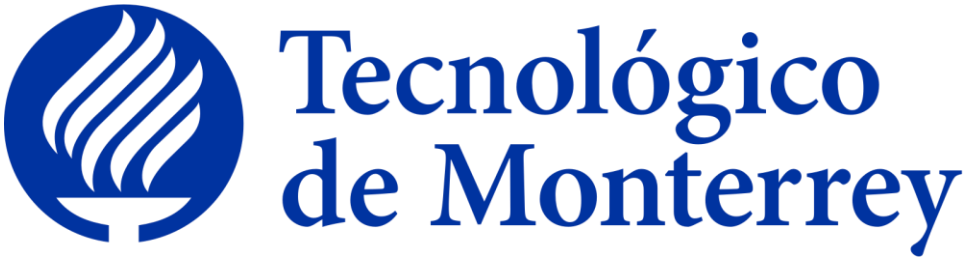
Instituto Tecnológico y de Estudios Superiores de Monterrey
Escuela de Ingeniería y Ciencias



Planeación de sistemas de software

TC3004B.102 Grupo 102

Elvia Itzamna Rosas Herrera
Lic. Martha E. Galindo V., MSI
Frumencio Olivas Álvarez
Magda Nayli Gonzalez Sánchez
Roberto Carlos Bustamante Bolaños

Integracióncon IA

Campus Monterrey

Integrantes del equipo:

Oscar (Sylvia Aurora) Cortes Rojas	A00825972
Luis Fernando Manzanares	A01283738
Mario Alberto González Méndez	A00832313
Andrea Galindo Yáñez	A01368483
Yudith Korina Hernández Palacios	A00834231

2/09/2024

Integración de la IA

La IA será utilizada para ayudar a los que usen la aplicación con sugerencias con cálculos de precios y horas, primero se separan los requerimientos en categorías, después descomponiendo los requerimientos en los pasos necesarios para ser completado y esos pasos serían las tareas y éstas con el método de T-Shirt Sizes se hace la estimación de esfuerzo, otro uso que se le dará a la IA es ser un compañero de sugerencias para ayudar a agilizar procesos

Pruebas de la IA

Por ahora, se ha utilizado Llama3.1, que resultó ser muy poderoso dando un buen rendimiento a una velocidad muy buena, dándonos unas 1000 palabras en 50 segundos, en pruebas de temperatura no se superaron los 57 grados en un cuarto con temperatura ambiente. Igual también se realizaron pruebas con llamadas a Postman utilizando los métodos POST para dar un prompt y que te responda con la respuesta de la IA. También con la detección de archivos para igual poder tener un contexto mediante archivos, por lo tanto se tiene la capacidad de agregarle contexto y entrenamiento al modelo para poder dar respuestas que tengan un margen de error adecuado (<5%).

Stack Tecnológico

Como Stack tecnológico, se ha decidido el utilizar como Frontend, React. Esto debido a que React tiene mucha capacidad para realizar aplicaciones muy eficientes con una excelente interfaz gráfica, con sus funcionalidades para hacer una aplicación sumamente eficiente y atractiva para el usuario.

Para el Backend, utilizaremos Golang o GO, es un lenguaje de programación muy eficiente, que pueda hacer cosas paralelas o concurrentes en milisegundos cuando en otros lenguajes puede llegar a tardar significativamente más, haciendo pruebas se realizó un manejador de login con MySQL hosteado en Oracle en Golang y Nodejs, donde el último se tardaba 3 segundos mientras que Go tomaba 300 milisegundos.

Como base de datos, utilizaremos MySQL. Esto porque es una buena herramienta para manejar datos de manera consistente que facilita el mantenimiento y la integridad de éstos al utilizar claves foráneas, normalización y seguridad de la información lo que lo hace una opción sólida para este proyecto.

Cascarón del proyecto

p0yo7/MoonTech: Proyecto MoonTech (github.com)

Para el cascaron del proyecto, tenemos nuestro github de MoonTech. En el cual se encuentra el video de prueba de la Llamada POST a la IA (Llama3.1) mediante Ollama, en el puerto 50000.

Ya está listo para trabajar con el stack deseado y los usuarios que van a participar ya tienen los accesos necesarios y tiene las ramas necesarias para este proyecto, que son AI, Frontend y Backend.

