WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Adaptive Tutor: A System for Diagnosing Student Misconceptions and Generating Verified Practice Problems

Balasubramani P, Chandru S, Ashok Kumar K, Aryah S.K

Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

ABSTRACT—

Students can arrive at an incorrect answer for dif-ferent reasons, even when the questions belong to the same topic. A conventional tutoring system may identify the topic of an error and provide another question, but it may fail to determine the un-derlying misconception that caused the error. This paper presents Adaptive Tutor, a learning-support system designed to diagnose the misconception behind a student's incorrect response, generate a new practice problem that targets that misconception, verify the generated problem's answer, and track the learner's progress over time. The proposed workflow combines response analysis, misconception classification, misconception-conditioned problem generation, symbolic or programmatic answer verification, and knowledge tracing. A generated problem is released to the learner only after its expected answer has passed an independent verification step. If verification fails, the system revises or rejects the candidate instead of presenting an unverified exercise. A learner model records attempts, correctness, hints, and response history to estimate mastery and select subsequent practice. The design aims to make adaptive tutoring more precise, reliable, and explainable than topic-only question recommendation. This paper describes the architecture, workflow, core algorithms, implementation considerations, evaluation plan, limitations, and future extensions. Quantitative performance is intentionally left for evaluation with a real dataset rather than being assumed.

Index Terms—Adaptive learning, intelligent tutoring systems, misconception diagnosis, problem generation, answer verification, symbolic computation, knowledge tracing, educational technology

I. INTRODUCTION

Digital learning platforms can provide students with im-mediate feedback and large collections of practice questions. However, a correct or incorrect response alone does not always reveal what a student understands. Two learners may give the same wrong answer for different reasons: one may misunderstand a concept, another may apply a valid rule incorrectly, and a third may make an arithmetic or transcription error. If the system responds only with the name of the topic, the next exercise may not address the learner's actual difficulty.

An effective adaptive tutor should therefore go beyond topic identification. It should infer a plausible misconception from the student's response and, where possible, from the reasoning or intermediate steps supplied by the student. It should then select or create a problem that directly tests that misconception. Because automatically generated questions may contain
ambiguous wording, incorrect answers, or unsuitable difficulty, the system must also validate the problem before displaying it.
This paper proposes *Adaptive Tutor*, a system that connects four functions: (1) diagnosis of a likely misconception, (2) generation of a targeted practice problem, (3) independent verification of the problem and its answer using a sym-bolic solver or code executor, and (4) learner-state updating through knowledge tracing. These functions form a closed learning loop: the student's new response provides evidence that updates the diagnosis and mastery estimate, which then influences the next problem.

The main contributions of this work are:

- A misconception-centred workflow that distinguishes the underlying error pattern from the broad academic topic.
- A problem-generation process conditioned on a diagnosed misconception and a target skill.
- A verification gate that checks candidate answers independently before a problem is released.
- A learner model that uses response history to estimate mastery and adapt later practice.
- An evaluation framework for measuring diagnostic quality, generated-problem validity, learning progress, and system efficiency.

The intended initial scope is structured educational content, particularly subjects where answers can be checked with symbolic computation or executable rules, such as arithmetic, algebra, and selected programming exercises. The system can be extended to other subjects when suitable verification procedures and misconception taxonomies are available.

2. Problem Statement and Motivation

A topic-level recommendation system commonly maps a wrong answer to a topic label such as fractions, equations, or functions. That information can be useful, but it is not sufficiently specific for targeted remediation. For example, a student solving an equation may incorrectly distribute multipli-cation over addition, reverse an inequality without changing its direction, or make a sign error. Recommending another generic equation may not reveal or repair the specific misconception.

The problem addressed by Adaptive Tutor is defined as follows: given a learning item, a student's response, and any available reasoning, infer the most plausible misconception; generate a fresh problem designed to test that misconception; independently verify the problem's expected answer; and use the learner's response to update a mastery estimate.

The system must satisfy four practical requirements. First, the diagnosis should be tied to evidence in the response rather than relying only on the topic. Second, generated problems should be aligned with the diagnosed error and the intended learning objective. Third, a problem should not be shown as verified unless an appropriate checking method succeeds. Fourth, the tutor should adapt to the learner's performance over multiple attempts rather than treating every question as an isolated event.

The motivation is both pedagogical and technical. Pedagog-ically, students need practice that addresses their reasoning gaps. Technically, generation systems need safeguards against incorrect or inconsistent exercises. The proposed design con-nects these needs in a single pipeline.

