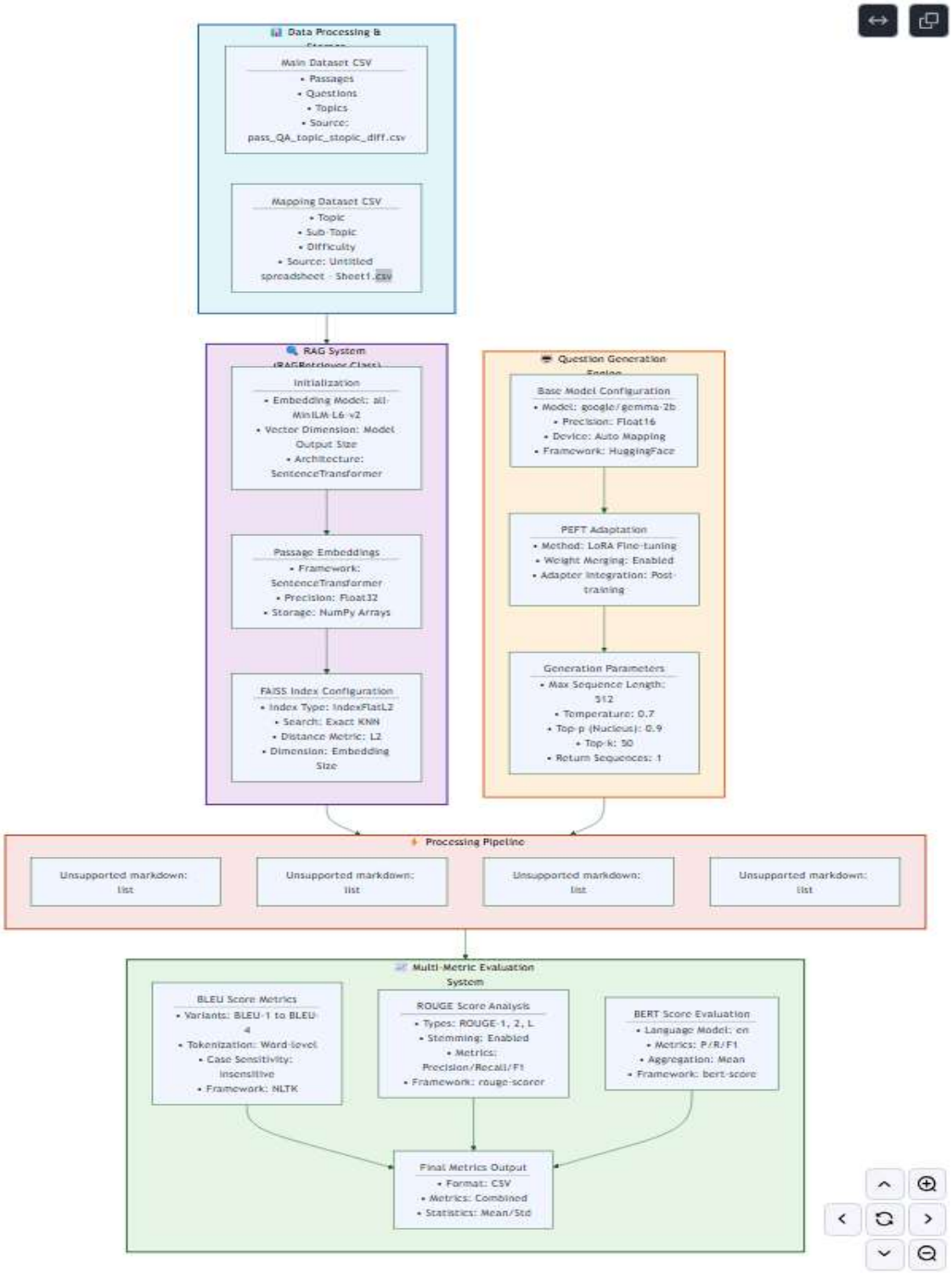
RAG-Based Question Generation and Evaluation System Documentation



1. System Overview

This system implements a comprehensive pipeline for generating and evaluating questions using a RAG (Retrieval-Augmented Generation) approach combined with fine-tuned language models. The system processes passages, generates relevant questions, and evaluates them using multiple metrics.