Hay varias formas de manejar la AI, una es mediante un servicio cloud como lo es ChatGPT entrenando una versión que permita identificar los elementos más importantes de un requerimiento para convertirlo en una tarea como lo es el contexto y el área que involucra, el problema de esto es que esa información puede llegar a ser confidencial y entregarla a una herramienta externa puede ser inseguro, la segunda es hacerlo completamente por nuestra cuenta en Python utilizando librerías como Tensorflow y Pandas, la tercera opción es hostearlo por nuestra cuenta utilizando una herramienta ya creada pero funcionando de manera local, un ejemplo es Ollama de Meta

Para poder hacer esta **prueba con Llama** utilizamos una computadora con:
Ryzen 7 7745HX
16GB de memoria RAM
Nvidia Geforce RTX 4070
1tb SSD

Es posible utilizar una computadora con menos requisitos, pero el rendimiento puede llegar a ser afectado haciendo los procesos más lentos, pero como mínimo se requiere:

16GB de RAM

Un procesador con 8 núcleos

10GB de Almacenamiento

Para usar Llama 3.1 8b es necesario la herramienta Ollama, que es donde se alojan las IAs open source y se pueden descargar desde ahí. Desde su página oficial; ollama.com se puede ingresar a los modelos, ahí se descarga primero en la computadora el ollama, y una vez descargado, en cmd buscas el comando específico para descargar un modelo. En este

caso:

ollama run llama3.1:8b

Una vez corriendo está comando, lo que sucede es que descargará el modelo de la IA, y una vez que se descargó, se podrá utilizar. Ahora, si queremos hacer un servidor local en el que podamos usarla, podemos utilizar un comando llamado

set OLLAMA_HOST=127.0.0.1:50000

Esto para seleccionar su puerto, en mi caso 50000, y una vez hecho esto podemos hacer el siguiente comando que es:

ollama serve

Este comando hará que nuestro Llama3.1 esté listo en nuestro puerto seleccionado.

Por lo que ahora en Postman podemos hacer peticiones POST para conseguir un formato JSON el cual nos regrese la respuesta de lo mencionado. Para esta prueba, utilizamos un formato muy simple el cual consta de 3 cosas:

```
{
  "model": "llama3.1",
  "stream": false,
  "prompt": "hola como estas?"
}
```

Model, es la Ia IA que utilizaremos para esta petición, stream significa si deseamos la respuesta en bloques o en tokens, o si la queremos toda completa. True, regresa

muchísimos bloques de JSON con muchas responses. False, hace que de solo una respuesta con todo dentro. Y, por último, prompt, que significa el mensaje que le escribirías a la IA. Una vez puesto ese mensaje. Lo que sucederá es que pasarán unos segundos, y al terminar nos devuelve un JSON de vuelta.


```
{
  "model": "llama3.1",
  "created_at": "2024-09-02T08:52:14.8827181Z",
  "response": "Hola! Estoy funcionando bien, gracias. ¿En qué puedo ayudarte hoy?",
  "done": true,
  "done_reason": "stop",
  "context": [
    128009,
    128006,
    882,
    128007,
    271,
    71,
    8083,
    8112,
    48591,
    30,
    128009,
    128006,
    78191,
    128007,
    271,
    69112,
    0,
    9589,
```

```
      2303,
      28963,
      4988,
      14707,
      11,
      67648,
      13,
      29386,
      1737,
      43388,
      81915,
      59237,
      20430,
      49841,
      30
    ],
    "total_duration": 2755753100,
    "load_duration": 13862700,
    "prompt_eval_count": 16,
    "prompt_eval_duration": 137510000,
    "eval_count": 19,
    "eval_duration": 2603142000
}
```


Lo que si tenemos que tomar mucho en consideración o lo más importante que necesitamos es el response, que ese nos ayudará a conseguir que nuestra IA pueda dar respuestas en nuestra futura aplicación web.

Lo que seguiría a futuro sería poder darle contexto, documentos, y todo eso sigue en desarrollo.

Como conclusión, el estar trabajando con una IA open source como lo es Llama3.1 nos deja con ese campo abierto de investigación el cual necesitamos descubrir para entender más a fondo sus límites y funcionamiento, se ve que es una opción sólida, eficiente, y poderosa, sin embargo, tenemos que seguir haciendo más desarrollo para implementarla de la mejor manera en nuestra aplicación.

Referencias

Awan, A. (2024). Fine-Tuning Llama 3 and Using It Locally: A Step-by-Step Guide. *Datacamp*

https://www.datacamp.com/tutorial/llama3-fine-tuning-locally

Sivan, V. (2024). Unlocking Llama 3: Your Ultimate Guide to Mastering Llama 3!. *Medium*

https://medium.com/pythoneers/unlocking-llama-3-your-ultimate-guide-to-mastering-the-llama-3-77712d1a0915