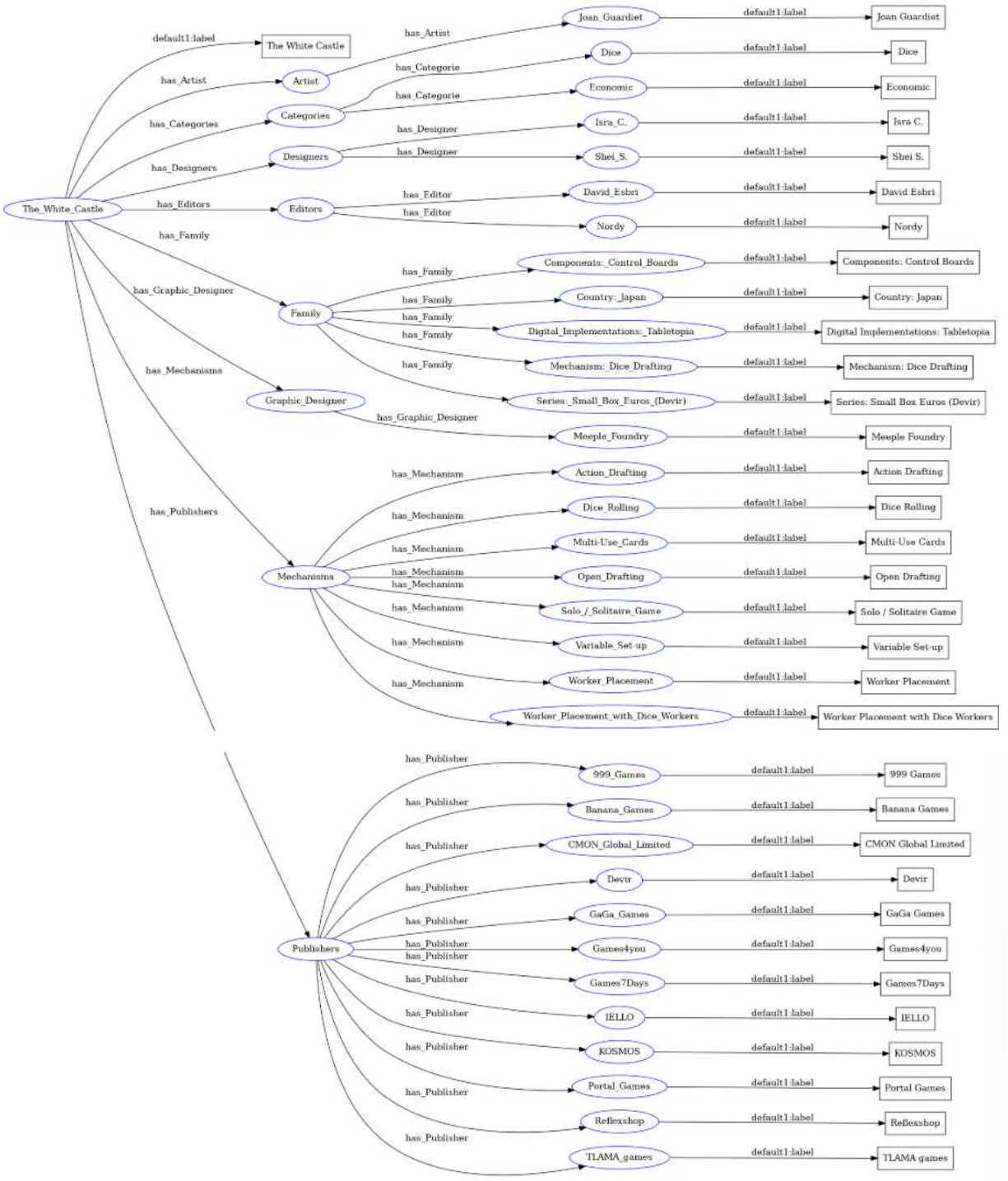
Se extrae información de la sección ‘credits’ (créditos) de Board Game Geek.

Estructura del grafo:

1. Nodo central: "The White Castle"
2. Nodos relacionados:

- Artistas
- Categorías
- Diseñadores
- Editoras
- Familia
- Diseñadores gráficos
- Mecanismos
- Editores

El grafo se guarda en formato Turtle y se visualiza con ‘Linked Data Finland’.