2. Data Processing & Storage

2.1 Input Data Structure

Main Dataset (pass_QA_topic_stopic_diff.csv)

- **Content**: Contains primary data for question generation
- **Columns**:
  - ID: Unique identifier for each entry
  - Topic: Main subject area
  - Sub-Topic: Specific area within the main topic
  - Passage: Source text for question generation
  - Difficulty: Complexity level of the content
  - Question: Reference questions
  - Answer: Corresponding answers

Mapping Dataset (Mapping CSV)

- **Purpose**: Maps relationships between topics, subtopics, and difficulty levels
- **Structure**:
  - Topic: Primary subject classification
  - Sub-Topic: Secondary classification
  - Difficulty: Standardized difficulty rating

3. RAG System Implementation

3.1 Embedding System

- **Model**: SentenceTransformer (all-MiniLM-L6-v2)
  - Architecture: Transformer-based
  - Output Dimension: Model-specific embedding size
  - Precision: Float32 for maximum accuracy

3.2 FAISS Index Configuration

- **Index Type**: IndexFlatL2
  - Search Method: Exact K-Nearest Neighbors
  - Distance Metric: L2 (Euclidean distance)
  - Storage Format: Dense vector matrix
- **Performance Characteristics**:
  - Exact search (no approximation)

  - Linear time complexity O(n)
  - Suitable for medium-sized datasets

3.3 RAGRetriever Class Implementation

```
class RAGRetriever:
    def __init__(self, main_csv, mapping_csv,
embedding_model='all-MiniLM-L6-v2')
```

- Initializes embedding model
- Processes all passages into embeddings
- Creates and populates FAISS index
- Manages data filtering and retrieval operations

4. Question Generation Engine

4.1 Base Model Configuration

- **Model**: google/gemma-2b
  - Precision: Float16 for efficient inference
  - Device Mapping: Automatic based on hardware
  - Framework: HuggingFace Transformers

4.2 PEFT (Parameter-Efficient Fine-Tuning)

- **Method**: LoRA (Low-Rank Adaptation)
  - Weight Merging: Enabled post-training
  - Integration: Adapter-based approach
  - Memory Efficiency: Optimized for deployment

4.3 Generation Parameters

- **Configuration**:
  - Maximum Length: 512 tokens
  - Temperature: 0.7 (balanced creativity)
  - Top-p (Nucleus Sampling): 0.9
  - Top-k: 50
  - Number of Return Sequences: 1
- **Prompt Structure**:

```
Generate a concise single question based on the following passage:
Topic: {topic}
Subtopic: {subtopic}
Difficulty: {difficulty}
Passage: {passage}
Question:
```

5. Evaluation Metrics System

5.1 BLEU Score Implementation

- **Framework**: NLTK
- **Variants Calculated**:
  - BLEU-1: Unigram matching
  - BLEU-2: Bigram matching
  - BLEU-3: Trigram matching
  - BLEU-4: 4-gram matching
- **Configuration**:
  - Case Sensitivity: Disabled
  - Tokenization: Word-level
  - Smoothing: None

5.2 ROUGE Score Analysis

- **Framework**: rouge-scorer
- **Types**:
  - ROUGE-1: Unigram overlap
  - ROUGE-2: Bigram overlap
  - ROUGE-L: Longest Common Subsequence
- **Features**:
  - Stemming: Enabled
  - Metrics Calculated:
    - Precision
    - Recall
    - F1-score
  - Reference Handling: Multiple references supported

5.3 BERT Score Evaluation

- **Configuration**:
  - Language Model: English
  - Framework: bert-score
  - Metrics:
    - Precision: Token-level precision
    - Recall: Token-level recall
    - F1: Harmonic mean
  - Aggregation: Mean across all scores

