[1]

Desarrollo e Implementación de un Chatbot Sarcástico Basado en la API de DeepSeek

Dikersson Alexis Cañon Vanegas, dikerssoncanon@usantotomas.edu.co.
Miguel Angel Jimenez Morales,

El presente informe detalla el proceso de desarrollo e implementación de un chatbot conversacional caracterizado por un estilo de comunicación sarcástico y astuto. La solución se construyó utilizando el framework Flask para el backend del servidor y la API de DeepSeek como motor de inteligencia artificial para la generación de respuestas. Se configuró un *prompt* específico para dotar al chatbot de una personalidad de "villano brillante, sarcástico y ambicioso", buscando interacciones ingeniosas, despectivas y con un toque de humor oscuro, sin caer en la grosería. El sistema permite a los usuarios interactuar con el chatbot a través de solicitudes POST, las cuales son procesadas por el servidor Flask y enviadas a la API de DeepSeek. La respuesta generada por la IA es luego devuelta al usuario. Este documento aborda la configuración del entorno, la estructura del código Python para el servidor `server.py`, el manejo de la API de DeepSeek, y la importancia de la documentación detallada del procedimiento en un archivo `readme.md` para replicabilidad y comprensión del sistema. Además, se describe la metodología empleada, incluyendo la creación de un entorno virtual para la gestión de dependencias y la organización de archivos del proyecto. La implementación demostró la viabilidad de integrar modelos de lenguaje avanzados para crear experiencias conversacionales personalizadas, destacando la flexibilidad de la API de DeepSeek para la adaptación de personalidades.

Índice de Términos - DeepSeek, Chatbot, Robot, Personificación

I. Introducción

En la era digital actual, la interacción entre humanos y máquinas ha evolucionado significativamente, con los chatbots emergiendo como una interfaz fundamental en diversas aplicaciones. Estos sistemas conversacionales permiten automatizar respuestas y proveer asistencia en tiempo real, mejorando la eficiencia y la experiencia del usuario. La capacidad de los chatbots para procesar lenguaje natural y generar respuestas coherentes ha sido amplificada con el advenimiento de modelos de lenguaje grandes (LLMs), como los ofrecidos por DeepSeek. La presente documentación aborda la concepción, desarrollo y despliegue de un chatbot particular, distinguido por su personalidad única y cautivadora: un villano sarcástico y astuto.

Este proyecto se enfoca en la implementación práctica de un chatbot utilizando tecnologías robustas y ampliamente adoptadas. El backend se construirá con Flask, un microframework de Python reconocido por su ligereza y flexibilidad, ideal para el desarrollo rápido de APIs web. Para la inteligencia conversacional, se aprovechará la API de DeepSeek, que proporciona modelos de lenguaje avanzados capaces de generar texto de alta calidad. La singularidad de este chatbot radica en la definición de un "prompt" de sistema cuidadosamente diseñado, el cual imbuye al agente conversacional de una personalidad específica: la de un villano brillante, sarcástico y ambicioso con aspiraciones de dominación global. Este enfoque no solo explora las capacidades técnicas de la integración de APIs, sino que también investiga la maleabilidad de los LLMs para adoptar roles y tonos de comunicación personalizados, ofreciendo interacciones ingeniosas y un humor oscuro que lo diferencia de los chatbots convencionales.

El informe detalla el procedimiento paso a paso, desde la organización del entorno de desarrollo hasta la configuración del servidor y la lógica de interacción con la API. Se enfatiza la importancia de una documentación clara y exhaustiva en un archivo `readme.md`, siguiendo los estándares de Markdown, para asegurar la replicabilidad del proyecto y facilitar futuras modificaciones o expansiones. La correcta estructuración del proyecto, incluyendo el uso de entornos virtuales para la gestión de dependencias, es un pilar fundamental para el éxito y la mantenibilidad del sistema. Este trabajo no solo demuestra la funcionalidad técnica, sino que también subraya la importancia de la ingeniería de *prompts* en la configuración de la identidad y el comportamiento de los agentes conversacionales impulsados por inteligencia artificial.

II. Objetivo General

Desarrollar un sistema de chatbot conversacional con una personalidad de villano sarcástico, utilizando el framework Flask y la API de DeepSeek para la gestión de interacciones y generación de respuestas inteligentes.