(grafo visualizado desde Linked Data Finland)

3.2.5 BÚSQUEDA HÍBRIDA Y RE-RANK

Modelo BM25 para la búsqueda por palabras clave

Se utiliza la clase ‘BM25Searcher’ proporcionada por la cátedra. La implementación requiere de una lista de documentos como strings (que luego serán pre-procesados) y la elección de un idioma para el tokenizador. La búsqueda devuelve resultados, que contiene el texto y el puntaje asignado al mismo. Se decidió que una buena práctica era traer los 10 mejores resultados obtenidos con la búsqueda.

```
Resultados de búsqueda BM25:
Score: 13.0437
Texto: dice, reroll them and place them on the bridges to start the next round. if y
Score: 11.6940
Texto: that i usually really appreciate in a game: - only after final scoring we knc
Score: 11.0772
Texto: and into the castle. the light action they chose gave them 2 money and trigge
Score: 10.7436
Texto: take the role of a clan leader in japan. throughout the game, you are going t
Score: 10.3113
Texto: who keeps doing their job even after clocking out. courtiers: climb the socia
Score: 10.2626
Texto: i copied the jpgs listed on thingiverse and put them on my phone in an album
```

(resultados a la pregunta ‘who wins in The White Castle?’)

Modelo de búsqueda por Embeddings con Sentence Transformer

Se obtiene de la colección creada en Chroma los documentos, metadatos y embeddings.

Se define una función ‘calculate_query_embedding’ que calcula el embedding para la query ingresada por el usuario y lo retorna. El modelo utilizado es ‘all-MiniLM-L6-v2’.

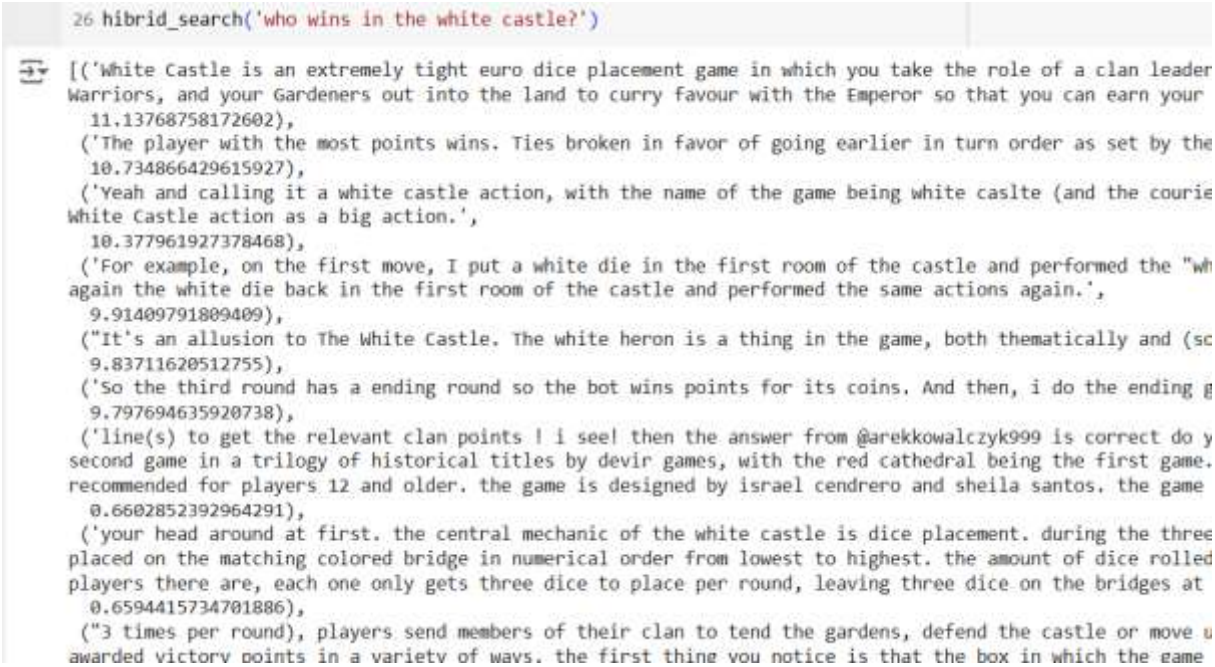
Finalmente, se crea una función llamada ‘find_most_similar_embeddings’ que devuelve una cantidad predefinida (15) de embeddings similares a la query del usuario. La similitud fue calculada por similitud del coseno. La función, además, necesita los parámetros ‘embeddings’ (embeddings de los chunks), ‘documents’ (chunks) y ‘metadatas’ (metadatos de los chunks).

```
Documentos más similares:
Documento: the white castle is a euro-like game with resource management mechanisms,
Metadatos: {'description': 'Chunk 1170', 'id': 'doc_1169', 'source': 'unknown'}
Similitud: 0.7138
Documento: is correct do you mean my answer ? … because now i am confused. the white
Metadatos: {'description': 'Chunk 87', 'id': 'doc_86', 'source': 'unknown'}
Similitud: 0.6326
Documento: white castle is a balancing act. every decision feels monumental, and the
Metadatos: {'description': 'Chunk 54', 'id': 'doc_53', 'source': 'unknown'}
Similitud: 0.6074
Documento: the central mechanic of the white castle is dice placement. during the th
Metadatos: {'description': 'Chunk 27', 'id': 'doc_26', 'source': 'unknown'}
Similitud: 0.6074
Documento: them and place them on the bridges to start the next round. if you are at
Metadatos: {'description': 'Chunk 10', 'id': 'doc_9', 'source': 'unknown'}
Similitud: 0.6071
Documento: is a lot to try and do, and little time to do all of those things, and nc
Metadatos: {'description': 'Chunk 944', 'id': 'doc_943', 'source': 'unknown'}
Similitud: 0.5946
Documento: is clearly erroneous. and has been recreated here without some of the flu
Metadatos: {'description': 'Chunk 938', 'id': 'doc_937', 'source': 'unknown'}
Similitud: 0.5940
```

