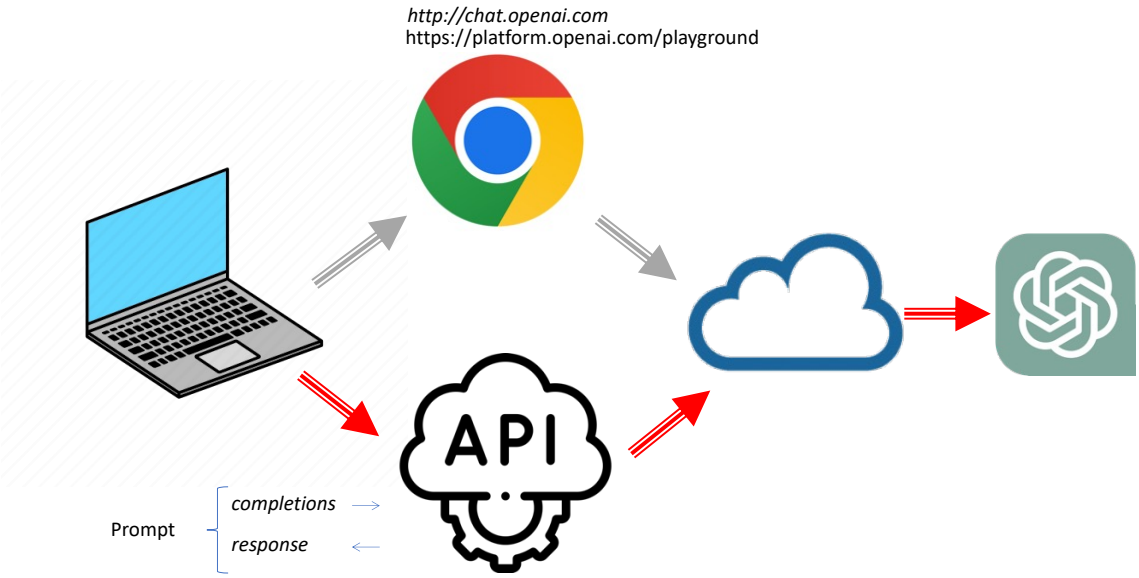
TP2 Arquitectura

Caso de Estudio chat GPT

Arquitectura base

El motor conversacional chat GPT producido por *OpenAI* puede utilizarse mediante diferentes formas de acceder a su arquitectura multicapa.



Mediante cliente ligero puede accederse vía los URL definidos y utilizar una interfaz conversacional directa bajo los parámetros del cliente ligero, la interfaz *playground* provee mejor control de los parámetros de uso mientras que la interfaz chat es más amigable para el uso casual.

También puede accederse por vía del API (*Application Program Interface*) donde se proveen una serie de parámetros y la respuesta es por medio de una estructura denominada "response". A la consulta propiamente dicha se la denomina "*prompt*" y es sujeto de considerable discusión que contenido debe tener para lograr una respuesta "óptima", a punto que se denomina, no sin cierta cuota de exageración, como "prompt engineering" al conjunto de criterios a utilizar para generar distintas respuestas.

Revisar la documentación de API provista por OpenAI, pero en líneas generales la estructura de llamada al servicio será, en lenguaje Python, como sigue:

```
response = client.chat.completions.create(
  model="gpt-3.5-turbo-0125",
  messages=[
    {
      "role": "system",
      "content": context },
    {
      "role": "user",
      "content" : usertask },
    {
      "role": "user",
      "content": userquery }

  ],
  temperature=1,
  max_tokens=4096,
  top_p=1,
  frequency_penalty=0,
  presence_penalty=0
)
print(response.choices[0].message.content)
```

Donde los valores típicos de los parámetros *usertask, context y userquery* se corresponden a la declaración de propósito, el contexto de operación y la consulta propiamente dicha respectivamente.

