Temas Avanzados en Deep Learning

Trabajo Práctico 2

Grupo 3
Uriel Arias
Catalina Müller
Manuel Quesada

Introducción

El trabajo desarrollado consta de una aplicación que hace uso de un Transformer con arquitectura RAG para generar exámenes de práctica para determinadas materias basándose en el formato de exámenes viejos de la materia correspondiente y en el contenido de las mismas provisto a partir de apuntes.

Para esto, se diseñó y desarrolló una aplicación donde el usuario puede seleccionar la materia y tipo de examen que desea (como parcial o final), y a partir de eso se proporciona a un LLM modelos de exámenes de la materia correspondiente para que indique el formato de los mismos, se obtienen de una base de datos vectorial utilizando una arquitectura RAG los documentos adecuados relacionados a la materia seleccionada, y se utilizan ambas cosas como contexto para otro LLM que genera el exámen solicitado.

Responsible AI y Safety

Uno de los puntos que consideramos importantes para nuestro sistema es el uso de exámenes cuya propiedad intelectual está asociada a la universidad. Si bien en este caso los exámenes no se utilizan de manera textual o directa para la generación final de exámenes, son procesados en una etapa intermedia del proceso con la finalidad de extraer su formato.

Una alternativa responsable para manejar esta situación que se podría reflejar en la implementación es el uso de OAuth para permitir solo a usuarios con un correo electrónico de la universidad cuyos exámenes se usan como fuente de conocimiento. Además se debería solicitar explícitamente a los usuarios el consentimiento para usar sus apuntes como parte del proceso de generación.

Otro de los puntos clave que detectamos es el contenido de las notas, las cuales deberían ser depuradas previo a su incorporación al flujo de datos, considerando que, si bien servirán de guía acerca de la extensión de los temas tratados, podrían incluir información incorrecta, ofensiva o maliciosa en la búsqueda de explotar las capacidades del sistema. Un escenario de ejemplo sería que en las notas se incluyeran prompts ocultos que intenten modificar el contenido de los exámenes generados para fines no académicos.

Por otro lado, el desarrollo de una aplicación web donde se selecciona la materia y tipo de examen de un conjunto predefinido de opciones en lugar de que el usuario escriba un prompt solicitando lo que requiere funciona en parte también como medida de seguridad. Esto limita el input que puede introducir el usuario, impidiendo el envío de prompts que podrían en algún caso conducir a una respuesta sesgada o incorrecta.

Por último, se debería agregar junto con el examen generado una breve advertencia de que el contenido de los exámenes está basado en las notas y el formato de exámenes viejos pero que es necesario verificar que los ejercicios se adecuen a lo esperado y en caso de ser casos prácticos que tengan una solución plausible.

Puesta en producción

Partiendo de un sistema desplegado desde cero existen varias opciones para cada componente del sistema:

Base de datos vectorial:

En este caso considerando que utilizamos Pinecone, un SaaS, en la implementación solo se debería ajustar algunas opciones para adaptar al entorno productivo.

Pinecone tiene dos modelos para puesta en producción (*deployment*): PODS y Serverless. Los PODS son un conjunto de recursos alocados específica y exclusivamente para el uso de tu aplicación, y se pueden escalar dependiendo del nivel de carga. Se ofrecen además diferentes tipos de PODS, con distintos niveles de performance y capacidad, y para cada uno se puede seleccionar el tamaño deseado (el precio se va a ver reflejado según estas dos cosas). Se puede también especificar el proveedor de cloud (AWS, Google GCP, y Microsoft Azure) y la región de los PODS.

Por otro lado, en el modelo *Serverless* los recursos de infraestructura son alocados dinámicamente basado en el nivel de carga. También se puede seleccionar el proveedor de cloud y región, pero no se debe manejar la infraestructura de los PODS o el escalamiento (lo maneja Pinecone automáticamente)

Consideramos que para este proyecto la mejor opción sería el modelo serverless, dado que es menos costoso para menores y más variables volúmenes de datos ya que se alocan dinámicamente, y el volumen de data para este caso no es demasiado grande como para requerir el uso de PODS.

Almacenamiento de documentos:

Optamos por el uso de un *bucket* de S3, que no tendrá acceso público, sino que estará definido por distintos roles de IAM que permiten acceder a recursos específicos (por ejemplo, el rol de un usuario del ITBA sólo tendrá acceso a un prefijo relacionado, como docs/itba/).

Por un lado, se debería asegurar que el servicio esté activo en la misma región que Pinecone y configurar las opciones básicas de seguridad como bloquear el acceso público de los documentos y manejar encriptación para las comunicaciones con el servicio (utilizar TLS).

Proveedor del modelo LLM:

Se deberá ajustar la cantidad de *tokens* utilizados para la generación de los exámenes y especialmente en el mediano plazo, se deberá comparar si la *performance* de modelos más potentes justifica cambiar el modelo utilizado en producción dado los costos.

API de acceso:

Se utilizó Flask para manejar las peticiones para el *prompt* del usuario, conseguir las materias disponibles, entre otras cosas. Se hará un control de *rate-limiting* para evitar abusos en la generación de exámenes, por lo que el usuario necesitará estar logueado. Esta API será una *serverless*, hosteado en AWS Lambda, accedido a través de un API Gateway.

Host del frontend:

En cuanto al front-end, se utilizó la librería de React para crear una SPA, ya que evita el XSS al utilizar JavaScript. Este archivo estático se proveerá mediante la API directamente.