Implementación

Recolección y preprocesamiento de datos

El primer paso en la metodología implica la recolección y el preprocesamiento de datos financieros de diversas fuentes para construir una base de conocimiento completa. Estos datos abarcan varios conceptos financieros, productos, regulaciones y tendencias del mercado relevantes para el dominio del asesoramiento. Actualmente existen varios conjuntos de datos disponibles para datos financieros, como informes estructurados, fuentes de datos de mercado y documentos regulatorios. Sin embargo, para este caso de uso particular, no nos centraremos en dichos datos, ya que pueden causar ruido entre la información relevante requerida para el conjunto de datos. En su lugar, nos centraremos en la creación de un conjunto de datos personalizado que se adapte al dominio del asesoramiento financiero.

Para lograr ese objetivo, nuestra herramienta principal es el web scraping, que nos permite extraer información de sitios web financieros, blogs, foros y organismos reguladores. En este caso particular, nos centraremos en la extracción de información de blogs financieros, ya que proporcionan una rica fuente de contenido actualizado y relevante, con especial interés en el sitio web https://www.investopedia.com/. Este sitio web es una plataforma de educación financiera muy conocida que cubre una amplia gama de temas, incluyendo inversiones, comercio, finanzas personales y economía. Al extraer contenido de Investopedia, podemos recopilar un conjunto diverso de conceptos financieros, definiciones y explicaciones que suelen buscar los usuarios que buscan asesoramiento financiero.

Justificación: El web scraping de Investopedia garantiza que los datos sean actuales, relevantes y completos, lo cual es crucial para proporcionar un asesoramiento financiero preciso. Este enfoque también permite la creación de un conjunto de datos personalizado que se adapte específicamente a las necesidades del dominio del asesoramiento financiero.

Los datos extraídos se preprocesan para eliminar contenido irrelevante, estandarizar formatos y anotar entidades y relaciones. Este paso de preprocesamiento implica técnicas de procesamiento del lenguaje natural como la tokenización, la lematización y el reconocimiento de entidades para garantizar que los datos estén estructurados y semánticamente enriquecidos. Para este proceso, utilizaremos NLTK, que es una biblioteca de Python de PNL de uso general. El objetivo principal de este paso es crear un conjunto de datos limpio y estructurado que pueda utilizarse eficazmente para la construcción de grafos y la recuperación de información.

Justificación: El uso de NLTK para el preprocesamiento garantiza que los datos estén limpios, estructurados y semánticamente enriquecidos, lo cual es esencial para la construcción precisa de grafos y la recuperación de información.

Construcción del grafo

El núcleo de la metodología reside en la construcción de una representación del conocimiento basada en grafos que capture las intrincadas relaciones entre los conceptos financieros. Este grafo se construye utilizando los datos preprocesados, donde cada concepto financiero se representa como un nodo y las relaciones entre conceptos se representan como aristas. La estructura del grafo permite un recorrido eficiente, una recuperación sensible al contexto y una comprensión semántica del dominio financiero.

Este paso no se realizó manualmente, sino utilizando una herramienta muy útil llamada Langchain, en particular LangGraph, que es una colección de herramientas para construir y consultar grafos de conocimiento a partir de datos de texto. LangGraph utiliza modelos de procesamiento del lenguaje natural de última generación para extraer entidades, relaciones y jerarquías de texto no estructurado, y luego construye una base de datos de grafos que puede consultarse para la recuperación de

información. Al aprovechar LangGraph, podemos automatizar el proceso de construcción del grafo y garantizar la escalabilidad y la precisión del grafo de conocimiento resultante. Cabe destacar que Langchain es un conjunto de herramientas que funciona con modelos LLM de terceros, pero no los proporciona por defecto, lo que significa que el usuario debe proporcionar claves API para los modelos que desea utilizar. Personalmente, utilicé el modelo GPT-4o-mini, que es un modelo de propósito general y relativamente rápido creado por OpenAI para esta tarea.

Justificación: Automatizar el proceso de construcción del grafo con Langchain garantiza la escalabilidad y la precisión, mientras que el uso del modelo GPT-4o-mini proporciona una extracción de entidades y una asignación de relaciones eficiente y precisa.