Preparación para el práctico

Para recrear el ambiente necesario para ejecutar el práctico deberán realizarse las siguientes actividades preliminares.

Cuenta en OpenAI

Ir al sitio http://www.openai.com y generar una cuenta (gratuita), luego de hacerlo solicitar y obtener una API KEY (revisar recomendaciones sobre como informarla en el sitio OpenAI).

Instalar librería OpenAI

Instalar la librería localmente según lo instruido por OpenAI en la documentación de uso del API.

Consignas

1. Escriba un programa en lenguaje Python que basándose en el esqueleto previamente proporcionado acepte una consulta del usuario, verifique si la misma tiene texto, imprima su contenido, invoque el API de chatGPT con esa consulta e imprima en pantalla el resultado que se obtenga como respuesta. El contenido de la consulta debe agregársele "*You:*" antes de imprimirlo y de enviarlo. La respuesta de chatGPT deberá agregársele "*chatGPT:* "antes de imprimirse.
2. Agregue al programa anterior estructuras *Try:/Except*: para gestionar problemas en la ejecución, coloque un nido para la aceptación de consulta desde el usuario, otro para su tratamiento y un tercero para la invocación.
3. Agregue la posibilidad de recuperar la última consulta realizada para poder editarla y volver a enviar a chatGPT, esto debe hacerse con la tecla "cursor Up".
4. Acepte un argumento de llamada "—*convers*" y en caso de existir genere un modo de "conversación" caracterizado por:
    a. Asegúrese de revisar si el argumento existe y es el esperado, caso contrario debe ser ignorado (consejo: revise primero si se ha indicado algún argumento).
    b. Cada vez que se acepta una consulta no nula se agrega a un buffer.
    c. Cada interacción con *chatGPT* utilizará la última consulta realizada agregada a todas las anteriores.
    d. La respuesta de *chatGPT* se agregará al buffer para ser también re-enviada en las próximas consultas.

5. Ejecute el programa *multimetric* sobre la última versión que disponga del programa utilizado en éste practico. Realice las siguientes consignas:
    a. De la sección "*Overall*" del resultado verifique el resultado de "*comment_ratio*" (proporción de comentarios). Si éste es menor de 1/3 (33%) explore medidas para mejorar éste parámetro hasta un valor en ese entorno.
    b. Explore el significado de y luego compare los valores de *halstead_effort* y *halstead_timerequired* con los que efectivamente le tomó el programa.
    c. Como compara el valor *halstead_bugprop con la cantidad de* defectos que tuvo que solucionar luego que lograra que el programa ejecute por primer vez (es decir, excluyendo errores de sintaxis).
    d. Que estrategias cree que se pueden aplicar para reducir el índice de McCabe (*cyclomatic_complexity*) en un 10%.
    e. Capture en la entrega del práctico las distintas acciones y resultados que va obteniendo (pueden hacer cut & paste si eso le ayuda a simplificar la escritura).
6. Instale un analizador estático de código denominado pylint (mediante el comando *pip install pylint*) y realice las siguientes acciones.
    a. Ejecute el programa pylint sobre el programa python desarrollado como parte de las consignas del punto -4- anterior.
    b. Analice el resultado que arroja.
        i. Introduzca correcciones en el programa fuente para abordar los comentarios. Incluya los comentarios en la primer corrida y como queda luego de todas las correcciones realizadas.
    c. Aquellos comentarios que decida no abordar justifique brevemente porque decidió ignorar las recomendaciones.
7. Solicite a chatGPT que sugiera modificaciones y mejoras al programa Python elaborado.

Entrega

Como resultado de la ejecución de la consigna deberá colocarse el programa final en una carpeta llamada

"chatGPT" en la rama "src" del repositorio GitHub personal creado en prácticos anteriores. Como entrega se entregará un enlace al código fuente escrito.

Deberá agregarse una memoria con los resultados obtenidos en la ejecución del programa **multimetric**