1. Diferencias entre 'completion' y 'chat' models

Ambos métodos están diseñados para generar texto en base a diferentes consultas, tomando un prompt como entrada y prediciendo la mejor respuesta a esa consulta.

La diferencia entre estos dos métodos radica en la cantidad de interacciones que tendrán, es decir, para el caso de completion el modelo generará una salida que es una continuacion del texto (un solo turno). Para el caso de chat, podemos tener varias sesiones de consulta asumiendo diferentes roles y manteniendo el historial de la conversación generada (múltiples turnos).

Además, teniendo en cuenta el contexto, para completion las solicitudes serán independientes de las anteriores. Caso contrario en chat, el modelo tiene en cuenta el historial para brindar contexto a la conversación ‘recordando’ lo que se ha consultado.

Las salidas son mas naturales y tienen una estructura de conversación cuando utilizamos completion, haciendolas mas adecuadas para aplicaciones de chatbots.

2. ¿Cómo forzar a que el chatbot responda 'si' o 'no'?¿Cómo parsear la salida para que siga un formato determinado?

Una manera de lograrlo es especificando en las indicaciones del prompt que su respuesta debe ser solo ‘Si’ o ‘No’ entre comillas especificando que no se desea contexto para las respuestas, especificando que respuestas queremos en una lista, dandole ejemplos de como deberia ser la respuesta esperada, especificando mediante expresiones regulares la salida esperada (alguno modelos cuentan con expresiones de paraad).De esta forma, estamos forzando al modelo a que tenga solo dos respuestas.

Para que siga un formato determinado, es posible especificarlo en el prompt (de forma similar a como hicimos en el caso de ‘Si’ o ‘No’), podemos definir si esperamos un objeto JSON y cual es la estructura que debe seguir indicando que variables completaran los valores del mismo. En cualquier caso, la temperatura del modelo deberia estar ajustada en valores minimos para que devuelva exactamente lo que se solicita. Existen también librerías como langchain, que permiten crear templates especificos, estructurando las salidas en el formato que el usuario desea.

3. Ventajas e inconvenientes de RAG vs fine-tunning

Una de las mayores ventajajs del RAG es su capacidad de aprovechar fuentes de informacion externas ya existentes para un tema en particular. Lo que permite generar respuestas más acordes y coherentes dentro del contexto proporcionado directamente de la fuente de información suministrada, reduciendo el riesgo de alucinación. Por otro lado, el fine-tunning, utiliza modelos preentrenados lo que le permite mejorar la precisión del modelo en un tema o

tarea particular, pero no permite una actualización adecuada para información que cambia constantemente.

Realizar tareas con RAG, tiene una gran dependencia de la calidad de la información suministrada, por lo que será necesario contar con fuentes confiables. Además, el proceso para recuperar información puede ser computacionalmente muy costoso si se trata de grandes cantidades de información. Caso contrario, el fine-tunning es más flexible y computacionalmente menos costoso, ya que, cuenta con una base preentrenada. Sin embargo, esto puede generar que el modelo olvide información aprendida cuando se entrene con nueva información.

Cabe mencionar que en el RAG, se minimizan los riesgos a exponer datos de índole privada ya que no son requeridos para el entrenamiento. Por el contrario, para casos que involucran temas especificos con información de conocimiento general lo adecuado utilizar el fine-tunning.

4. ¿Cómo evaluar el desempaño de un bot de Q&A? ¿Cómo evaluar el desempeño de un RAG?

Existen diferentes métricas que podemos utilizar para medir el desempeño de un chatbot:

- Cantidad de respuestas correctas entregadas por el chatbot,
- la cantidad de preguntas por conversación, que da una idea de la cantidad de preguntas que son necesarias para que el chat responda correctamente la pregunta que se le ha realizado,
- A/B testing, que consta de tener dos versiones del chatbot para evaluar cual performa mejor. Incluyendo, diferentes flujos de conversaciones, estilos de respuestas, funcionalidades, etc.
- evaluar la latencia de respuesta, que es el tiempo que transcurre hasta obtener una respuesta,
- comparar que similitud existe entre las respuestas que genera el chatbot con las respuesta que esperamos obtener,

Para RAGs:

Existen metricas mencionadas para chatbots que también pueden utilizarse este caso, ademas de:

- Medir qué tan relevantes son las respuestas, es decir, que tan especifica es la respuesta y qué nivel de detalle es provisto,
- la confianza y exactitud de la respuesta, son métricas a tener en cuenta, ya que nos dará una idea de la precisión entregada dado un cierto contexto,
- utilizar modelos que ya cuentan con un índice de precisión y confianza definido para comparar las respuestas obtenidas por nuestro modelo (por ejemplo, ChatGPT 4),
- existen métricas como ROUGE, para evaluar traducciones y resumenes automáticos; BLUE, para evaluar traducciones automáticas; BLEURT, basado en el modelo BERT para evaluar la generación de lenguaje natural y; METEOR para evaluar resultados de traducción automática,

contando con metricas como stemming, coincidencias sinonímicas y coincidencia de palabras. (Para mi caso en particular, no he utilizado estas métricas dada la naturaleza de los trabajo involucrados).