La forma en que funciona es:

Proporcionar un conjunto limpio de documentos (en este caso, artículos de blogs financieros de Investopedia después del preprocesamiento).

Pasar esos documentos a través del pipeline de LangGraph, utilizando un enfoque de Herramienta o de Prompt. El enfoque de herramienta utiliza el enfoque de salida estructurada subyacente del modelo, haciendo que la respuesta se produzca en formato json con los nodos y aristas del grafo. El enfoque de prompt es semánticamente el mismo, pero a veces difiere en la salida, en lugar de utilizar la salida estructurada, utiliza un prompt para modelar la respuesta del sistema, un prompt que puede ser personalizado por el usuario.

Después de ese procesamiento, el sistema recopila todas las entidades encontradas en todos los documentos, así como las entidades entre ellas. Esto es particularmente útil, ya que permite al sistema crear relaciones entre entidades que no están directamente relacionadas en el mismo documento, y también utiliza el poder de un modelo de lenguaje para crear un grafo más preciso y completo.

Luego podemos construir el grafo utilizando una base de datos de grafos, en este caso utilizaremos Neo4j, que es una base de datos de grafos popular que permite el almacenamiento, la consulta y la visualización eficientes de datos de grafos. Las entidades se almacenan como nodos en el grafo, y las relaciones se almacenan como aristas, formando una representación conectada y estructurada de la base de conocimiento financiero.

Justificación: El uso de Neo4j para el almacenamiento y la consulta de grafos garantiza una gestión eficiente y escalable del grafo de conocimiento, mientras que el enfoque de salida estructurada de LangGraph garantiza una construcción precisa y completa del grafo.

El grafo resultante es un grafo textual, lo que significa que las relaciones y los nodos son texto puro, no se les añaden propiedades adicionales.

Incrustación de vectores

Para permitir la recuperación y generación eficiente de asesoramiento financiero, la metodología incorpora técnicas de incrustación de vectores para representar datos textuales en un espacio vectorial continuo. Este paso es crucial para calcular similitudes semánticas entre las consultas de los usuarios y los nodos del grafo, así como para generar respuestas contextualmente relevantes utilizando modelos de lenguaje.

Para esta tarea, utilizaremos un modelo de Hugging Face en forma de Sentence Transformer, que es un modelo basado en transformadores que mapea oraciones a vectores de alta dimensión en un espacio semántico. Esto nos permite codificar tanto las consultas de los usuarios como los nodos del grafo como vectores, lo que permite cálculos de similitud rápidos y precisos.

Justificación: El uso de Sentence Transformers de Hugging Face garantiza una alta precisión y rendimiento en los cálculos de similitud semántica, lo cual es esencial para la recuperación precisa de información y la generación de respuestas.

El proceso de incrustación de vectores implica la codificación de cada nodo y arista del grafo como una representación vectorial, que luego se almacena en una base de datos vectorial para su recuperación eficiente (MongoDB), cada uno en su propia colección (arista y nodo). Cuando se recibe una consulta de usuario, también se codifica como un vector utilizando el mismo modelo Sentence Transformer, y se recuperan los nodos y aristas más similares del grafo basándose en la similitud del coseno.

Justificación: Almacenar incrustaciones de vectores en MongoDB garantiza una recuperación eficiente y escalabilidad, mientras que el uso de la similitud del coseno para la coincidencia garantiza una recuperación de información precisa y relevante.

Para fines de prueba, también realizaremos una incrustación de vectores para cada documento en su conjunto.

Responder a una consulta

El mecanismo de recuperación y generación del sistema está diseñado para recuperar eficientemente conceptos y relaciones financieras relevantes basándose en las consultas de los usuarios. Este proceso consta de varios pasos clave: procesamiento de consultas, cálculo de similitud vectorial, extracción de subgrafos y generación de respuestas.

Como se mencionó anteriormente, la consulta se limpia de la misma manera que los documentos, y luego se pasa a través del modelo Sentence Transformer para obtener una representación vectorial de la consulta. Luego, el sistema calcula la similitud del coseno entre el vector de consulta y todos los nodos y aristas del grafo. Se seleccionan los k nodos y aristas más similares para formar un subgrafo que contenga la información más relevante para la consulta del usuario.

