udp

Tarea 1: Plataforma de análisis de preguntas y respuestas en Internet

Profesor: Nicolás Hidalgo
Ayudantes: Isidora González, César Muñoz Rivera, Natalia Ortega y Joaquín Villegas

Introducción y Contexto

Tras el desarrollo del sistema base en la primera entrega, se ha logrado establecer un flujo funcional para la comparación de respuestas entre un LLM y el dataset histórico de Yahoo! Answers. Sin embargo, la interacción directa y síncrona con servicios externos, como la API de un LLM, ha expuesto debilidades inherentes a este enfoque, tales como la gestión de límites de cuota (rate limiting) y la vulnerabilidad ante fallos o sobrecargas del servicio.
El presente entregable busca robustecer la arquitectura implementada, evolucionando hacia un modelo de procesamiento asíncrono y resiliente. Para ello, se introducirán dos herramientas estándar en la industria de los sistemas distribuidos: Apache Kafka como bus de mensajes para desacoplar los servicios y gestionar las cargas de trabajo, y Apache Flink como motor de procesamiento de flujos para analizar los resultados en tiempo real e implementar un ciclo de mejora continua.
El objetivo es transformar el sistema actual en un pipeline de datos robusto, capaz de manejar fallos de manera elegante y de mejorar activamente la calidad de las respuestas generadas.

Entregable 2: Resiliencia, procesamiento de flujos y calidad de respuestas

El objetivo principal de esta segunda entrega es rediseñar la arquitectura del sistema para incorporar mecanismos de tolerancia a fallos y un procesamiento de datos más avanzado. Esto se logrará mediante la integración de un sistema de colas de mensajes (Apache Kafka) y un framework de procesamiento de streams (Apache Flink).
La nueva arquitectura deberá ser capaz de encolar las solicitudes al LLM para gestionarlas de forma asíncrona, manejando específicamente los errores de sobrecarga y límite de cuota. Adicionalmente, se implementará un flujo que, utilizando Flink, analice la calidad de las respuestas ya generadas y active un proceso de regeneración para aquellas que no cumplan con un umbral de calidad definido por el alumno.
La interacción entre los componentes se presenta en la Figura 1.

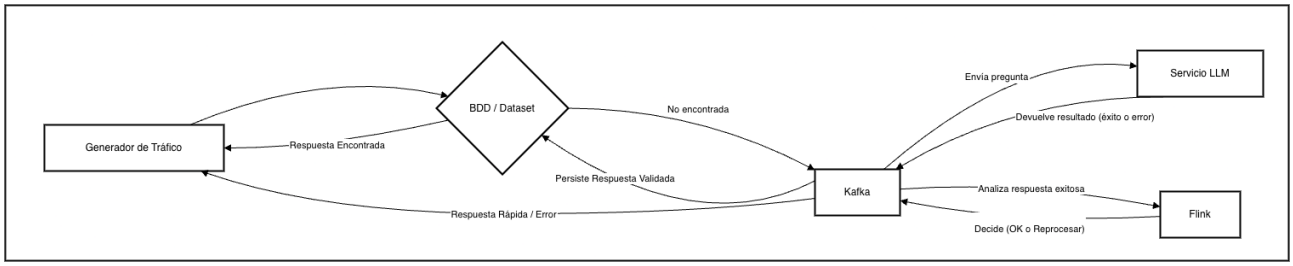

Figura 1: Arquitectura Asíncrona con Kafka y Flink

El funcionamiento de la plataforma se puede resumir en los siguientes pasos:

1. El Generador de Tráfico selecciona una pregunta y consulta al sistema de almacenamiento.

2. El sistema debe consultar primero la base de datos del Almacenamiento para verificar si esa pregunta ya ha sido procesada por el LLM y tiene un score asociado. Si existe, se utiliza esa respuesta (y opcionalmente, se podría popular la caché).

3. Si la pregunta no se encuentra en el sistema de Almacenamiento, se inicia el pipeline de procesamiento asíncrono. La pregunta se envía a Apache Kafka para ser procesada por los servicios correspondientes.

4. Un servicio consumidor se encargará de obtener la pregunta desde Kafka y gestionará la comunicación con el LLM. Este servicio debe ser capaz de manejar las respuestas exitosas y los diferentes tipos de errores (como sobrecarga del modelo o límites de cuota), derivando cada caso a un flujo de procesamiento adecuado dentro de Kafka.

5. Un trabajo en Apache Flink consumirá los mensajes de las respuestas que fueron generadas exitosamente por el LLM. Este calculará el score de calidad (definido en la Tarea 1).

   - Si el score supera un umbral definido por el alumno, el resultado final se considerará válido y se enviará a través de Kafka para ser persistido por el módulo de Almacenamiento.
   - Si el score es inferior al umbral, Flink deberá reenviar la pregunta al flujo de procesamiento inicial en Kafka para que el sistema intente generar una mejor respuesta.

6. Finalmente, el módulo de Almacenamiento consumirá los resultados validados y los persistirá de forma definitiva en la base de datos.

