Incremental MMR Clustering & Query Rewriting Experiment

This experiment implements an **Incremental Multi-Turn RAG Pipeline** that simulates a real-world conversation flow. Unlike standard benchmarks that use pre-computed history, this pipeline generates its own context (agent responses) turn-by-turn.

Pipeline Workflow

For each conversation, the system processes turns sequentially ($T_1 \rightarrow T_N$):

1. **Context Building**:
   - For Turn N, context is built from previous User Queries (from dataset) and **Generated Agent Responses** (from turns $1...N-1$).
2. **Query Rewriting (MMR Clustering)**:
   - Extracts sentences from the conversation history.
   - Clusters sentences (using BGE embeddings) to identify topics.
   - Selects diverse context using **Maximal Marginal Relevance (MMR)**.
   - Uses LLM (Mixtral) to rewrite the current query into a standalone version using the selected context.
3. **Retrieval (ELSER v2)**:
   - Retrieves documents for the *rewritten query* using Elasticsearch (ELSER v2 model).
4. **Generation**:
   - Generates an answer using the retrieved documents and history.
   - **Crucial Step**: This generated answer becomes the history for Turn $N+1$.

Step-by-Step Method

1. **Sentence Extraction**
   - Breaks down conversation history into individual sentences.
   - **User turns**: Treated as single units.
   - **Agent turns**: Split into sentences; filters out conversational filler (e.g., "Thank you") and short phrases.
2. **Embedding**
   - Generates vector embeddings for the current query and all extracted history sentences using `BAAI/bge-base-en-v1.5`.
3. **Clustering (Topic Discovery)**
   - Groups historical sentences using **K-Means clustering**.
   - **Cluster Count (k)**: Calculated as $k \approx \sqrt{n}$, clamped between 2 and 7.
     - *Why 2-7?* This range is an empirical heuristic for context window management. It ensures there are enough clusters to capture

distinct topics ($min = 2$) but prevents the candidate pool from becoming too large ($max = 7$) for efficient MMR processing.
- **Representative Selection**: From each cluster, the system selects the top sentences closest to the centroid.
  - Default: 3 representatives per cluster.
  - *Example*: With 7 clusters, $7 \times 3 = 21$ candidate sentences are passed to the MMR stage.

4. **MMR Selection (Diversity & Relevance)**
   - Applies **Maximal Marginal Relevance (MMR)** to the candidate pool (cluster representatives).
   - **Formula**: $\text{Score} = \lambda \cdot \text{Sim}(s, \text{Query}) - (1 - \lambda) \cdot \max_{s_j \in \text{Selected}} \text{Sim}(s, s_j)$
     - λ (Lambda, default 0.7): Weight for relevance vs diversity. 0.7 favors relevance slightly more than diversity.
   - **Process**:
     1. **Loop 1**: Selects the sentence with highest similarity to the current query (Redundancy term is 0).
     2. **Loops 2+**: Selects the next sentence that is relevant to the query but *dissimilar* to already selected sentences.
   - Selects a fixed number of sentences (default: 5) to form the final context for the LLM.
5. **LLM Rewriting**
   - Constructs a prompt with the **selected context sentences** and current query.
   - Uses Mixtral to rewrite the query into a standalone version.
   - Post-processes the output (capitalization, punctuation).

Example Walkthrough (Conversation `3f5fa37...`)

Turn 1: “what makes the different shapes of the moon” * *No history yet.* * **Retrieve & Generate**: System answers about “lunar phases”.

Turn 2: “What is the distance between the moon and earth?” * *Context*: Turn 1 Q + Turn 1 Generated Answer. * **Rewrite**: “Could you tell me the distance between the Earth and the Moon?” * **Retrieve & Generate**: System answers with average distance.

Turn 5: “what makes monkeys suitable for space travel?” * *Context*: Turns 1-4 Qs + Generated Answers. * **Rewrite**: “Why are monkeys suitable for space travel?” (Resolved reference). * **Retrieve**: Finds documents about primates in space. * **Generate**: Answer becomes context for Turn 6.

Output Structure

The pipeline produces two main output types:

1. Rewritten Queries (`datasets/`)

JSONL file containing the final result of the rewriting process for each turn.

```
{
  "_id": "task_id",
  "text": "|user|: Rewritten query text",
  "agent_response": "Generated answer text used for next turn's context"
}
```

2. Intermediate Data (`intermediate/`)

JSONL file containing detailed debugging and analysis information for every step of the pipeline.

Key Fields: * `original_query`: The user's raw query. * `rewritten_query`: The standalone query produced by the LLM. * `num_extracted_sentences`: Total sentences found in the conversation history. * `extracted_sentences`: List of all sentences from history. * `num_clusters`: Number of semantic clusters formed (based on history size). * `cluster_sizes`: Distribution of sentences across clusters. * `representative_sentences`: Sentences closest to the centroid of each cluster. * `selected_sentences`: Final sentences chosen by MMR for the rewriting context. * `sentence`: Text of the sentence. * `speaker`: Who said it (user/agent). * `turn`: Which turn it came from. * `cluster_id`: Which cluster it belongs to. * `contexts`: **Retrieved Documents** used for generation. * `title`: Document title. * `text`: Document content snippet. * `score`: Retrieval relevance score (ELSER). * `url`: Source URL. * `agent_response`: The answer generated by the LLM using the retrieved contexts.

Usage

Prerequisites: * `ES_URL` and `ES_API_KEY` (Elasticsearch) * `TOGETHER_API_KEY` (LLM API)

Run Full Experiment:

```
python3 mmr_cluster_rewrite.py --domain clapnq --num-sentences 5
```

Output: Results are saved to `datasets/` (rewritten queries) and `intermediate/` (pipeline stats).