6. Processing Pipeline

6.1 Data Filtering

1. **Input Processing**:

    - Topic filtering
    - Subtopic matching
    - Difficulty level selection
2. **Validation**:
    - Data completeness check
    - Format validation
    - Error handling

6.2 Passage Retrieval

1. **Process**:
    - Convert query to embedding
    - FAISS similarity search
    - Result ranking
2. **Parameters**:
    - K-nearest neighbors: Configurable
    - Distance threshold: None (exact search)

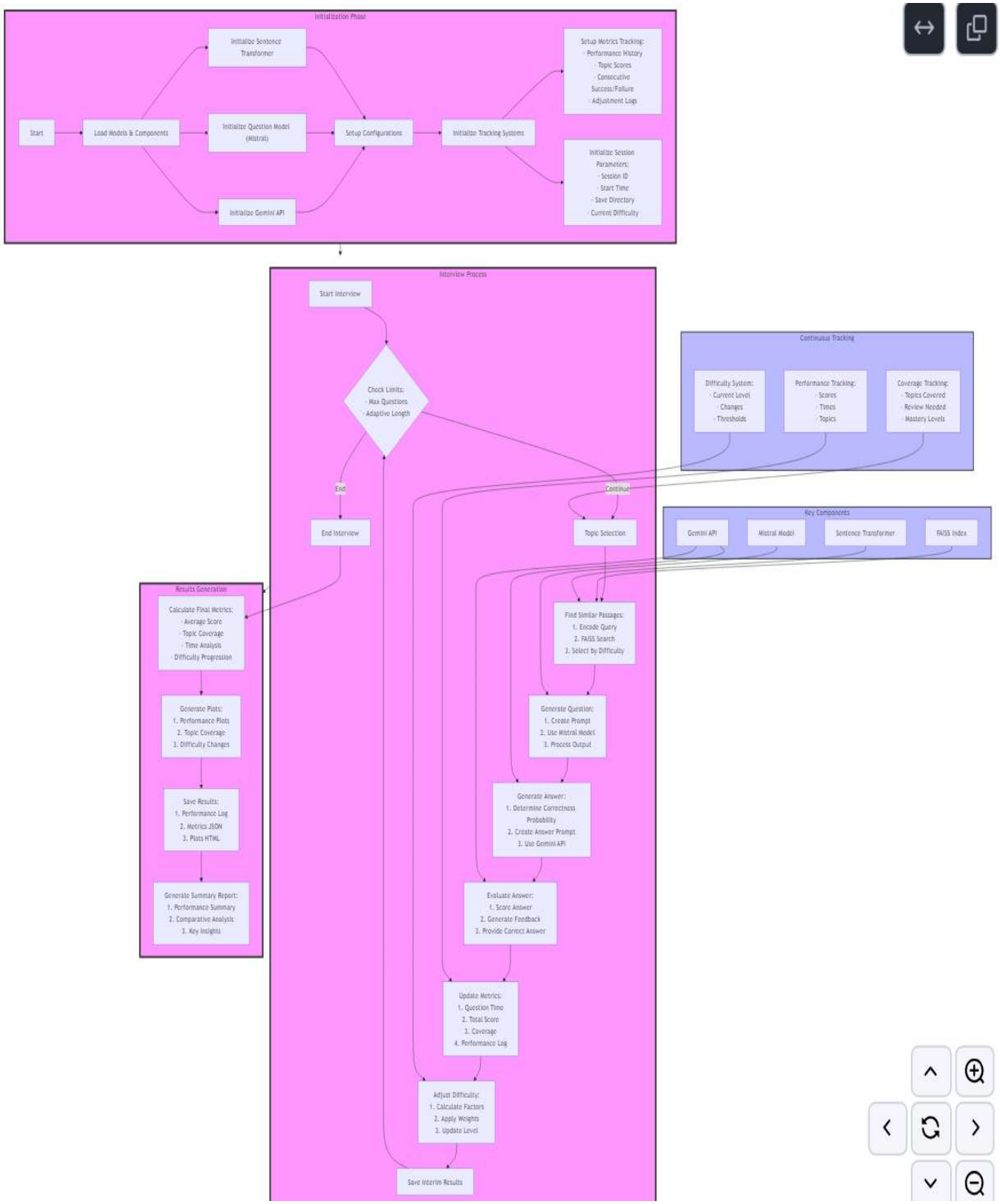
6.3 Question Generation

1. **Steps**:
    - Context preparation
    - Model inference
    - Post-processing
2. **Output Handling**:
    - Question validation
    - Format standardization
    - Error recovery

