Informe Final del Proyecto Víctor Fernández: ETL y RAG para el Mercado de Coches

1. Resumen Ejecutivo

Este proyecto integra un pipeline de ETL desarrollado en Python y un sistema de Retrieval Augmented Generation (RAG) para el análisis del mercado de coches. Utilizando técnicas de scraping (con librerías como *BeautifulSoup* y *requests*), se han extraído datos relevantes de la web de km77, tales como el año del modelo, caballaje, tipo de tracción, dimensiones, tipo de cambio, entre otros. También se han extraído las reviews de los coches realizadas profesionalmente por el equipo de KM77. Los datos se almacenan en una base de datos en la nube (Supabase) y se optimizan mediante una base de datos vectorial y similitudes de coseno, lo que permite realizar consultas precisas aun cuando los nombres de los modelos pueden variar o presentarse de forma inconsistente. El proyecto innova a través de aplicar NLP a las reviews de cada coche, lo que permite dar un resultado más conciso y personal acorde con las necesidades delusuario, mientras que fomenta la "familiaridad" del lenguaje.

El RAG se integra a través de un workflow de agentes: la entrada del usuario se procesa mediante un modelo de lenguaje (LLM) para transformar las consultas a SQL, adecuarlas según el contexto y ejecutar la request a Supabase. Posteriormente, otro LLM interpreta los resultados obtenidos y genera respuestas en función de las necesidades del usuario.

Se han aplicado nuevas tecnologías a las aprendidas en el bootcamp, como Langchain,Streamlit, FuzzyWuzzy o TypedDict.

2. Descripción del Caso de Negocio

La industria del automóvil demanda cada vez más herramientas que permitan analizar grandes volúmenes de datos provenientes de múltiples fuentes y en formatos heterogéneos. En este contexto, el proyecto se orienta a:

- **Optimizar la consulta de información:** Permitir a los usuarios obtener respuestas precisas a preguntas complejas, incluso cuando la nomenclatura de los modelos varíe. También permite crear un "hub" donde se integren una gran cantidad de datos automotrices de modelos de más de 20 años-
- **Aportar valor a la toma de decisiones:** Al contar con datos actualizados y bien estructurados, se facilita la identificación de tendencias, el análisis comparativo de modelos y la evaluación de parámetros técnicos que impactan en la competitividad del mercado. Al mismo tiempo, permite

aumentar la agilidad y aprendizaje de los equipos de ventas y de cara al cliente, que cuentan en sus manos con la información de los coches.

- **Integración de tecnologías de vanguardia:** Combinando técnicas de scraping, procesamiento ETL y modelos avanzados de lenguaje, se crea una solución integral que mejora la eficiencia en la gestión y explotación de la información, todo girando en torno al entrono de OPENAI y LANGCHAIN.

Este enfoque no solo optimiza el flujo de información, sino que también reduce los tiempos de respuesta y mejora la precisión en la interpretación de datos, lo cual es crucial para empresas que buscan mantenerse competitivas en un entorno dinámico y tecnológicamente exigente.

3. Pipeline ETL

El pipeline ETL desarrollado se compone de las siguientes fases:

Extracción

- **Scraping de datos:** Se utiliza Python con librerías como *BeautifulSoup* y *requests* para extraer información detallada de la web de km77.
- **Selección de atributos:** Los datos recopilados incluyen, entre otros, el año del modelo, caballaje, tipo de tracción, dimensiones del vehículo, tipo de cambio y la descripción de un equipo de periodistas profesionales..

Transformación

- **Limpieza y normalización:** Los datos extraídos se someten a procesos de depuración para eliminar inconsistencias, duplicados y valores nulos usando librerías típicas como pandas o numpy.
- **Estandarización de formatos:** Se realiza una transformación que asegura la uniformidad en la presentación de la información, facilitando la posterior integración en la base de datos.

Carga

- **Almacenamiento en la nube:** Los datos se cargan en Supabase, con formato POSTGRESQL, con dos tipos diferentes de bases, una vectorial para encontrar similitudes de coseno entre nombres y modelo y una típica SQL consistente de tres tabalas para el resto de la información. Situar los datos en la nube que ofrece escalabilidad y alta disponibilidad, así como la portabilidad del proyecto.

Este proceso ETL garantiza que la información utilizada por el sistema RAG sea de alta calidad y esté adecuadamente preparada para su análisis y consulta.

4. Modelo estadístico y resultados

Se ha implementado un modelo de recomendación basado en RAG, flexible y que permite un trato personalizado, con la arquitectura de GPT-4o-mini. Sigue el siguiente workflow:

1. **Transformación del Input del Usuario:**
    - El input se envía a un LLM que analiza y traduce la consulta a una estructura SQL.
    - Se aplican procesos de adecuación para corregir posibles errores o ambigüedades en la consulta.
2. **Ejecución y Adaptación de la Query:**
    - El modelo traduce la query a lenguaje SQL y ejecuta la consulta en Supabase y, mediante la base vectorial, se asegura la relevancia de los resultados, aun cuando existan variaciones en la nomenclatura de los modelos.
    - Se optimizan las consultas SQL para maximizar la eficiencia en la extracción de datos.
3. **Generación de Respuestas:**
    - Otro LLM (OpenAI) interpreta los resultados obtenidos y genera respuestas contextuales y detalladas que responden a las preguntas del usuario.
    - Se garantiza una interacción fluida y coherente, combinando precisión en la consulta con la capacidad interpretativa del lenguaje natural.