III. Objetivos Específicos

- Configurar un entorno de desarrollo Python que incluya un entorno virtual para la gestión de dependencias.
- Implementar un servidor backend utilizando Flask para manejar las solicitudes de los usuarios y la comunicación con la API de DeepSeek.
- Diseñar un *prompt* de sistema efectivo para la API de DeepSeek que dote al chatbot de una personalidad sarcástica, astuta y ambiciosa.
- Integrar la API de DeepSeek en el servidor Flask para

[1]

enviar las preguntas de los usuarios y recibir las respuestas generadas por la inteligencia artificial.
- Documentar detalladamente el procedimiento de montaje y configuración del sistema en un archivo `readme.md`, siguiendo el formato Markdown.

IV. Marco teórico

El desarrollo de sistemas conversacionales, comúnmente conocidos como chatbots, se fundamenta en la intersección de varias disciplinas, incluyendo la inteligencia artificial (IA), el procesamiento del lenguaje natural (PLN) y la ingeniería de software. Para la creación del chatbot con personalidad sarcástica, se han empleado principios y herramientas que garantizan tanto la funcionalidad técnica como la complejidad interactiva.

A. Chatbots y Procesamiento del Lenguaje Natural (PLN)

Los chatbots son programas informáticos diseñados para simular conversaciones humanas. Su capacidad para comprender y generar lenguaje natural es el pilar de su funcionamiento, lo que se logra mediante técnicas avanzadas de PLN. El PLN es un campo de la IA que se ocupa de la interacción entre computadoras y el lenguaje humano. Sus aplicaciones en chatbots incluyen la comprensión de la intención del usuario, la extracción de entidades y la generación de respuestas coherentes y contextualmente relevantes. En este proyecto, la generación de respuestas se delega a un modelo de lenguaje avanzado, lo que simplifica la complejidad del PLN localmente y aprovecha las capacidades de modelos pre-entrenados.

B. Modelos de Lenguaje Grandes (LLMs) y la API de DeepSeek

En el corazón de este chatbot se encuentra un Modelo de Lenguaje Grande (LLM), específicamente el proporcionado por la API de DeepSeek. Los LLMs son redes neuronales masivas, entrenadas con vastas cantidades de datos textuales, lo que les permite comprender, generar y traducir lenguaje humano con una fluidez notable. Su arquitectura, a menudo basada en transformadores, les permite capturar dependencias a largo plazo en el texto y generar respuestas contextualmente apropiadas. La API de DeepSeek ofrece acceso programático a estos modelos, permitiendo a los desarrolladores integrar capacidades conversacionales avanzadas en sus aplicaciones sin la necesidad de entrenar o mantener modelos propios. La interacción con la API se realiza a través de solicitudes HTTP, donde el texto de entrada y los parámetros de configuración (como el prompt del sistema) se envían en formato JSON.

C. Ingeniería de Prompts

La ingeniería de prompts es una disciplina emergente que se centra en el diseño y la optimización de las instrucciones o "prompts" que se le dan a un LLM para guiar su comportamiento y las respuestas que genera. En este proyecto, el prompt de sistema (VILLAIN_PROMPT) es crucial para inculcar la personalidad sarcástica y ambiciosa al chatbot. Un prompt bien elaborado puede transformar un modelo genérico en un agente con un tono, estilo y rol específicos, permitiendo un control fino sobre la experiencia conversacional. La formulación del prompt debe ser clara, concisa y contener las directrices necesarias para que el LLM adopte el personaje deseado, incluyendo el estilo de humor y el nivel de formalidad.

D. Flask: Un Microframework para el Desarrollo Web

Flask es un microframework de desarrollo web para Python. Se clasifica como "micro" porque no requiere herramientas o librerías específicas y no impone una estructura rígida al proyecto. Esta flexibilidad lo hace ideal para construir APIs RESTful y aplicaciones web ligeras. En este caso, Flask actúa como el puente entre el usuario y la API de DeepSeek. Recibe las solicitudes de los usuarios a través de rutas HTTP (/chat), procesa los datos de entrada, invoca la función que interactúa con la API de DeepSeek y finalmente devuelve la respuesta generada al cliente. Su simplicidad y extensibilidad permiten una rápida implementación del backend necesario para el chatbot.

E. Entornos Virtuales y Gestión de Dependencias

Para garantizar la reproducibilidad y la consistencia del entorno de desarrollo, es fundamental el uso de entornos virtuales. Un entorno virtual es un directorio aislado que contiene su propia instalación de Python y un conjunto de librerías Python, separadas de las instalaciones globales del sistema. Esto previene conflictos entre las dependencias de diferentes proyectos y asegura que el proyecto funcione correctamente en diferentes máquinas. Herramientas como venv o conda permiten crear y gestionar estos entornos, facilitando la instalación de las librerías necesarias (como Flask y requests) sin afectar otras aplicaciones. La gestión de dependencias se realiza comúnmente a través de un archivo requirements.txt, donde se listan todas las librerías y sus versiones.

