Informe de Desarrollo de Producto de IA Generativa para TUYA SA

Resumen

Este informe describe el desarrollo de un producto de IA generativa que proporciona información contextualizada sobre los productos financieros de TUYA SA. El producto utiliza una arquitectura RAG (Retrieve, Augment, Generate) que combina un modelo LLM (Large Language Model) con una base de datos vectorial para generar respuestas personalizadas. A lo largo del texto encontrará links útiles que se han consultado para el desarrollo de la prueba.

Esquema de desarrollo

1. Creación del Repositorio en GitHub: Se creó un repositorio en GitHub para almacenar el código del proyecto. El repositorio se estructuró siguiendo las mejores prácticas de desarrollo de código, incluyendo la creación de ramas (branches) para diferentes etapas del desarrollo.

2. Estructuración del Proyecto: Se utilizó una propuesta basada en la dada por Cookiecutter Data Science para estructurar el proyecto de manera lógica y fácil de navegar. Esto incluyó la creación de carpetas para datos, notebooks, modelos y aplicaciones.

3. Desarrollo del Producto de IA generativa: El desarrollo del producto se dividió en varias etapas, cada una de las cuales se realizó en un notebook separado para facilitar la reproducción del desarrollo.

- **3.1. Obtención de Información de Páginas Web** Se obtuvo la información de las páginas web de TUYA SA utilizando técnicas de scraping web con BeatifulSoup y con langchain.
- **3.2. Configuración del Modelo LLM** Se descargó, instaló y probó el modelo LLM unsloth/Llama-3.2-1B-Instruct en un pipeline de Langchain y Hugging Face. Se configuró el modelo para ejecutarse en GPU. En este punto, la configuración de hiperparámentros del modelo jugó un punto crucial ya que según los valores que tomaran parámetros como la temperatura o la máximo longitud el modelo podía ser más o menos creativo, o incluso fallar cuando el contexto obtenido era demasiado largo, así que se experimentó múltiples configuraciones entre el modelo y el proceso RAG que permitieron dar suficiente contexto para que el modelo de una respuesta aceptable y que a su vez pudiese manejar la cantidad de texto ingresado.
- **3.3. Procesamiento de Información y Creación de Base de Datos Vectorial** Se procesó la información (texto) extraída de las páginas web utilizando embeddings de Langchain. Esta información fue limpiada retirando caracteres especiales y espacios innecesarios dentro del documento, asegurando que se trabaje con utf-8 para una correcta comprensión de las sentencias en español y además los documentos originales se partieron en trozos más pequeños de manera que la información pueda ser recuperada con mayor precisión y por tanto centralizar el contexto que se le va a pasar al modelo. Luego, se creó una base de datos vectorial en local utilizando Chroma, dicha base contiene 2 colecciones de datos, la original y la procesada, para evitar el reprocesamiento de la información y permitir escalar la solución a la obtención de nueva información que se pueda indexar en un futuro de manera eficiente y ágil.

- **3.4. Definición del Template Prompt y Proceso RAG** Se definió un template prompt y se configuró un proceso RAG que toma una pregunta, busca en la base de datos vectorial, trae los documentos relacionados, los pasa como contexto al prompt y luego genera una respuesta utilizando el modelo LLM. El proceso de template fue crucial ya que en función de éste el modelo podía sufrir de aluciones, por lo cual se buscó aplicar 2 principios básicos de la ingeniería de promps
    1. Ser claro y conciso: Esto le permite al modelo entender cuál es su tarea, cómo debe desarrollarla y cómo debe entregar las respuestas que genera. Para ello se procuró darle un contexto de "agente de servicio al cliente" y además se delimitó claramente qué era el contexto y cuál era la pregunta a responder dentro del prompt.
    2. Dejar pensar al LLM: Este principio busca que el modelo procure revisar su respuesta antes de darla, así pues, se le indicó al modelo que no supere una cantidad de caracteres para lograr precisión y además, en algunas pruebas realizadas se le indicaba que de no conocer la respuesta simplemente dijera "No conozco la respuesta"; sin embargo, fue apreciable durante el desarrollo que al limitar el modelo con esta condición se evitaba su "creatividad" por lo que tales sentencias fueron removidas en el producto final.
- **3.5. Desarrollo de la Aplicación en Gradio** Se desarrolló una aplicación en Gradio utilizando un template sencillo y luego se personalizó utilizando el contexto Blocks. En este punto se desarrolló la aplicación de cara a un usuario básico que únicamente ingrese un prompt al chatbot, así como para uno avanzado que esté familiarizado con la configuración de hiperparámetros del modelo, permitiendo que desde la interfaz gráfica se haga una configuración de variables como "temperature", "max_lenght" para el modelo y "search_type" para definir el tipo de búsqueda de documentos o "size_context" para el número de documentos que se recuperan.
- **3.6. Documentación y Integración de Ramas en GIT** Se documentaron todas las funcionalidades y se integraron las ramas en GIT. Durante este proceso se producró emplear el estilo de numpy o scipy y además se buscó que todo el código escrito en paradigma POO respondiera a buenas prácticas de desarrollo como lo son la atomización y la reutilización de código.

Próximos Pasos

1. Desplegar la aplicación de Gradio en la nube a través de los Spaces de Hugging Face y una base de datos vectorial con la ayuda de Milvus o Qdrant, o alguna solución de código abierto; incluso podrian explorarse soluciones comerciales como MongoDB Atlas o Azure Cognitive Search.
2. Evaluar cómo se integran los LLMs con MLflow para hacer tracking de modelos.
3. Desarrollar un esquema de CI/CD con GitHub Actions o alguna solución que lo permita.

Aplicaciones futuras

- El ejercicio desarrollado puede ser empleado como base para el desarrollo de un chatbot de nivel industrial al cual se le de contexto por ejemplo de los aliados de la empresa TUYA S.A o de procedimientos reales desarrollados por agentes de servicio reales, de manera que podría desplegarse dentro de la plataforma para que de una orientación precisa a los usuario de los productos financieros de TUYA S.A.
- Otro posible aplicativo es contextualizar el LLM con información de la documentación existentes de modelos de ML que ha desarrollado el equipo de analítica en el pasado, de manera que pueda crearse una herramienta interna que brinde orientación acerca del

desarrollo de modelos, anteriores técnicas empleadas en este desarrollo y posibles mejoras de los modelos ya implementados.

- Finalmente, se piensa que este prototipo podría alimentarse de información de caracter legal de manera que se pueda crear un "agente de IA legal" que apoye la toma de decisiones basados en la normativa interna de la compañía.