(resultados a la pregunta ‘who wins in The White Castle?’)

Búsqueda híbrida y Re-Rank

Se crea una función llamada ‘hibrid_search’ que tiene un funcionamiento básico: extrae los resultados de la búsqueda por palabras clave con BM25 y la búsqueda semántica por Embeddings y los retorna junto con los puntajes de cada uno.



```
26 hibrid_search('who wins in the white castle?')

[('White Castle is an extremely tight euro dice placement game in which you take the role of a clan leader
Warriors, and your Gardeners out into the land to curry favour with the Emperor so that you can earn your
  11.13768758172602),
 ('The player with the most points wins. Ties broken in favor of going earlier in turn order as set by the
  10.734866429615927),
 ('Yeah and calling it a white castle action, with the name of the game being white caslte (and the courie
White Castle action as a big action.',
  10.377961927378468),
 ('For example, on the first move, I put a white die in the first room of the castle and performed the "wh
again the white die back in the first room of the castle and performed the same actions again.',
  9.91409791809409),
 ("It's an allusion to The White Castle. The white heron is a thing in the game, both thematically and (sc
  9.83711620512755),
 ('So the third round has a ending round so the bot wins points for its coins. And then, i do the ending g
  9.797694635920738),
 ('line(s) to get the relevant clan points ! i see! then the answer from @arekkowalczyk999 is correct do y
second game in a trilogy of historical titles by devir games, with the red cathedral being the first game.
recommended for players 12 and older. the game is designed by israel cendrero and sheila santos. the game
  0.6602852392964291),
 ('your head around at first. the central mechanic of the white castle is dice placement. during the three
placed on the matching colored bridge in numerical order from lowest to highest. the amount of dice rolled
players there are, each one only gets three dice to place per round, leaving three dice on the bridges at
  0.6594415734701886),
 ("3 times per round), players send members of their clan to tend the gardens, defend the castle or move u
awarded victory points in a variety of ways. the first thing you notice is that the box in which the game
```

Luego, se define la clase ‘Reranker’, que realiza el ReRanking con un modelo cross-encoder. La función ‘rerank’ dentro de la misma recibe como parámetros la query, nodos y el número de devoluciones que queremos. Devuelve una lista de nodos con las respuestas con mejor puntaje. El puntaje se calcula comparando cada resultado con la query ingresada.

Para poder implementar el Reranker, se definió una clase Nodo que es básica pero necesaria para el correcto funcionamiento de la clase.

Finalmente, se crea una función que aplica el Re-Ranking y devuelve el contexto generado como string.

```
Contexto generado:
'the gameplays is captivating. it's not excessively different from other euros, but
still, there's no other game like the white castle - i don't know how else to descr
ibe it. moreover, there are two things that i usually really appreciate in a game:
- only after final scoring we know who won the game. if such is the case, a game al
ways get an extra point: 8/10. published on 13-10-2023, with a rating of 8/10 stars
https://www.bordspelwereld.nl/en/en-reviews/the-white-castle the white castle fits
\nthem and place them on the bridges to start the next round. if you are at the end
of the third round, the game ends. after setting the turn order, move to final scor
ing. in addition to the points earned through the run of play, score: the player wi
th the most points wins. ties broken in favor of going earlier in turn order as set
by the final round. the white castle is a very tight game where every action has a
large impact on your overall performance. you essentially only get 9 turns each
\n'
```

3.2.6 QUERYS DINÁMICAS PARA ACCEDER A LAS BASES DE DATOS

Para acceder a la Base de Datos Vectorial:

La función return_context devuelve un contexto generado a partir de la búsqueda híbrida. Para trabajar los resultados de la búsqueda híbrida, se interpretan como Nodos. A esos resultados se le aplica el re-ranking y se construye y retorna el contexto como string.

