Zeta AI, chatbot inteligente.

https://github.com/almarlin/chatIA

Álvaro Martínez Lineros

Ilerna Sevilla

Curso de Especialización en Inteligencia Artificial y Big Data

26 de Mayo de 2025

Resumen

Zeta AI es un chatbot inteligente que aprende de las conversaciones pasadas y adapta su comportamiento a cada usuario. El modelo usado es Mistral-7B-Instruct-v0.2, un robusto modelo de 7 mil millones de parámetros para el desarrollo conversacional.

Desarrollo

Requisitos.

Para que el proyecto funcione es necesario contar con diversas características de hardware y software:

- Node.js.
- Nvidia Rapids.
- Token de Hugginface.

Sin estos elementos el programa no podría funcionar, Node.js es necesario para la interfaz del modelo. Nvidia Rapids es necesario para que el modelo se ejecute correctamente y a una velocidad aceptable. Finalmente, el token es necesario para descargar el modelo desde Hugginface.

Diseño.

La premisa del proyecto era: “Crear un chatbot que aprenda de las conversaciones pasadas y adapte su comportamiento a cada usuario. Usar técnicas de NLP modernas (transformers, embeddings) e integración en entorno web o móvil.”.

Modelo.

Para la realización de dicha tarea fue necesario realizar una búsqueda de modelos de IA conversacionales puesto que el entrenamiento de un modelo de dichas características requeriría más tiempo del estipulado y más recursos de hardware de los disponibles.

Se valoraron diferentes modelos:

- DialoGPT. Descartado por su bajo nivel conversacional y generación de texto inconsistente.
- GPT2. Descartado por su bajo nivel conversacional y generación de texto inconsistente.
- OpenChat 3.5. Descartado por alto consumo de recursos.
- Mistral-7B-Instruct-v0.2. Seleccionado pese a su alto consumo de recursos porque su nivel conversacional es óptimo incluso en español.

Interfaz.

Para la interfaz web se seleccionó React.js como herramienta para el desarrollo. Su facilidad para el uso de interfaces dinámicas y el uso de bibliotecas de componentes fueron clave para su utilización. La biblioteca de componentes utilizada es HeroUI, una recopilación de componentes (navbar, footer, button, lists, etc.) con diseño minimalista y personalizable.

Backend.

En el backend se decidió utilizar FastAPI una biblioteca de Python altamente utilizada para el desarrollo de aplicaciones web con Python.

Funcionamiento del código.

El aplicativo completo funciona de la siguiente forma:

1. Se despliega la aplicación.
2. El usuario accede a la página e inicia sesión.
3. El servidor en FastAPI recibe la petición de inicio de sesión y devuelve los datos del usuario.

4. El usuario accede al chat y empezará a conversar con el chatbot. Las respuestas se envían a través del servidor y se muestran en la página.
5. El usuario abandona la página y con ello la sesión.

Funcionamiento del modelo.

El fichero que contiene el modelo es “mistral_chatbot.py”. Primeramente se carga el modelo y el tokenizador del mismo. Se define una clase para que el modelo detecte cuando ha de dejar de generar texto con palabras clave como Usuario, User, usuario o Human. Si se detectan, se detiene la generación de texto.

Posteriormente tiene varias funciones: construcción del prompt, generación de la respuesta en streaming y el wrapper del stream.

Construcción del prompt.

Al inicio de todos los prompts enviados por el usuario se añaden unas instrucciones y el historial de la conversación con ese usuario. De tal forma que se pueda simular la memoria del modelo. Finalmente se devuelve el prompt.

Generación de la respuesta en streaming.

Con el prompt creado y tokenizado convertido ahora a input, se envía al modelo con “generation_kwargs”. En esta función se indica y se hace fine-tuning al modelo. Se ha determinado un máximo de tokens devueltos de 60, con una temperatura de 0.7 y con ejemplificación (do_sample). Adicionalmente se envía el streamer y el criterio de parada.

Se genera todo el texto en un loop y se va enviando con el streamer a la función wrap_stream.

Wrap_stream.

Se envía a la interfaz token a token la respuesta generada por el modelo.

Resultados

El resultado final es un aplicativo sencillo y demostrativo de la aplicación de un modelo existente como chatbot, donde se ha utilizado herramientas NLP, transformers e interfaz web.

No obstante el modelo falla frecuentemente en la generación de texto, repitiendo respuestas a preguntas del historial o generando preguntas él mismo. Con un modelo más potente podría funcionar con mayor robustez y fiabilidad, no obstante, con el equipo disponible es un resultado aceptable.

El proyecto está disponible para su uso en https://github.com/almarlin/chatIA . El README.md es el mismo manual de instalación del aplicativo.