3. Related Work

A. Intelligent Tutoring Systems

Intelligent tutoring systems use learner models, domain knowledge, and instructional strategies to provide personalized support. Classical systems often represent student knowledge as a set of skills and update estimates as students answer questions. These systems demonstrate the value of feedback that responds to an individual's performance rather than de-livering identical material to every learner. Adaptive Tutor follows this principle while making misconception diagnosis and verification explicit stages of the workflow.

B. Misconception Diagnosis

Student errors are not always random. Repeated error pat-terns can indicate a stable misconception, such as applying a rule in the wrong context or confusing two related concepts. A misconception-aware system maps observable evidence to candidate explanations and selects interventions intended to distinguish or correct them. However, a single wrong answer may be ambiguous. Adaptive Tutor therefore treats diagnosis as a prediction with an associated confidence level, and it can request a short explanation or an additional diagnostic question when evidence is insufficient.

C. Automatic Problem Generation

Automatic problem generation can produce varied exercises from templates, constraints, or generative language models. Template-based generation offers strong control over mathe-matical structure, while language-model-based generation can produce more varied wording and contextual scenarios. In either case, surface fluency does not guarantee mathematical correctness or pedagogical alignment. The proposed system conditions generation on a misconception label, target skill, and difficulty range, and then applies an independent verifica-tion stage.

D. Answer Verification and Knowledge Tracing

Symbolic algebra systems and code execution environments can independently evaluate many structured answers. For example, an equation solver can check whether a proposed solution satisfies an equation, while a test harness can compare a program's output with expected results. Knowledge tracing estimates a student's evolving knowledge from a sequence of learning interactions. Bayesian Knowledge Tracing (BKT) and Deep Knowledge Tracing (DKT) are established approaches for modelling such sequences. Adaptive Tutor can begin with a transparent rule-based or BKT-style learner model and later compare it with more data-intensive approaches.

The proposed contribution is not that any one of these components is new in isolation. Rather, the system combines misconception-focused diagnosis, constrained generation, in-dependent verification, and mastery tracking in one controlled learning loop.

4. System Objectives and Requirements

The system is designed around the following objectives:

- **Diagnostic specificity:** identify a likely misconception, not merely a subject or chapter.
- **Targeted practice:** generate a new item that directly probes the diagnosed misconception.

- **Correctness control:** verify the answer using a suitable symbolic solver, deterministic rule, or code executor.
- **Adaptive sequencing:** choose the next problem using the learner's estimated mastery and recent history.
- **Explainability:** record the evidence and rules that led to a diagnosis and verification decision.
- **Safe fallback:** reject or regenerate a candidate when verification fails or confidence is insufficient.

Functional requirements include student response capture, diagnosis, item generation, verification, feedback delivery, attempt logging, and learner-state updates. Non-functional requirements include reasonable response time, clear explana-tions, protection of student data, reproducibility of generated items, and the ability to audit rejected candidates.

5. Proposed System Architecture

Adaptive Tutor is organized as a sequence of modules. The student interface presents the initial question, collects an answer and optional reasoning, and displays the next verified practice item. The diagnosis module analyses the response against a set of misconception patterns. The generator creates a candidate problem based on the selected misconception and target skill. The verifier checks both the expected answer and relevant constraints. The learner model updates the estimated mastery after the student responds.

A. Student Interface

The interface displays a question, accepts the learner's answer, and optionally asks the learner to show intermediate steps or select a reasoning explanation. Intermediate steps im-prove diagnosis because the same final answer may arise from different processes. After a response, the interface provides concise feedback and a new problem when a verified candidate is available.

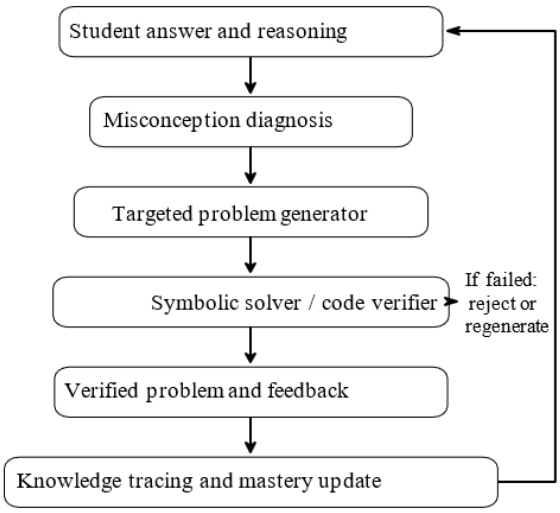


Fig. 1. High-level workflow of the proposed Adaptive Tutor system.