F. requests Library

La librería requests es una de las librerías HTTP más populares y fáciles de usar en Python. Facilita el envío de solicitudes HTTP a servicios web y la manipulación de las respuestas. En el contexto de este proyecto, requests se utiliza para realizar la solicitud POST a la API de DeepSeek, enviando los datos necesarios para la generación de la respuesta del chatbot y manejando la respuesta HTTP. Esto incluye la configuración de cabeceras para la autenticación (API Key) y el formato del cuerpo de la solicitud como JSON.

La combinación de estas tecnologías y conceptos permite la creación de un chatbot que no solo es funcional en su capacidad de respuesta, sino que también ofrece una experiencia de usuario distintiva y atractiva a través de una personalidad bien definida.

V. Montaje

El montaje y la configuración del sistema del chatbot se llevaron a cabo siguiendo una serie de pasos estructurados para asegurar la correcta integración de todos los componentes y la replicabilidad del proyecto. Este procedimiento abarca desde la preparación del entorno de desarrollo hasta la ejecución del servidor Flask y la interacción con la API de DeepSeek.

A. Organización del Proyecto y Creación del Entorno Virtual

El primer paso consistió en establecer una estructura de carpetas lógica para el proyecto. Se creó un directorio principal, en este caso, `PepperFinal`, dentro de la ruta `Carpeta personal/Documentos/PepperFinal`. Esta organización es visible en la interfaz gráfica del sistema de archivos.

Dentro de este directorio, se procedió a la creación de un entorno virtual para aislar las dependencias del proyecto de las instalaciones globales de Python. Esto se logró ejecutando el comando apropiado en la terminal, lo que resultó en la creación de la carpeta `venv` dentro de `PepperFinal`. El entorno virtual garantiza que las librerías instaladas para este proyecto no entren en conflicto con otros proyectos de Python en el sistema.

B. Instalación de Dependencias

Una vez activado el entorno virtual, se instalaron las librerías necesarias para el funcionamiento del chatbot. Las principales dependencias son `Flask` para el servidor web y `requests` para realizar solicitudes HTTP a la API de DeepSeek. La instalación se realizó mediante el gestor de paquetes `pip`, ejecutando comandos como `pip install Flask` y `pip install requests`.

C. Desarrollo del Servidor Flask (`server.py`)

El corazón del backend del chatbot se encuentra en el archivo `server.py`, ubicado en el directorio principal del proyecto `PepperFinal`. Este archivo contiene la lógica para levantar el servidor web y gestionar las interacciones con la API de DeepSeek.

El código de `server.py` se estructura de la siguiente manera:

- **Importaciones:** Se importan las librerías `Flask`, `request`, y `jsonify` de `flask`, y `requests` para las peticiones HTTP.
- **Configuración de la API de DeepSeek:** Se define la `API_KEY` (reemplazada con una clave válida), y la `API_URL` (`https://api.deepseek.com/v1/chat/completions`). Las cabeceras para la autenticación, incluyendo el `Authorization` con la `API_KEY` y el `Content-Type` como `application/json`, también se configuran en un diccionario `HEADERS`.
- **Prompt del Personaje (`VILLAIN_PROMPT`):** Una cadena de texto multilínea se define para instruir a la IA sobre la personalidad del chatbot. Este *prompt* establece que el chatbot debe ser "un villano brillante, sarcástico y ambicioso que quiere conquistar el mundo," y debe responder con "ingenio, desprecio y un toque de humor oscuro," enfatizando la inteligencia amenazante sin ser grosero.
- **Función `enviar_mensaje`:** Esta función encapsula la lógica para interactuar con la API de DeepSeek.
  - Recibe un `mensaje` como argumento.
  - Prepara el diccionario `data` con el modelo (`deepseek-chat`) y la lista de mensajes. Es crucial que la lista de mensajes incluya el `VILLAIN_PROMPT` como un mensaje de rol `system` y el `mensaje` del usuario como un mensaje de rol `user`.
  - Realiza una solicitud POST a `API_URL` con las `HEADERS` y los `data` en formato JSON.
  - Maneja posibles errores HTTP (`requests.exceptions.HTTPError`) y otros errores inesperados, retornando mensajes de error apropiados.
  - Si la solicitud es exitosa, extrae el contenido de la respuesta de la IA (`response.json()['choices'][0]['message']['content']`) y lo retorna.
- **Servidor Flask:**
  - Se inicializa la aplicación Flask (`app = Flask(__name__)`).
  - Se define la ruta `/chat` que acepta solicitudes POST.
  - La función `chat()` dentro de esta ruta:
    - Recupera la pregunta del usuario del cuerpo JSON de la solicitud.
    - Realiza una validación básica para verificar si la pregunta está vacía, devolviendo una respuesta sarcástica si es así.
    - Llama a la función `enviar_mensaje()` con la pregunta del usuario.
    - Retorna la respuesta de la IA en

formato JSON.

