Comprehensive Revision Notes: System Design for Student Case Study Evaluation

Introduction

This session focused on building an end-to-end machine learning system that automates the evaluation of case studies submitted by students. The core aim is to design a system that reduces the turnaround time from possibly weeks to just seconds using cloud technologies, notably AWS. The session introduced key components, including creating a front-end interface for uploads, processing pipelines, and leveraging AWS services for various tasks like storage, text extraction, and embedding creation.

System Components

Front-End Development

- **User Interface**: A simple interface should be created where students can upload their case study PDFs. This interface can be built using technologies like Streamlit, React, or any other familiar frameworks 【4:0†transcript.txt】.

AWS S3 for Storage

- **File Storage**: Once students upload their PDFs, they should be stored in AWS S3, a basic object storage service. This acts as the entry point for the system【4:0†transcript.txt】.

PDF Processing and Text Extraction

1. **Triggering the Pipeline**: The upload to S3 triggers a pipeline using AWS Lambda, which will handle subsequent processing【4:0†transcript.txt】.
2. **Text Extraction**: Text is extracted from PDFs using AWS Textract or libraries like PyMuPDF for text retrieval. This step is crucial to convert PDF content to a processable format 【4:0†transcript.txt】【4:18†transcript.txt】.

Data Preprocessing

- **Preprocessing Steps**: This involves cleaning the extracted text, removing special characters, and preparing data for further processing. Preprocessing ensures the text is in a clean, structured format suitable for generating embeddings【4:0†transcript.txt】.

Generating Embeddings

- **Embedding Creation**: After preprocessing, the system generates embeddings using AWS services or frameworks like LangChain and OpenAI models. Embeddings are vector representations of the text which capture semantic meanings【4:0†transcript.txt】【4:18†transcript.txt】.

1. **Vector Database Usage:** Systems like ChromaDB, Pinecone, or FIAS (Facebook AI Similarity Search) are used to store and retrieve embeddings efficiently. These databases are optimized for high-speed searches over complex datasets【4:1†transcript.txt】.
2. **Data Sharding:** Embeddings are partitioned into multiple shards for efficient storage and retrieval【4:1†transcript.txt】.
3. **Indexing and Similarity Search:** Indexing these embeddings enables quick retrieval during query evaluation, using techniques like Inverted File Index (IVF) and HNSW (Hierarchical Navigable Small World)【4:5†transcript.txt】【4:9†transcript.txt】.

Evaluation and Scoring

1. **Automated Scoring:** The system is designed to replace manual evaluation with an automated scoring mechanism that mimics human evaluation. This is done by training models on previously evaluated submissions【4:13†transcript.txt】.
2. **Feedback Generation:** The system should also be capable of providing qualitative feedback similar to human evaluators【4:13†transcript.txt】.

Additional Features and Considerations

- **Moderation**: Implement optional moderation features using AWS Comprehend to ensure appropriate handling of inappropriate content【4:6†transcript.txt】【4:17†transcript.txt】.
- **Result Storage and Accessibility**: Results can be stored in AWS DynamoDB for NoSQL needs or AWS Redshift for SQL storage. Alternatively, results can be stored back in S3 with querying available via AWS Athena【4:14†transcript.txt】.

Conclusion

The entire process is aimed at providing a scalable, efficient solution to automatically evaluate student case studies with minimal manual intervention. By utilizing AWS and various advanced ML services, the system maintains accuracy while significantly reducing evaluation time, allowing instructors and administrators to manage large-scale educational operations more effectively.