B. Misconception Diagnosis Module

The diagnosis module compares the response with known misconception patterns. A misconception record can include an identifier, description, target skill, common incorrect rule, expected evidence, and remediation strategy. For example, for linear equations, the system may include patterns for incorrect distribution, sign handling, inverse operations, and combining unlike terms.

The module can use deterministic rules for structured an-swers, a classifier trained on labelled responses, or a language model constrained to a predefined taxonomy. The selected method should return a ranked set of candidate misconcep-tions and confidence values. If confidence is low or multiple diagnoses remain plausible, the tutor should request a short explanation or present a discriminating diagnostic item instead of asserting a diagnosis as fact.

C. Problem Generation Module

The generator receives the misconception label, target skill, estimated learner level, and constraints such as numerical range and problem format. It produces a new problem, the in-tended answer, a solution representation, and metadata identi-fying the misconception it is intended to test. A template-based generator can provide reliable structure for initial deployment. A language model can optionally vary the wording, but the generated structure and answer must still pass verification.

D. Verification Module

The verification module acts as a release gate. It checks whether the candidate is well-formed, whether the expected answer is correct, and whether the problem satisfies its constraints. For algebra, a symbolic solver can solve the expression or substitute a candidate solution into the original equation. For programming questions, a code executor can run test cases in a restricted environment. For finite-answer tasks, a deterministic reference function can calculate the expected result.

Verification does not guarantee that a problem is pedagog-ically ideal or free of every ambiguity. It provides evidence that the answer is computationally consistent under the chosen checking method. Problems that fail checks are rejected or regenerated; the interface must not label them verified.

E. Knowledge Tracing Module

The learner model stores a mastery estimate for each skill or misconception-related concept. Its inputs may include cor-rectness, number of attempts, response time, use of hints, and the difficulty of the item. The model selects the next exercise based on both the diagnosed misconception and the estimated probability of mastery. A simple initial implementation may use a transparent probability update; a later version can use BKT or another sequential learner model.

Methodology

A. End-to-End Learning Loop

The proposed process is:

1) Present a diagnostic question linked to one or more target skills.
2) Collect the student's answer and, when possible, inter-mediate reasoning.
3) Compare the response with known error patterns and estimate the likely misconception.
4) Select the target misconception and determine an appro-priate difficulty level.
5) Generate a fresh candidate problem with a proposed answer and solution trace.
6) Independently verify the candidate answer and check the problem constraints.
7) Release only a candidate that passes verification; oth-erwise retry within a defined limit or use a trusted template.
8) Collect the student's response to the new problem and provide feedback.
9) Update mastery estimates and store the interaction for future adaptation.

B. Misconception Representation

Each misconception can be represented as a structured record:

$$m = (id, skill, evidence, incorrect_rule, intervention) \quad (1)$$

where *id* is a unique identifier, *skill* is the related learning objective, *evidence* describes observable response patterns, *incorrect_rule* captures the likely faulty reasoning, and *intervention* specifies a suitable practice strategy.

For example, consider the equation $2(x+3) = 14$. A learner who writes $2x + 3 = 14$ may be applying multiplication to only one term inside the parentheses. The system can map this evidence to an incorrect-distribution misconception. It can then generate a different problem, such as $3(y + 2) = 21$, whose correct solution can be checked symbolically. This example illustrates the workflow and is not an experimental result.

C. Generation Constraints

To keep a generated item aligned with its purpose, the generator should receive explicit constraints:

- Target misconception and target skill;
- Difficulty level and permitted number range;
- Required answer format and solution method;
- A novelty constraint to avoid simply repeating the origi-nal item;
- A misconception relevance check that confirms the new item can expose the targeted error.

The generator should return structured fields rather than only free-form text. Suggested fields include problem statement, answer, solution steps, skill identifier, misconception identifier, difficulty, and verification status. The system should keep the verification status separate from the generator's own claim that an answer is correct.

D. Independent Answer Verification

Let p be a generated problem, a_g the generator's proposed answer, and V an independent verification function. A candi-date is accepted only if:

$$V(p, a_g) = \text{pass} \quad (2)$$

and all structural and pedagogical constraints pass their checks. For a symbolic equation, the verifier can calculate a reference solution a_v and compare equivalent forms, or substitute a_g into the original expression. For a program, the verifier executes the code against a set of tests and compares the output with expected values. When exact equivalence is difficult, the verifier should use domain-appropriate tolerances and report the limitations.