Para acceder a la Base de Datos Tabular:

Se extrae y compara el contenido del prompt con las posibles categorías de datos del CSV. Si coinciden, se extrae el valor del dato numérico contenido en el archivo.

Para acceder a la Base de Datos de Grafos:

Al igual que en la Base de Datos mencionada anteriormente, se compara el prompt con las posibles categorías del grafo. Si coincide, se realizan consultas a la Base de Datos del Grafo de tipo SPARQL. Se retorna el resultado como un string.

3.2.7 CLASIFICADOR BASADO EN MODELO ENTRENADO CON EJEMPLOS Y EMBEDDINGS

Se implementó un modelo de clasificación supervisado para determinar automáticamente qué fuente de datos es más apropiada para responder cada consulta del usuario. Este modelo utiliza embeddings y regresión logística para categorizar las preguntas en tres clases: grafo, tabular, o vectorial.

Los datos de entrenamiento fueron solicitados a ChatGPT, dándole un contexto de qué contenía cada base de datos y cuál era el objetivo de esos datos.

El modelo SentenceTransformer se eligió por su capacidad para capturar la semántica de las preguntas en vectores densos de 384 dimensiones. Se eligió la Regresión Logística por:

- Simplicidad y eficiencia computacional
- Buena interpretabilidad
- Capacidad de manejar múltiples clases
- Probabilidades bien calibradas

Se evaluó el modelo y se obtuvieron las siguientes métricas:

```
Reporte de clasificación:
              precision    recall  f1-score   support

         csv       1.00      0.88      0.93         8
       graph       1.00      1.00      1.00         6
   vectorial       0.80      1.00      0.89         4

    accuracy                           0.94        18
   macro avg       0.93      0.96      0.94        18
weighted avg       0.96      0.94      0.95        18

Precisión del modelo: 0.94
```

(reporte de clasificación del modelo basado en embeddings)

El modelo basado en embeddings obtuvo excelentes resultados, con una precisión global de 0.94 con un desempeño robusto y consistente. En particular:

- csv: Alta precisión (1.00) y un recall sólido (0.88), lo que demuestra que identifica correctamente la mayoría de las instancias, aunque pierde algunas.
- graph: Perfecto en todas las métricas (1.00), indicando que el modelo es altamente confiable en esta categoría.
- vectorial: A pesar de tener una precisión algo menor (0.80), logra un recall perfecto (1.00), lo que sugiere que clasifica correctamente todas las instancias, aunque incluye algunos falsos positivos.

3.2.8 CLASIFICADOR BASADO EN LLM

Para implementar el sistema de clasificación de consultas basado en LLM, se desarrolló una solución que utiliza procesamiento de lenguaje natural mediante el modelo 'microsoft/Phi-3-mini-4k-instruct', accedido a través de la API de Hugging Face. Esta elección se fundamenta en la capacidad del modelo para comprender y clasificar texto en lenguaje natural de manera eficiente y precisa.

La implementación utiliza un enfoque de promp, donde se proporciona al modelo instrucciones específicas sobre los criterios de clasificación para cada tipo de base de datos. Este método permite que el modelo tome decisiones informadas sobre qué sistema de almacenamiento es más apropiado para cada consulta, basándose en el contenido y la naturaleza de la pregunta.

```
Reporte de clasificación del modelo:
                    precision    recall  f1-score   support

               csv       1.00      0.62      0.77         8
database_not_found       0.00      0.00      0.00         0
             graph       0.67      0.67      0.67         6
         vectorial       0.60      0.75      0.67         4

          accuracy                           0.67        18
         macro avg       0.57      0.51      0.53        18
      weighted avg       0.80      0.67      0.71        18

Precisión del modelo: 0.67
```

(reporte de clasificación del modelo basado en LLM)

El modelo basado en LLM presenta un desempeño mucho más inconsistente, con una precisión global de 0.67. Sus métricas revelan debilidades importantes:

- csv: Si bien la precisión es alta (1.00), el recall es bajo (0.62), lo que significa que identifica correctamente solo algunas instancias y omite otras.
- graph y vectorial: Ambos tienen métricas aceptables, pero no sobresalientes, con valores de precisión y recall en el rango de 0.60 a 0.67.