Los resultados del modelo han demostrado una alta precisión en la recuperación y presentación de la información, aunque cierta falta de abstracción y memoria.

5. Impacto de Negocio y Recomendaciones

Impacto en el Negocio

- **Mejora en la toma de decisiones:** La integración de un sistema RAG con una robusta pipeline ETL permite a los responsables tomar decisiones fundamentadas basadas en datos precisos y actualizados.
- **Reducción de tiempos y errores:** La automatización de la extracción y consulta de datos minimiza la intervención manual, reduciendo errores y acelerando los procesos de análisis.
- **Adaptabilidad y escalabilidad:** La infraestructura basada en Supabase y bases vectoriales se adapta fácilmente a cambios en el mercado o la incorporación de nuevos parámetros, asegurando la continuidad y relevancia de la solución.

Recomendaciones

- **Optimización continua:** Realizar revisiones periódicas del pipeline ETL y del proceso de adecuación de queries para adaptarse a posibles cambios en la estructura de datos de la fuente original.
- **Ampliación de fuentes de datos:** Considerar la integración de nuevas fuentes de información para enriquecer el análisis del mercado y proporcionar una visión más completa.
- **Monitorización y alertas:** Implementar sistemas de monitorización que permitan detectar anomalías en tiempo real, garantizando la integridad y precisión de los datos.
- **Formación y documentación:** Asegurar que el equipo esté formado en el uso de las tecnologías implicadas y que la documentación se mantenga actualizada, facilitando la incorporación de nuevos miembros o la colaboración con otros departamentos.

6. Conclusiones y Próximos Pasos

El proyecto presentado demuestra la viabilidad y eficacia de integrar técnicas de scraping, procesamiento ETL y RAG para el análisis del mercado de coches. Entre las conclusiones destacamos:

- La sinergia entre las tecnologías empleadas (Python, Supabase, bases vectoriales y LLMs) permite un análisis profundo y preciso, incluso ante la complejidad de los datos.
- La capacidad de adecuar y transformar consultas mediante LLMs mejora notablemente la interacción del usuario con el sistema, ofreciendo respuestas contextuales y de alta relevancia.

- El enfoque modular y escalable del pipeline ETL asegura que la solución pueda adaptarse a futuros requerimientos y nuevas fuentes de información.

Próximos Pasos:

- **Implementar mejoras en el RAG y frontend:** Si bien el RAG funciona correctamente, carece de memoria implementada y puede resultar anti intuitivo a la hora de realizar consultas complejas
- **Expandir la base de datos:** Se han usado pocos datos para limitar costes, pero el proyecto es perfectamente escalable
- **Mejora de la interfaz gráfica y complemento con otras bases de datos:** Como parte de los proyectos cancelados, se incluía un mergeo de la base de datos con la base de datos del parque móvil de la DGT usando la distancia de Levenhstein como equiparador entre modelos. La base de la DGT incluía información sobre el consumo y las ventas por geografía, pero no se ha incluido por motivos de optimización.

---

7. Descripción del Repositorio

Este repositorio contiene el desarrollo de un sistema RAG (Retrieval-Augmented Generation) aplicado al mercado de coches, integrando un pipeline ETL, una base de datos en la nube con Supabase y una base de datos vectorial para mejorar la recuperación de información.

Estructura del Repositorio:

```
├── ETL
│   ├ scrapeo_km_77.ipynb
├── Modelizacion
│   ├ RAG call
│   ├ RAG agent
│   ├ app.py
├── src
├── .gitignore
└── requirements.txt
```

1. **ETL/**

- Contiene el código relacionado con la extracción, transformación y carga de datos.
- scrapeo_km_77.ipynb: Notebook donde se ha realizado el web scraping de km77 para obtener datos sobre los modelos de coches (año, caballaje, tracción, tamaño, cambio, etc.).
-

2. **Modelizacion/**
    - Contiene la implementación del RAG con diferentes enfoques:
        - RAG call/: Implementación de un RAG simple que genera consultas SQL directamente a la base de datos.
        - RAG agent/: Implementación más avanzada, utilizando agentes para mejorar la formulación de consultas y la interacción con la base de datos.
        - app.py: Archivo principal de la aplicación que ejecuta el modelo RAG final con agentes y adecuación a la tarea en Streamlit.
3. **src/**
    - Carpeta destinada a funciones y módulos auxiliares de limpieza de datos y conexión a sql para el proyecto.
4. **.gitignore**
    - Archivo que define qué archivos y carpetas deben ser excluidos del control de versiones.
5. **requirements.txt**
    - Lista de dependencias necesarias para ejecutar el proyecto.

Diferencia entre RAG Call y RAG Agent

- RAG Call: Implementación básica donde el LLM genera consultas SQL directamente y las ejecuta en la base de datos.
- RAG Agent: Enfoque más avanzado con un agente intermedio, que revisa y ajusta las consultas SQL antes de ejecutarlas, asegurando que sean correctas incluso si los nombres de los modelos no coinciden exactamente.

Este repositorio está diseñado para proporcionar una solución eficiente y escalable para la recuperación de información en el mercado de coches mediante técnicas de LLM, SQL y bases de datos vectoriales.

- **Control de Versiones:**
    - Se utiliza Git para el control de versiones.
