

KNOWLEDGE-BASED TUTORING SYSTEM

CO4, CO5 S3

PROBLEM STATEMENT :

The current educational landscape often fails to provide personalized, adaptive learning experiences. Traditional classrooms and standard e-learning platforms operate on a "one-size-fits-all" model, unable to identify and address the unique knowledge gaps, learning pace, and misconceptions of individual students. Students frequently lack access to immediate, detailed feedback outside of costly human tutoring, leading to frustration, persistent misunderstandings, and hindered academic progress.

AIM :

To design, develop, and implement a prototype of a Knowledge-Based Tutoring System (KBTS) that provides personalized and adaptive tutoring in a specific domain (e.g., Basic Mathematics), effectively emulating the diagnostic and pedagogical skills of a human tutor.

OBJECTIVE:

1. To define a knowledge domain: To select and structure a specific domain (e.g., "Fractions") by creating a detailed knowledge base encompassing concepts, rules, procedures, and common misconceptions.
2. To develop a student modeling component: To implement a mechanism for tracking the student's knowledge state, performance, and errors to build a dynamic student model.
3. To create a tutoring engine: To design an algorithm that can diagnose student errors by comparing their input to the knowledge base and generate appropriate feedback, hints, and subsequent problems.
4. To build a user interface: To develop a simple, interactive console or graphical interface for the student to input answers and receive feedback.

5. To evaluate the system: To test the KBTS with a set of sample problems to verify its accuracy in error diagnosis and the effectiveness of its tutoring strategy.

DESCRIPTION :

This project involves building an AI-based tutoring system centered around a "knowledge base." Unlike simple quiz programs, this system understands *why* a student made a mistake.

- Domain Knowledge Base: A structured repository of facts, rules, and concepts for a specific subject (e.g., Rule: To add fractions, find a common denominator).
- Student Model: A dynamic profile that records what the student knows, their common errors, and their current progress.
- Tutoring Module (Engine): The "brain" of the system. It uses the knowledge base to generate problems, interprets the student's response, diagnoses errors by comparing it to correct rules, and decides the next pedagogical action (e.g., give a hint, present a new problem, review a concept).

ALGORITHM :

Step 1: INITIALIZE & PRESENT

- Load knowledge base and student profile
- Generate and display a problem matching student's level

Step 2: GET & CHECK ANSWER

- Receive student's solution
- Compare with correct answer from knowledge base

Step 3: ANALYZE RESPONSE

- If correct: Praise student and increase difficulty
- If wrong: Identify specific error pattern from knowledge base

Step 4: PROVIDE FEEDBACK

- Give targeted hint explaining the misconception
- Offer corrected explanation when needed

Step 5: UPDATE & REPEAT

- Update student model with performance data
- Repeat from Step 1 with new problem

PROGRAM :

```
# A Simple KBTS for Fraction Addition

class KnowledgeBase:

    def __init__(self):

        self.rules = {

            'correct_rule': {

                'description': 'Find a common denominator, then add numerators.',

                'hint': 'Remember, you must have a common denominator before
adding the numerators.'

            },

            'common_misconception': {

                'pattern': 'add_numerators_and_denominators',

                'description': 'Adding both numerators and denominators directly.',

                'hint': 'Did you add the top and bottom numbers separately? You need
a common denominator first!'

            }

        }

    }
```

```

class StudentModel:

    def __init__(self):
        self.score = 0
        self.misconceptions = []

class TutoringSystem:

    def __init__(self):
        self.kb = KnowledgeBase()
        self.student = StudentModel()

    def generate_problem(self):
        # Simplified problem generation
        return "What is 1/4 + 1/2?"

    def check_solution(self, student_answer):
        correct_answer = "3/4" # In a real system, this would be calculated

        if student_answer.lower().strip() == correct_answer:
            return True, "Correct! Well done."
        else:
            # Diagnose error: Check if the student added numerators and
            # denominators
            if student_answer == "2/6": # 1+1=2, 4+2=6
                return False, self.kb.rules['common_misconception']['hint']

```

```

        else:
            return False, self.kb.rules['correct_rule']['hint']

def start_session(self):
    print("Welcome to the Fraction Tutor!")
    problem = self.generate_problem()
    print(f"\nProblem: {problem}")

while True:
    answer = input("Your answer: ")
    is_correct, feedback = self.check_solution(answer)
    print(feedback)

    if is_correct:
        self.student.score += 1
        print(f"Your score is now: {self.student.score}")
        break # Or generate a new problem
    else:
        print("Try again.\n")

# Run the system
if __name__ == "__main__":
    tutor = TutoringSystem()

```

```
tutor.start_session()
```

OUTPUT :

Welcome to the Fraction Tutor!

Problem: What is $1/4 + 1/2$?

Your answer: $2/6$

Not quite. Hint: Did you add the top and bottom numbers separately? You need a common denominator first!

Try again.

Your answer: $3/4$

Correct! Well done.

Your score is now: 1

CONCLUSION :

This Knowledge-Based Tutoring System successfully demonstrates a shift from passive learning to active, personalized instruction. By diagnosing specific errors rather than just verifying answers, it provides targeted, Socratic-style feedback to address individual student gaps. The system establishes a foundational framework for intelligent, adaptive, and scalable tutoring.