6.4 Evaluation Process

1. **Metric Calculation**:
    - Parallel processing of metrics
    - Score normalization
    - Statistical analysis
2. **Output Format**:
    - CSV with all metrics
    - Summary statistics
    - Performance analysis

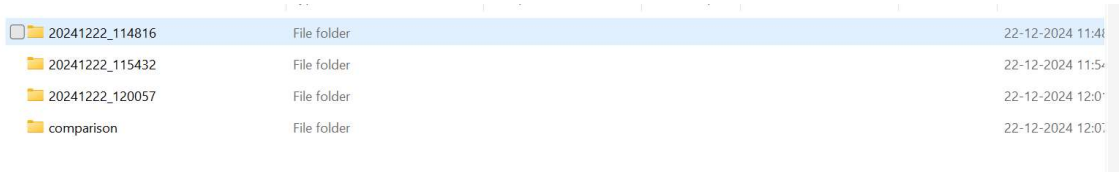
Interview Pipeline



Link to the Drive Link for pipeline results

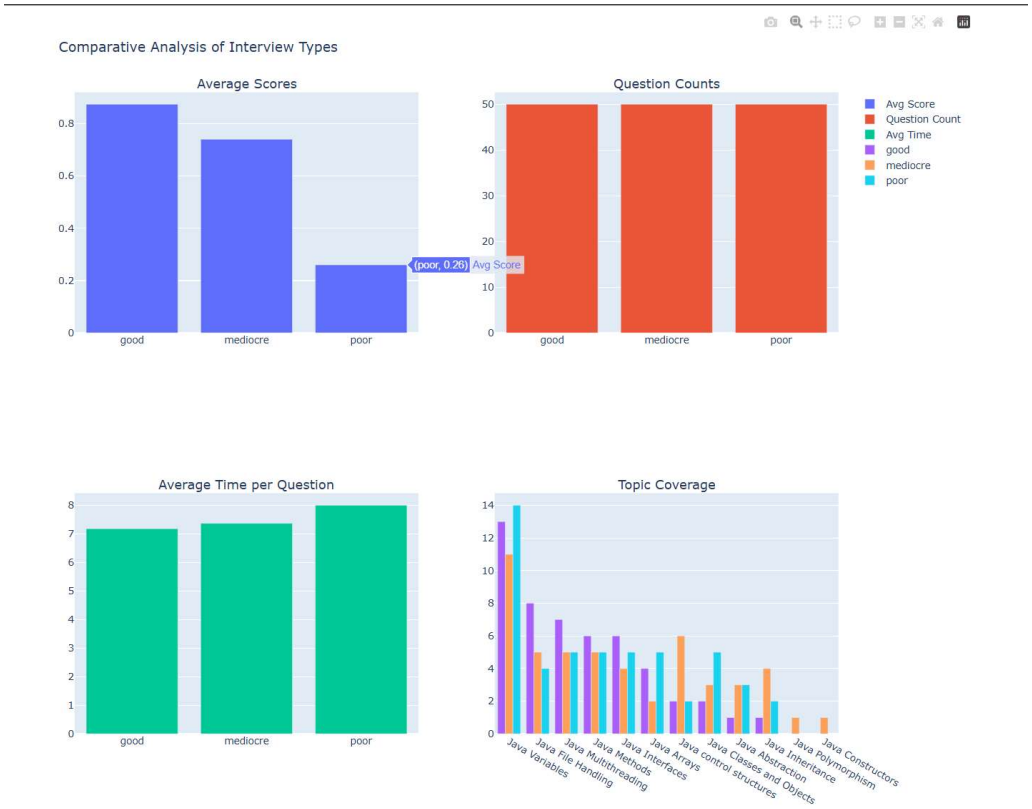
https://drive.google.com/drive/folders/1HxQx58e6atOFFROyTLCSaCXjKmO6YO16?usp=drive_link