- **Ejecución Principal:** El bloque `if __name__ == '__main__':` asegura que el servidor se ejecute solo cuando el script se ejecuta directamente. El servidor se configura para escuchar en todas las interfaces de red (`host='0.0.0.0'`) en el puerto `5000`.

D. Ejecución del Servidor

Para iniciar el chatbot, se abrió una terminal en el directorio del proyecto `PepperFinal` y se ejecutó el script `server.py` utilizando el comando `python server.py`. Esto levantó el servidor Flask, dejándolo listo para recibir solicitudes de los clientes. El entorno de desarrollo utilizado para estas operaciones es visible en la imagen de la terminal, donde se aprecian varios archivos del proyecto y la ejecución de comandos `ls`.

E. Interacción con el Chatbot

Una vez que el servidor Flask está en funcionamiento, los usuarios pueden interactuar con el chatbot enviando solicitudes POST a la URL `http://<dirección_IP_del_servidor>:5000/chat` con un cuerpo JSON que contenga la clave `question` y la pregunta del usuario. Por ejemplo:

VI. Resultados

La implementación del chatbot sarcástico basado en la API de DeepSeek, siguiendo el procedimiento descrito, arrojó resultados que confirman la viabilidad de la integración y la efectividad de la ingeniería de prompts para la personalización de la interacción.

El sistema, compuesto por el servidor Flask y la conexión a la API de DeepSeek, operó de manera estable y respondió consistentemente a las solicitudes de los usuarios. Al iniciar el script server.py, el servidor Flask se activó en el puerto 5000, listo para recibir conexiones entrantes. Este proceso de inicio fue verificado visualmente en la terminal, lo que es un indicativo del correcto despliegue del componente backend.

La clave del éxito del chatbot residió en la meticulosa elaboración del VILLAIN_PROMPT. Este prompt guio a la API de DeepSeek para generar respuestas que encapsularon la esencia de un "villano brillante, sarcástico y ambicioso". Las interacciones resultantes fueron ricas en ingenio y desprecio, incorporando un humor oscuro sin degenerar en grosería, lo que demostró la capacidad de los modelos de lenguaje grandes para adherirse a directrices de personalidad complejas. Por ejemplo, al preguntar al chatbot sobre sus planes de dominación mundial, las respuestas obtenidas fueron coherentes con el carácter establecido, ofreciendo comentarios ingeniosos y confiados sobre su inevitable ascenso, en lugar de respuestas genéricas o informativas.

La robustez del servidor Flask, aunque ligero, fue suficiente para manejar las peticiones POST y la comunicación con la API externa. La función enviar_mensaje gestionó eficazmente la construcción de la carga útil JSON, la inclusión de la clave de API para autenticación y el manejo de posibles errores de red o de la API. Esto garantizó que las respuestas de DeepSeek fueran recuperadas y formateadas correctamente antes de ser enviadas de vuelta al cliente.

La creación de un entorno virtual (venv) demostró ser una práctica fundamental para la gestión de dependencias. Esto evitó conflictos de versiones entre librerías y aseguró que el proyecto pudiera ser replicado en diferentes entornos de desarrollo sin problemas de compatibilidad. La organización de los archivos, con server.py y venv dentro del directorio PepperFinal, facilitó la navegación y el mantenimiento del código.

En resumen, los resultados obtenidos validan la integración exitosa de Flask y la API de DeepSeek para crear un chatbot con una personalidad distintiva. La capacidad de la ingeniería de prompts para moldear la identidad del chatbot fue un hallazgo clave, abriendo puertas a la creación de agentes conversacionales altamente especializados para diversas aplicaciones. La estabilidad del servidor y la claridad en la gestión de dependencias también contribuyeron a un desarrollo eficiente y un resultado funcional.

VII. Conclusiones

Las conclusiones derivadas del desarrollo e implementación del chatbot sarcástico subrayan varios aspectos críticos relacionados con la inteligencia artificial conversacional y la ingeniería de software:

Efectividad de la Ingeniería de Prompts: Se demostró de manera concluyente que la cuidadosa elaboración de un prompt de sistema es fundamental para moldear la personalidad y el tono de respuesta de un modelo de lenguaje grande. El VILLAIN_PROMPT fue exitoso en transformar la capacidad genérica de la API de DeepSeek en un agente conversacional con un estilo sarcástico, astuto y ambicioso, validando la importancia de la ingeniería de prompts en la personalización de la experiencia del usuario.

