**Architectural Decision Record (ADR)**

**1. Title**

**Adoption of FAISS K-Means for Clustering Ungraded Answers to Optimize LLM API Usage**

**2. Context**

In our short-answer grading system, we need to process a large volume of ungraded responses efficiently while minimizing API costs. Each response requires an LLM call, making individual processing expensive and inefficient. Our key objectives include:

* **Batch Processing:** Reduce the number of LLM API calls by grouping similar responses.
* **Lower API Costs:** Optimize LLM token usage by avoiding redundant processing of similar answers.
* **Scalability:** Ensure the system can handle large datasets without performance degradation.
* **High Accuracy in Grading:** Maintain grading consistency across clustered answers.

Given these requirements, we explored multiple clustering techniques to group similar responses before sending them to the LLM for batch processing.

**3. Decision**

We have selected **FAISS K-Means** as the clustering method for ungraded answers to enable batch processing and reduce redundant LLM API calls.

**Rationale**

* **Efficient Clustering:** FAISS (Facebook AI Similarity Search) provides **high-speed nearest neighbour search**, allowing us to efficiently cluster similar short answers.
* **Reduced LLM API Calls:** Grouping similar answers allows us to send a **single batched request** instead of individual API calls, cutting costs significantly.
* **Optimized Token Usage:** By clustering, we ensure that answers within a batch stay within LLM token constraints, avoiding unnecessary splits.
* **Scalability:** FAISS K-Means is highly optimized for large-scale clustering and can handle millions of responses efficiently.

**Rejected Alternatives & Trade-offs**

| **Alternative** | **Reason for Rejection** |
| --- | --- |
| **DBSCAN** | Struggles with high-dimensional embeddings and may classify too many responses as noise, reducing batch efficiency. |
| **Hierarchical Clustering** | Computationally expensive for large datasets, leading to higher latency. |
| **Manual Rule-Based Batching** | Difficult to maintain and lacks adaptability to diverse answer variations. |
| **No Clustering (Individual API Calls)** | Leads to excessive API costs and redundant token usage. |

**4. Architecture Impact**

* **New Components:**
  + **FAISS K-Means Clustering Module**: Groups ungraded answers into semantically similar clusters before passing them to the LLM.
  + **Batch Processing System**: Handles API requests efficiently by sending grouped responses for grading.
* **Integration Points:**
  + FAISS integrates with our **vector database** for similarity search.
  + Works alongside **RAG (Retrieval-Augmented Generation)** to ensure responses are retrieved effectively.
  + Outputs are stored in the **Aptitude Test Grade Database** for future reference.
* **Performance Considerations:**
  + Clustering must be executed **within seconds** to maintain system responsiveness.
  + Cluster sizes should balance **batch efficiency** and **grading accuracy**.

**5. Risks & Mitigation**

| **Risk** | **Impact** | **Mitigation Strategy** |
| --- | --- | --- |
| Clustering may not be perfect | Medium | Fine-tune clustering parameters (number of clusters, embedding similarity threshold). |
| Uneven batch sizes | Low | Dynamically adjust batch sizes based on token constraints. |
| FAISS model drift over time | Medium | Periodic retraining with newly graded responses to improve clustering accuracy. |

**6. Acceptance Criteria**

| **Requirement** | **Description** | **Acceptance Threshold** |
| --- | --- | --- |
| **API Cost Reduction** | Percentage reduction in LLM API calls | **≥ 70%** |
| **Processing Speed** | Time taken to cluster and batch process responses | **≤ 2 seconds per batch** |
| **Accuracy** | Agreement between AI-graded and human-reviewed responses | **≥ 85%** |
| **Batch Size Optimization** | Number of responses per LLM call | **≥ 50 responses per batch** |

**7. Implementation Plan**

* **Phase 1:** Develop and integrate FAISS K-Means clustering module.
* **Phase 2:** Test clustering effectiveness and optimize batch sizes.
* **Phase 3:** Deploy in a staging environment and analyse cost reductions.
* **Phase 4:** Full deployment with continuous monitoring and optimization.

**8. Decision Status**

* **🟢 Accepted** (Implementation in progress)

**9. Related Documents**

* Short-Answer Grading Architecture Diagram
* FAISS Clustering Performance Benchmark Report
* API Cost Analysis Before & After FAISS Implementation