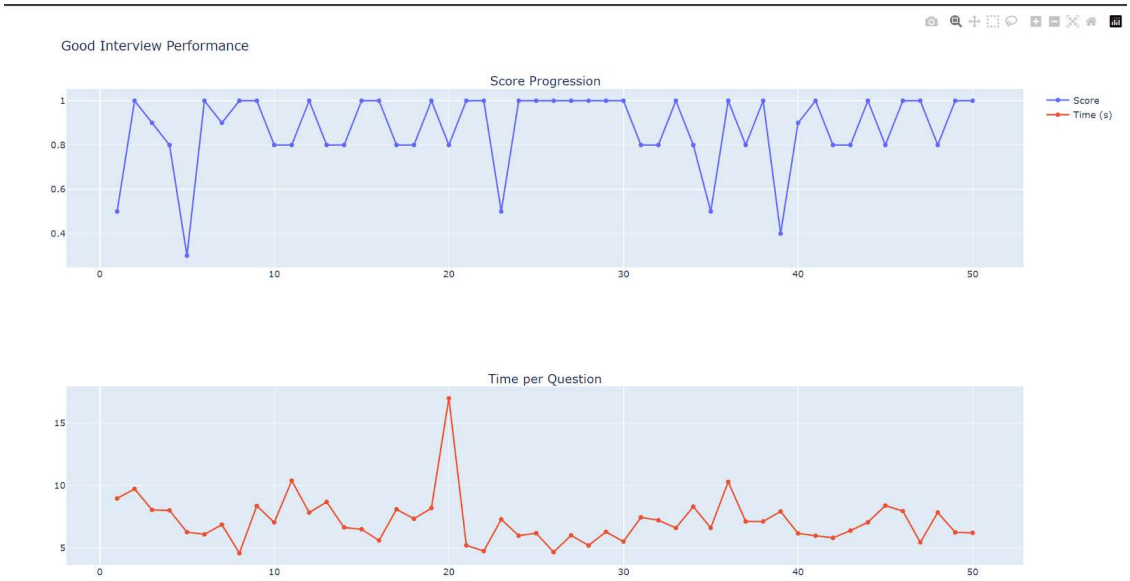
Folder Structure of the zip file

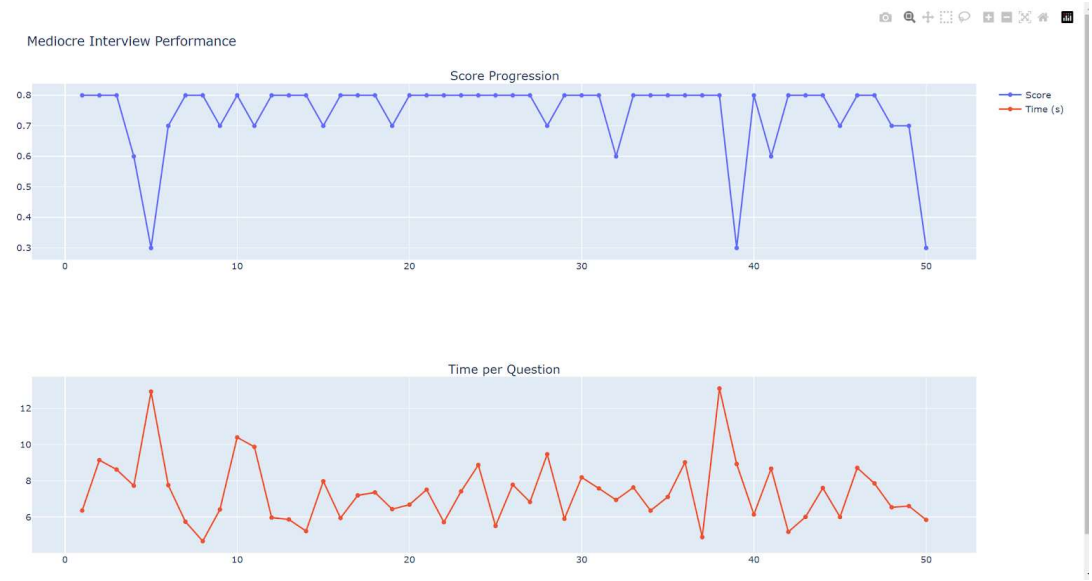


image

Sample Output of comparision btw the three types of interviews







image

image

Advanced AI Interview Pipeline Documentation

Table of Contents

- Overview
- Initialization
- Interview Process
- Performance Types
- Difficulty Adjustment System

- Topic Management
- Evaluation System
- Metrics and Analytics
- Results and Artifacts
- Data Storage

Overview

The AI Interview Pipeline is a sophisticated system designed to conduct adaptive technical interviews using multiple AI models. It simulates different interviewer personalities (good, mediocre, poor) and automatically adjusts difficulty based on candidate performance.

Initialization

Model Loading

1. **Sentence Transformer**
   - Model: 'all-MiniLM-L6-v2'
   - Used for: Passage similarity search and embedding generation
   - Configuration: Default parameters for efficient embedding
2. **Question Generation Model**
   - Model: Mistral-7B with custom fine-tuning
   - Configuration:
     - Max sequence length: 700
     - Temperature: 0.7
     - Top-p: 0.9
     - Top-k: 50
3. **Answer Evaluation Model**
   - Model: Google's Gemini Pro
   - Used for: Answer generation and evaluation
   - Temperature: 0.7 for controlled randomness

Initial Setup

1. **Session Parameters**
   - Unique session ID (timestamp-based)
   - Interview type selection
   - Initial difficulty level (default: medium)
   - Save directory creation
2. **Tracking Systems**
   - Performance history array
   - Topic coverage tracking
   - Difficulty adjustment logs
   - Time tracking initialization

Interview Process

Question Generation Flow

1. **Topic Selection**
   - Check uncovered topics
   - Verify topic distribution