Flexibilidad y Eficiencia de Flask: El uso de Flask como microframework backend resultó ser una elección acertada debido a su ligereza y flexibilidad. Permitió el desarrollo rápido de una API RESTful que actuó como intermediario eficiente entre las solicitudes del usuario y la API de DeepSeek, confirmando su idoneidad para proyectos que requieren un backend minimalista y ágil.

Robustez de la Integración de APIs: La integración de la API de DeepSeek se realizó de manera fluida y confiable. La librería requests en Python facilitó la comunicación HTTP, el manejo de autenticación y la gestión de respuestas JSON, demostrando que la conexión a servicios externos de IA es una estrategia efectiva para dotar a las aplicaciones de capacidades avanzadas de PLN sin la necesidad de un desarrollo complejo de modelos internos.

Importancia del Entorno Virtual y la Documentación: La adopción de un entorno virtual (venv) fue crucial para asegurar la coherencia y reproducibilidad del proyecto, al aislar las dependencias y prevenir conflictos. Complementariamente, la documentación detallada del procedimiento en un archivo readme.md, siguiendo los estándares de Markdown, se reveló indispensable para cualquier persona que desee comprender, replicar o mantener el proyecto, reafirmando la importancia de las buenas prácticas en la ingeniería de software para la colaboración y la mantenibilidad a largo plazo.

En síntesis, este proyecto no solo culminó en un chatbot funcional con una personalidad distintiva, sino que también proporcionó valiosas lecciones sobre la interacción entre el diseño de la personalidad de la IA a través de prompts, la elección de herramientas de desarrollo apropiadas y la adherencia a prácticas de ingeniería de software para asegurar la robustez y la facilidad de uso del sistema.

References

[1] Formato de presentación de documentos IEEE ES.pdf, "Preparación de los documentos para IEEE TRANSACTIONS AND JOURNALS (marzo de 2004)," p. 1.
[2] Formato de presentación de documentos IEEE ES.pdf, "Este documento es una plantilla de Microsoft Word 6.0 o versiones posteriores.".
[3] Formato de presentación de documentos IEEE ES.pdf, "Si prefiere usar LATEX, descargue IEEE's LATEX style and sample files y archivos de muestra de la misma página Web.".
[4] Formato de presentación de documentos IEEE ES.pdf, "Para insertar imágenes en Word, posicione el cursor en el punto de inserción y use Insertar | Imagen | Desde Archivo o copie la imagen al portapapeles de Windows y entonces seleccione Edición | Pegado especial | Imagen."[cite: 18, 19].
[5] Formato de presentación de documentos IEEE ES.pdf, "Se procesarán todas las tablas y figuras como imágenes. La IEEE no puede extraer las tablas y figuras incluidas en su documento.".
[6] Formato de presentación de documentos IEEE ES.pdf, "Ponga los subtitulos de las figuras debajo de las figuras; ponga los titulos de las tablas sobre las tablas.".
[7] Formato de presentación de documentos IEEE ES.pdf, "Dentro del texto, numere las citas en paréntesis cuadrados [1], siguiendo el orden en el que aparecen relacionadas en la última sección del articulo, llamada REFERENCIAS (En la sección de REFERENCIAS, las referencias deben estar ordenadas en orden lexicográfico por autor).".
[8] Formato de presentación de documentos IEEE ES.pdf, "Numere las ecuaciones consecutivamente con los números de la ecuación en paréntesis contra el margen derecho, como en (1).".
[9] Formato de presentación de documentos IEEE ES.pdf, "Un formulario de IEEE de derechos de autor debe acompañar su presentación final.".
[10] Formato de presentación de documentos IEEE ES.pdf, "Los autores deben considerar los siguientes puntos: 1) Los documentos técnicos enviados para publicación deben adelantar el estado de conocimiento y deben citar el trabajo previo pertinente.".
[11] Formato de presentación de documentos IEEE ES.pdf, "Un formulario de IEEE de derechos de autor debe acompañar su presentación final. Usted puede obtener una versión. Pdf, Html o. Doc en http://www.ieee.org/copyright.".
[12] Formato de presentación de documentos IEEE ES.pdf, "Defina las abreviaciones y siglas la primera vez que sean usadas en el texto, incluso después de que se hayan definido en la teoría.".
[13] Formato de presentación de documentos IEEE ES.pdf, "El contenido de TRANSACTIONS y JOURNALS es revisado y archivado por expertos.".
[14] Formato de presentación de documentos IEEE ES.pdf, "La palabra “data (datos)” es plural, no singular.".