Justificación: El uso de la similitud del coseno para la coincidencia garantiza que se seleccionen los nodos y aristas más relevantes, mientras que la formación de un subgrafo garantiza que la información recuperada sea contextualmente relevante y completa.

El subgrafo resultante de la recuperación es un término amplio, pero hemos utilizado dos enfoques diferentes para crearlo, cada uno con sus propias ventajas e inconvenientes:

Básico: este enfoque es extremadamente simple, ya que el subgrafo resultante solo incluirá los nodos y aristas recuperados y sus vecinos directos. Este enfoque es rápido y eficiente, pero puede que no capture todo el contexto de la consulta. Sin embargo, los resultados siguen siendo muy buenos, como veremos más adelante.

Avanzado: este enfoque es más complejo y costoso desde el punto de vista computacional. También conocido como el Subgrafo de Expansión Mínima, es esencialmente un problema de optimización en el que nos gustaría encontrar el subgrafo conectado que minimice la cantidad de aristas y nodos involucrados, al tiempo que se asegura de que todos los nodos y aristas relevantes estén presentes. Si no existe tal subgrafo, entonces se divide el grafo en componentes conectados y se devuelve un Subgrafo de Expansión Mínima para cada componente, por supuesto trabajando con los nodos y aristas más relevantes que están presentes en el componente respectivo.

Justificación: El uso de enfoques básicos y avanzados para la extracción de subgrafos garantiza la flexibilidad y la precisión en la captura del contexto completo de la consulta, al tiempo que se equilibra la eficiencia computacional, dependiendo de cuál se elija utilizar.

Ahora, dado el subgrafo, utilizamos un modelo de lenguaje para generar una respuesta que sea contextualmente relevante para la consulta del usuario. En este caso utilizaremos el modelo Gemini-2.0-exp, un modelo de vanguardia desarrollado por el equipo de Google AI, experto en tareas textuales. El prompt se construye concatenando la consulta del usuario con la representación textual del subgrafo, lo que proporciona al modelo el contexto necesario para generar respuestas informativas y coherentes.

Justificación: El uso del modelo Gemini-2.0-exp garantiza una generación de respuestas de alta calidad y contextualmente relevante, mientras que la construcción del prompt con el subgrafo garantiza que el modelo tenga el contexto necesario para generar respuestas precisas.

Utilizando un ejército de expertos

Lo que hemos visto hasta ahora es un sistema capaz de generar respuestas basadas en una consulta del usuario, pero ¿qué ocurre si la consulta del usuario no es lo suficientemente clara o el sistema no es capaz de recuperar la información necesaria? Aquí es donde entra en juego el ejército de expertos.

El ejército de expertos es una colección de expertos en el dominio construidos de la misma manera que el sistema, pero con la diferencia de que cada uno es experto en un campo diferente del dominio financiero. Cada uno de los llamados “expertos” idealmente tendrá un conjunto diferente de documentos y un prompt diferente para generar la respuesta. Haciendo que la respuesta que un experto genera, aunque también relativamente similar a la de otro, tenga un enfoque completamente diferente. Por ejemplo, en este trabajo vamos a utilizar 3 expertos diferentes: finanzas personales, economía e inversión. Cada uno tendrá una base de datos designada a partir de artículos y blogs de su campo.

Justificación: El uso de un ejército de expertos garantiza que el sistema pueda manejar una amplia gama de consultas financieras con alta precisión y profundidad, al tiempo que aprovecha la experiencia específica del dominio para obtener respuestas más precisas y completas.

Luego pediremos a cada experto que genere una respuesta basada en la consulta proporcionada y tendremos un agente orquestador que recopilará esas respuestas y, en otra llamada LLM, las combinará en una única respuesta para el usuario. Esto da como resultado una respuesta multifacética, que depende en gran medida de los expertos que elijamos para la tarea y de cómo compongamos el conjunto de datos para cada uno de ellos.

Justificación: La combinación de respuestas de múltiples expertos garantiza que la respuesta final sea multifacética y completa, aprovechando los puntos fuertes de cada experto específico del dominio para obtener una respuesta más precisa e informativa.