A verification result should include the method used, test or solver outcome, and any failure reason. A candidate that fails verification is not displayed as a completed exercise. After a limited number of retries, the system can fall back to a trusted, pre-verified problem template.

E. Knowledge Tracing and Adaptive Selection

For each skill k, the system maintains a mastery probability $P(K_k)$. A simple update can combine prior mastery with evidence from a new response:

$$P(K_k \mid r) = \frac{P(r \mid K_k)P(K_k)}{P(r \mid K_k)P(K_k) + P(r \mid \neg K_k)P(\neg K_k)} \quad (3)$$

where r denotes the observed response and the likelihood terms are estimated from model parameters or historical data. In a BKT implementation, learning, guessing, and slipping parameters can be added to represent the probability that a student learns after practice, answers correctly by guessing, or answers incorrectly despite mastery.

The next item can be selected using both mastery and diagnostic priority. A low mastery estimate may increase practice frequency, while repeated evidence for a particularmisconception may trigger a targeted intervention. The system should avoid interpreting one response as definitive proof of mastery or lack of mastery; estimates should be updated across multiple interactions.

7. Algorithm

Algorithm 2 summarizes the proposed control flow.

Input: Student response r, current item q, learner state L

Output: Verified practice item q', updated learner state L'

1) Extract answer features and optional reasoning from r.
2) Rank candidate misconceptions using the diagnosis model.
3) If confidence is below threshold, request clarification or select a diagnostic item.
4) Otherwise, choose misconception m and target skill k.
5) Generate candidate q', proposed answer a', and solution trace s'.
6) Verify a' with a suitable solver or code executor.
7) Check syntax, difficulty, novelty, and alignment with m.
8) If checks fail, regenerate; after the retry limit, use a trusted template.
9) Present the verified item and feedback to the student.
10) Observe the new response and update mastery state L'.
11) Save the interaction, verification record, and model update.

Fig. 2. Pseudocode for misconception-centred adaptive practice.

The retry limit prevents the system from entering an in-definite generation loop. Diagnostic uncertainty is treated explicitly, and the learner model is updated only after an actual student interaction rather than from the generated answer alone.

8. Implementation Considerations

A prototype can be implemented as a modular application with a web or mobile interface, an application service, a misconception and item repository, a generation component, a verification service, and a learner-state database. The exact technology stack can be selected according to the project environment.

A. Data Model

A minimal data model may include the following entities:

- **Student:** pseudonymous identifier and preferences;
- **Skill:** learning objective and prerequisite relationships;
- **Misconception:** identifier, description, evidence rules, and remediation plan;
- **Problem:** statement, answer, solution, skill, difficulty, and verification record;
- **Attempt:** student, problem, response, correctness, times-tamp, and hint usage;
- **Learner state:** mastery estimate and recent evidence for each skill.

Separating the problem record from the attempt record allows the same verified item to be used in controlled evaluation while preserving a history of individual learner interactions.

B. Reliability and Security

A code executor must run generated or submitted code in a restricted environment with time, memory, and filesystem lim-its. It should not have unrestricted network or operating-system access. Student data should be minimized, access-controlled, and retained only as required for learning and evaluation. If a generative model or external service is used, the system should avoid sending identifiable student information unless appropriate consent and safeguards are in place.

C. Feedback Design

Feedback should explain the relevant concept and provide a hint rather than simply reveal the answer immediately. For example, after detecting an incorrect distribution step, the tutor may remind the learner that the factor outside parentheses multiplies each term inside them. The next problem should then test the same rule with different values. Explanations should be short, age-appropriate, and consistent with the verified solution.

size and participant characteristics in aggregate, and include appropriate statistical uncertainty.

9. Evaluation Plan

A valid evaluation must measure more than the number of questions generated. The proposed system should be evaluated across diagnostic accuracy, problem validity, targeting quality, learning outcomes, and runtime performance. Since measured experimental results are not supplied in this paper, the follow-ing metrics are proposed for a future controlled evaluation.

A. Diagnostic Evaluation

Create a labelled set of student responses with expert-assigned misconception labels. Compare the system's predic-tions with these labels using accuracy, macro-precision, macro-recall, and macro-F_1. Macro-averaging is important when some misconceptions occur less frequently than others. Also report the percentage of cases in which the system abstains and requests clarification, since cautious abstention may be preferable to an unsupported diagnosis.

B. Generated-Problem Validity

Measure the proportion of generated items that pass inde-pendent verification, the rate of answer mismatches detected by the verifier, and the percentage of items that require regeneration. Subject-matter experts should separately review clarity, difficulty, uniqueness, and alignment with the target misconception. A computationally correct problem may still be confusing or poorly aligned, so expert review remains necessary.