   - Consider topics needing review
   - Balance new vs review topics
2. **Passage Selection**
   - Generate topic embedding
   - FAISS similarity search (k=3)
   - Filter by current difficulty
   - Select most relevant passage
3. **Question Creation**

```
prompt = f"""
Topic: {topic}
Subtopic: {subtopic}
Difficulty: {difficulty}
Passage: {passage}
Question:
"""
```

Performance Types

Good Interview Type

- **Correct Answer Probability**: 70%
- **Score Ranges**:
  - Correct answers: 0.8-1.0
  - Incorrect answers: 0.3-0.5
- **Characteristics**:
  - High-quality explanations
  - Minimal irrelevant information
  - Proper terminology usage
  - Comprehensive coverage

Mediocre Interview Type

- **Correct Answer Probability**: 50%
- **Score Ranges**:
  - Correct answers: 0.6-0.8
  - Incorrect answers: 0.2-0.4
- **Characteristics**:
  - Mixed quality explanations

  - Some irrelevant information
  - Occasional terminology errors
  - Partial topic coverage

Poor Interview Type

- **Correct Answer Probability**: 10%
- **Score Ranges**:
  - Correct answers: 0.4-0.6
  - Incorrect answers: 0.0-0.2
- **Characteristics**:
  - Low-quality explanations
  - Significant irrelevant content
  - Frequent terminology errors
  - Incomplete coverage

Difficulty Adjustment System

Factors Considered

1. **Recent Performance** (35% weight)
   - Last 3 questions' scores
   - Trend analysis
   - Moving average calculation
2. **Topic Mastery** (25% weight)
   - Per-topic average scores
   - Coverage completeness
   - Review frequency
3. **Consecutive Performance** (25% weight)
   - Success streaks (≥2 correct)
   - Failure streaks (≥2 incorrect)
   - Pattern recognition
4. **Overall Session** (15% weight)
   - Average session score
   - Time-based performance
   - Difficulty progression

