AI-Powered Adaptive Teaching Assistant

Technical Documentation :

Author: Mouli P

Repository: https://github.com/moulendra/ai-teaching-assistant

Introduction:

This project implements an AI-powered adaptive teaching assistant designed to simulate how modern EdTech platforms understand student queries and personalize learning paths.

The system integrates Natural Language Processing (NLP), Machine Learning (ML), and reinforcement-inspired adaptive logic to provide structured query understanding and intelligent learning recommendations.

The objective is to build a modular, explainable, and technically sound system that satisfies the following capabilities:

• Student Query Understanding
• Adaptive Learning Path Recommendation

Problem Decomposition

The problem was divided into two major components to ensure clarity and modular system design.

2.1 Q1 – Student Query Understanding

Given a natural language student query, the system must classify:

• Intent Type (Explanation, Example, Doubt Clarification, Revision)
• Topic (e.g., Backpropagation, Optimization, Neural Networks)
• Difficulty Level (Beginner / Intermediate / Advanced)

Example:

Input:
“I don’t understand backpropagation”

Output:
```
{
"intent": "Explanation",
"topic": "Backpropagation",
"difficulty_level": "Beginner"
}
```

2.2 Q2 – Adaptive Learning Path Recommendation

Based on student performance and behavior, the system must recommend:

• Next topic to study
• Whether revision is required
• Difficulty adjustment (Increase / Decrease / Same)

Inputs considered:

• Quiz score
• Number of attempts
• Time spent on topic
• Mastery level
• Current topic

Example:

Input:
```
{
"mastery": 0.3,
"score": 40,
"attempts": 3,
"time_spent": 120,
"current_topic": "Backpropagation"
}
```

Output:
```
{
"next_topic": "Backpropagation",
"action": "Revision",
"difficulty_adjustment": "Decrease"
}
```

System Architecture

The system follows a modular architecture separating NLP understanding from adaptive logic.

High-Level Flow:

Student Query
→ Sentence Embedding Layer
→ ML Classification Layer
→ Structured Output
→ Adaptive Learning Engine
→ Personalized Recommendation

This separation ensures clean ML vs reasoning logic distinction.

Dataset

No dataset was provided by the organizers. Therefore, a synthetic dataset was generated as per competition guidelines.

Dataset file:
data/synthetic_queries.csv

The dataset includes:

- 200+ student queries
- Intent labels
- Topic labels
- Difficulty labels

Dataset size is below 50MB as required.

4.1 Data Generation Method

Synthetic queries were created using template-based generation.

Example templates:

- “Explain {topic}”
- “Give an example of {topic}”
- “Why is {topic} not working?”
- “Revise {topic} concepts”

Difficulty levels were assigned based on:

- Vocabulary complexity
- Conceptual depth
- Query structure

4.2 Assumptions

Query phrasing reflects student intent

Difficulty correlates with conceptual depth

Mastery can be approximated from quiz score

Topic progression is simplified as linear

4.3 Limitations

Synthetic data may lack real-world variability

Reinforcement learning implementation is simplified

No long-term student memory tracking

LLM refinement is simulated via logic layer

Q1 – Student Query Understanding

5.1 Sentence Embeddings

Model Used:
sentence-transformers/all-MiniLM-L6-v2

This model converts each query into a 384-dimensional semantic vector representation.

Reason for selection:

• Lightweight and efficient
• Strong semantic performance
• Suitable for small-scale applications

5.2 Intent Classification

Technique:

• Supervised Machine Learning
• Logistic Regression classifier

Input: Sentence embeddings
Output: Intent label

Classes:

• Explanation
• Example
• Doubt Clarification
• Revision

5.3 Topic Classification

Supervised classification approach using embedding features.

Topics include:

- Backpropagation
- Optimization
- Neural Networks
- Gradient Descent
- Loss Functions

5.4 Difficulty Classification

Multi-class classification model trained on embedding features.

Output levels:

- Beginner
- Intermediate
- Advanced

5.5 LLM Refinement Layer

After ML predictions, a structured refinement layer formats and validates output.

Purpose:

- Maintain JSON consistency
- Demonstrate LLM-style reasoning integration
- Separate ML prediction from interpretation logic

This ensures clear ML vs LLM architectural separation.

Q2 – Adaptive Learning Path Recommendation

6.1 Knowledge Tracing

Mastery is estimated as:

Mastery = score / 100

Low mastery indicates difficulty understanding.
High mastery indicates readiness to progress.

6.2 Reinforcement-Inspired Policy

The adaptive engine uses a simplified reinforcement learning style decision policy.

State variables:

• Mastery
• Attempts
• Time spent

Actions:

• Revision
• Maintain difficulty
• Increase difficulty
• Decrease difficulty

The policy aims to:

• Reduce frustration
• Encourage gradual progression
• Improve learning stability

6.3 Hybrid Rule-Based Logic

Decision rules:

If mastery < 0.4
→ Recommend revision
→ Decrease difficulty

If 0.4 ≤ mastery ≤ 0.7
→ Maintain difficulty
→ Continue practice

If mastery > 0.7
→ Increase difficulty
→ Move to next topic

This hybrid system ensures adaptive, non-generic outputs.

Evaluation

7.1 Q1 Evaluation

Metrics:

• Accuracy
• Precision
• Recall
• F1-score

Classification reports generated during training phase.

7.2 Q2 Evaluation

Evaluation performed using:

• Scenario simulation
• Logical consistency validation
• State-action verification

Evaluation script:
evaluate_q2.py

System Design Strengths

• Clear ML vs LLM separation
• Transformer-based semantic understanding
• Modular code architecture
• Hybrid reinforcement-inspired personalization
• Fully reproducible environment using requirements.txt

Innovation

• Combination of embeddings + classical ML
• Reinforcement-inspired adaptive policy
• Knowledge tracing integration
• Structured and scalable design

Future Improvements

• Integration with real educational datasets
• Deep reinforcement learning models
• Contextual bandit personalization
• Real-time student tracking
• Advanced topic dependency graphs

Conclusion:

The AI-Powered Adaptive Teaching Assistant demonstrates an end-to-end system integrating NLP, Machine Learning, and reinforcement-inspired personalization.

The solution satisfies all competition constraints and evaluation criteria while maintaining modularity, clarity, and explainability.