3.3 CONCLUSIONES:

La implementación y evaluación de los sistemas de clasificación de consultas ha proporcionado valiosas perspectivas sobre el procesamiento de consultas en sistemas de chatbot especializados. Las principales conclusiones son:

Efectividad de los Clasificadores:

- El clasificador basado en LLM demostró una gran flexibilidad y capacidad de comprensión contextual, aunque con mayor costo computacional.
- El clasificador basado en embeddings y regresión logística mostró un rendimiento eficiente y resultados consistentes, con la ventaja de ser más ligero y rápido en la ejecución.

Se concluye que dado el contexto en el que nos encontramos con el trabajo, y teniendo en cuenta que las consultas van a ser relacionadas a un juego de mesa específico y solo hay 3 Bases de Datos a las cuales dirigirla, el modelo de preferencia es el basado en embeddings. Además, la relación costo computacional con cantidad de ejemplos requeridos para un funcionamiento razonable del mismo, es de alta conveniencia.

El modelo ‘microsoft/Phi-3-mini-4k-instruct’ ha sido ajustado para seguir instrucciones de forma precisa, lo que lo hace excelente para aplicaciones conversacionales donde las respuestas deben alinearse con instrucciones específicas del usuario. Es capaz de manejar diferentes estilos de pregunta-respuesta, generación de contenido y asistencia basada en instrucciones. A pesar de ser un modelo compacto ("mini"), está diseñado para ofrecer un equilibrio entre rendimiento y eficiencia computacional.

Sistema de Recuperación:

- La búsqueda híbrida, combinando BM25 y embeddings, demostró ser más robusta que cualquiera de los métodos por separado.
- El sistema de re-ranking mejoró significativamente la relevancia de los resultados recuperados.

Arquitectura de Bases de Datos:

- La división de la información en tres tipos de bases de datos especializadas permitió una gestión más eficiente según la naturaleza de los datos.
- Las consultas dinámicas facilitaron el acceso específico a cada tipo de información.

Limitaciones Identificadas:

- La detección de idioma en la limpieza de textos requiere mejoras para mayor robustez.
- Los parámetros de fragmentación de textos podrían optimizarse según el tipo de contenido.
- La base de datos tabular necesita una mejor estructuración de datos numéricos.

Mejoras Propuestas:

- Implementar sistemas más robustos de detección y traducción de idiomas.
- Optimizar los parámetros de fragmentación mediante análisis automático (o no) del contenido.
- Desarrollar un sistema de caché para consultas frecuentes.
- Incorporar un sistema de retroalimentación para mejorar la precisión de las clasificaciones.

Este trabajo sienta las bases para futuras investigaciones en sistemas de clasificación de consultas y recuperación de información en chatbots especializados, destacando la importancia de combinar diferentes técnicas y aproximaciones para obtener resultados óptimos.

4 EJERCICIO 2:

4.1 RESUMEN:

El código define un agente conversacional basado en el paradigma ReAct (Reasoning and Acting), utilizando un modelo de lenguaje (LLM) y herramientas especializadas para responder consultas relacionadas con juegos de mesa.

El modelo de lenguaje 'llama3.2:latest' está diseñado para manejar tareas de procesamiento de lenguaje natural (NLP) con eficacia, lo que es clave para un agente que debe interpretar consultas y realizar razonamientos basados en herramientas. La configuración de baja temperatura (0.1) favorece respuestas consistentes y predecibles, necesarias para garantizar que el agente siga un formato específico como ReAct. El tiempo de respuesta fue aumentado lo suficiente para que el modelo pueda procesar el prompt y generar la respuesta.

En el caso práctico evaluado, el agente identificó correctamente la herramienta adecuada para obtener la información solicitada, ajustando sus acciones ante errores y consolidando respuestas claras y precisas. Esto evidencia la robustez del diseño del agente y su capacidad para manejar errores de manera autónoma.

4.2 ENLACES A MODELOS Y LIBRERÍAS UTILIZADAS:

Lista de herramientas empleadas junto con los enlaces a sus respectivas páginas oficiales:

- Selenium: https://www.selenium.dev/documentation/
- BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/css-selector-update-doc/
- pandas: https://pandas.pydata.org/docs/
- docx: https://python-docx.readthedocs.io/en/latest/user/documents.html
- webdriver_manager: https://github.com/bonigarcia/webdrivermanager
- chromadb: https://pypi.org/project/chromadb/
- rapidfuzz: https://pypi.org/project/RapidFuzz/
- langchain: https://langchain-doc.readthedocs.io/en/latest/index.html
- gdown: https://pypi.org/project/gdown/
- spaCy: https://spacy.io/
- SentenceTransformers: https://huggingface.co/sentence-transformers
- NLTK: https://www.nltk.org/