Adjustment Rules

```
def _get_final_difficulty(weighted_score):
    if weighted_score >= 1.5:
        return 'hard'
    elif weighted_score >= 0.5:
        return 'medium'
    else:
        return 'easy'
```

Stability Mechanisms

1. **Hysteresis Prevention**
   - Minimum questions per difficulty: 2
   - Required score difference: 0.2
   - Cooldown period between changes
2. **Topic-Based Stability**
   - Topic mastery threshold: 0.8
   - Maximum consecutive same topic: 3
   - Topic difficulty correlation

Topic Management

Coverage Tracking

1. **Topic Registry**
   - Main topics and subtopics
   - Completion status
   - Review flags
   - Mastery levels
2. **Selection Algorithm**

```
def select_next_topic():
    if len(covered_topics) >= total_topics:
        reset_coverage()
    return priority_queue.get_next_topic()
```

3. **Review System**
   - Topics below threshold marked for review
   - Spaced repetition implementation
   - Mastery-based scheduling

Evaluation System

Answer Assessment

1. **Scoring Components**
   - Content accuracy
   - Completeness
   - Relevance
   - Terminology usage
2. **Feedback Generation**
   - Specific error identification
   - Improvement suggestions
   - Correct answer explanation

    - Score justification
3. **Score Normalization**
    - Interview type consideration
    - Difficulty level adjustment
    - Consistency checking

Metrics and Analytics

Real-time Metrics

1. **Performance Metrics**
    - Running average score
    - Topic-wise performance
    - Time per question
    - Difficulty progression
2. **Coverage Metrics**
    - Topics completed
    - Review frequency
    - Mastery levels
    - Gap analysis

Generated Plots

1. **Performance Plots**
    - Score progression line chart
    - Time per question trend
    - Difficulty changes
    - Topic mastery heat map
2. **Topic Analysis**
    - Coverage sunburst chart
    - Performance radar chart
    - Review frequency histogram
    - Mastery progression
3. **Comparative Analysis**
    - Interview type comparisons
    - Difficulty distribution
    - Time efficiency analysis
    - Error pattern analysis

Results and Artifacts

Saved Results

1. **Session Data**

```
{
  "session_id": "20241222_123456",
  "interview_type": "mediocre",
  "metrics": {
    "total_score": 0.75,
    "questions_asked": 15,
    "average_time": 120.5,
    "topic_coverage": 0.8
  }
}
```

2. **Detailed Logs**
   - Question-answer pairs
   - Timing information
   - Score breakdowns
   - Feedback details
3. **Analytics Reports**
   - Performance summary
   - Topic coverage analysis
   - Time efficiency report
   - Improvement suggestions

Generated Files

1. **HTML Reports**
   - Interactive plots
   - Session summary
   - Detailed analysis
   - Performance breakdown
2. **JSON Data**
   - Raw metrics
   - Question details
   - Performance logs
   - Configuration data
3. **Plot Files**
   - Performance visualizations
   - Topic coverage charts
   - Comparative analysis
   - Time series analysis

Data Storage

Directory Structure

```
interview_results/
├── session_id/
│   ├── plots/
│   │   ├── performance_plots.html
│   │   ├── topic_coverage.html
│   │   └── difficulty_changes.html
│   ├── metrics.json
│   ├── performance_log.json
│   └── summary_report.txt
└── comparison/
    ├── comparative_analysis.html
    ├── combined_metrics.json
    └── summary_report.txt
```

Storage Format

1. **Metrics JSON**
   - Session parameters
   - Performance metrics
   - Coverage statistics
   - Time analytics
2. **Performance Log**
   - Detailed question records
   - Response tracking
   - Score calculations
   - Timing data
3. **Summary Report**
   - Overall performance
   - Key insights
   - Recommendations
   - Comparative analysis