Requerimientos y Organización del Sistema

Para la correcta evaluación de esta entrega, el sistema deberá cumplir con los siguientes objetivos específicos:

- **Diseño de un Pipeline Asíncrono con Kafka:** Se debe reestructurar el sistema para que la comunicación con el LLM y el procesamiento posterior se realice de forma asíncrona utilizando Apache Kafka.
  - El equipo deberá diseñar, implementar y justificar una topología de tópicos y consumidores que permita gestionar de forma resiliente el ciclo de vida de una consulta. Se debe explicar el propósito de cada tópico, el rol de cada productor y de cada consumidor en el sistema.
  - El diseño debe contemplar, como mínimo, el flujo para nuevas preguntas, el manejo de respuestas exitosas y la gestión de, al menos, dos tipos de errores provenientes del LLM que requieran reintentos.
  - Los módulos existentes (Almacenamiento, Scoring, etc.) deben ser adaptados para funcionar como productores y/o consumidores de Kafka.
- **Gestión de Fallos y Estrategias de Reintento:** Se debe implementar la lógica para manejar los errores de la API del LLM de forma desacoplada.
  - El equipo deberá proponer y justificar una estrategia de reintento para cada tipo de error gestionado (ej. exponential backoff para sobrecarga, reintento retardado para cuotas), implementándola a través de consumidores de Kafka.
- **Procesamiento de Flujos con Flink:** Se requiere la implementación de un trabajo en Apache Flink que procese el stream de respuestas obtenidas del LLM.
  - Este trabajo deberá leer desde el tópico de respuestas exitosas, aplicar la función de score desarrollada en la entrega anterior y tomar una decisión basada en el resultado.
  - Si una respuesta es considerada de baja calidad (score bajo), Flink debe reinyectar la pregunta al sistema publicándola nuevamente en el tópico de preguntas pendientes. Se debe implementar un mecanismo para evitar ciclos infinitos (ej. limitar el número de reintentos por pregunta).
- **Distribución y Despliegue**
  - El sistema completo deberá estar en contenedores utilizando Docker. Esto asegura la portabilidad del entorno y simplifica el proceso de despliegue y escalabilidad de los servicios.
  - Todos los nuevos servicios (Kafka, Flink, consumidores, etc.) y los modificados deberán estar contenerizados utilizando Docker y orquestados mediante docker-compose.yml
- **Documentación y Buenas Prácticas**
  - Se deberá entregar una documentación técnica exhaustiva que abarque tanto la funcionalidad implementada como el código fuente[1]. Esta incluirá una justificación rigurosa de las decisiones de diseño, la elección de tecnologías y las metodologías de desarrollo aplicadas.

[1]Se requiere que el código esté alojado y documentado en un sistema de control de versiones, como GitHub o GitLab, siguiendo las buenas prácticas de la industria (ej. README.md detallado, comentarios en el código, etc.).

Análisis y Discusión

En esta sección, se debe realizar un análisis crítico y una discusión fundamentada sobre el nuevo diseño y los resultados obtenidos, respaldado por datos empíricos. Se deben abordar obligatoriamente los siguientes puntos:

Análisis del Sistema de Colas (Kafka)

- Expliquen las ventajas y desventajas de migrar a un modelo asíncrono. ¿Cómo impactó esta decisión en la latencia percibida por el usuario (simulado) y en el throughput general del sistema?
- Describan y justifiquen la estrategia de reintento implementada para los casos de sobrecarga y cuota. Presenten datos que demuestren su efectividad (ej. número de preguntas recuperadas exitosamente, tiempo de espera promedio en la cola).

Análisis del Procesamiento de Flujos (Flink)

- Definan y justifiquen rigurosamente el umbral de score que utilizaron para decidir si una respuesta era de baja calidad. ¿Cómo llegaron a ese valor?
- Presenten métricas que demuestren la efectividad del feedback loop. ¿Mejoró el score promedio de las respuestas tras la re-generación? ¿Cuál es el costo computacional de este ciclo (ej. aumento de llamadas al LLM)?

Requisitos de la Entrega

La evaluación de este hito se basará en los siguientes entregables:

- **Informe Técnico:** Un documento en formato LaTeX (entregado como PDF) que describa de manera concisa y rigurosa la arquitectura y el enfoque del sistema. El informe debe centrarse en el **análisis crítico** que justifique las decisiones tomadas y los resultados obtenidos. El contenido debe incluir:
  - Una descripción detallada de los componentes y las funcionalidades implementadas en cada módulo del sistema.
  - Una justificación fundamentada de las decisiones de diseño, así como de las tecnologías y metodologías empleadas.
  - Un análisis exhaustivo de los resultados, respaldado por datos empíricos. Se deben utilizar gráficos, tablas comparativas y otros recursos visuales para facilitar la interpretación y evaluación de las métricas definidas.
- **Video de Demostración:** Un video con una duración de 10 minutos donde se muestre el sistema en funcionamiento, explicando sus componentes y el flujo de datos.
- **Código Fuente:** El código fuente completo del proyecto, el cual deberá estar alojado en un repositorio público en GitHub o GitLab. El enlace al repositorio deberá incluirse tanto en la sección de comentarios de la plataforma CANVAS como en el informe técnico.
- **Archivos de Despliegue:** Un archivo `Dockerfile` y/o `docker-compose.yml` que permita la construcción y ejecución de todos los servicios implementados. El archivo `README.md` del repositorio deberá contener instrucciones claras y precisas para el despliegue y la ejecución del sistema.

Reglas y consideraciones de la entrega

- **Fecha de Entrega:** La fecha límite para la entrega de esta tarea es el día 29/10/2025 hasta las 23:59 hrs. Se recomienda gestionar el tiempo de forma efectiva. La entrega es vía Canvas del curso.
- **Integrantes:** La tarea debe ser realizada en grupos de hasta **2** alumnos/as.
- **Ética y Autoría:** Se debe respetar el reglamento de la universidad en cuanto a plagio y autoría. Cualquier evidencia de copia o falta de autoría conllevará sanciones según lo establecido.