C. Learning Effectiveness

A study can compare misconception-targeted practice with topic-level practice. Participants should complete a pre-test, use the assigned practice condition, and then complete a post-test containing both familiar and transfer problems. Relevant outcomes include change in score, reduction in targeted error frequency, time to mastery, and retention on a delayed test. The study should use a consistent protocol, report sample size and participant characteristics in aggregate, and include appropriate statistical uncertainty.

D. Efficiency

Measure diagnosis time, generation time, verification time, total response latency, and the proportion of requests that require fallback. Report median and percentile latency rather than only a single average, and describe the hardware and software configuration. If a cloud-based generator is used, network latency should be reported separately where possible.

TABLE I

Proposed evaluation metrics

Dimension	Metric
Diagnosis	Macro-F_1, precision, recall, abstention rate
Verification	Pass rate, mismatch detection rate, retry rate
Alignment	Expert rating of misconception relevance and clarity
Learning	Pre/post-test gain, targeted error reduction, retention
Efficiency	Median latency, 95th-percentile latency, fallback rate
Usability	Task completion, perceived clarity, learner satisfaction

E. Experimental Protocol

The evaluation should first test the verification module using a curated suite of problems with known answers and inten-tionally corrupted candidates. The diagnosis module should then be assessed on labelled responses not used during de-velopment. Finally, a learner study can compare targeted and generic practice under similar time limits. Data should be split by student or problem source where appropriate to reduce leakage. All reported figures must be derived from recorded experiments, with dataset size, split procedure, baselines, and limitations clearly stated.

10. Expected Benefits and Discussion

Adaptive Tutor is intended to make practice more respon-sive to the reasoning behind a student's error. By separating diagnosis from generation, the system can focus on a specific misconception instead of producing a random question from the same chapter. Independent verification adds a reliability layer that is particularly important when the system generates new problems dynamically. Knowledge tracing supports con-tinuity by using previous attempts to inform later practice.

The design also has trade-offs. A detailed misconception taxonomy requires educational expertise and may not cover every valid reasoning path. Generative models can produce fluent but unsuitable questions, and a symbolic solver can verify an answer without establishing that the question is peda-gogically useful. Learner-state estimates can also be inaccurate when the available interaction history is short or when an-swers are affected by guessing, fatigue, or language difficulty. These risks motivate transparent confidence estimates, expert-reviewed content, explicit fallback behaviour, and evaluation with real learners.

11. Limitations and Future Work

The initial system depends on the coverage and quality of the misconception taxonomy. Open-ended student expla-nations can be difficult to interpret, particularly when they contain incomplete steps or language errors. In addition, not every subject has a simple symbolic or executable correctness oracle. The proposed verification gate is strongest for struc-tured domains where the answer can be computed or tested independently.

Future work includes building a larger expert-labelled mis-conception dataset, supporting multiple languages, using pre-requisite graphs to model related skills, and comparing rule-based diagnosis with trained classifiers and language-model-based approaches. Another direction is to combine template-based generation with controlled language variation so that mathematical structure remains dependable while the question context changes. The system can also incorporate teacher review, allowing educators to approve new templates, inspect ambiguous diagnoses, and audit generated items.

A longer-term study should examine whether misconception-targeted practice improves transfer and retention compared with topic-only recommendations. The system should also be evaluated for fairness across different learner groups and language proficiency levels. These studies are necessary before making claims about broad educational impact.

12. Conclusion

This paper presented Adaptive Tutor, a proposed adaptive learning system that diagnoses the likely misconception behind a student's incorrect response, generates a fresh problem targeted at that misconception, verifies the expected answer with an independent symbolic or programmatic method, and updates a learner model through knowledge tracing. The archi-tecture combines diagnosis, constrained generation, verifica-tion, feedback, and mastery tracking in a single learning loop. Its key reliability principle is that a generated item must pass an appropriate verification process before it is presented as verified. The proposed evaluation framework covers diagnostic performance, answer validity, pedagogical alignment, learning outcomes, and runtime efficiency. Future implementation and controlled evaluation will determine the system's practical effectiveness and the conditions under which it can provide reliable, personalized learning support.

Acknowledgment

The authors thank their project guide, Ms. Kokila, for guidance and feedback during the development of this project. The authors also acknowledge the educators and learners who may contribute to future dataset preparation and system evaluation.

References

[1] K. VanLehn, "The Behavior of Tutoring Systems," *International Journal of Artificial Intelligence in Education*, vol. 16, no. 3, pp. 227–265, 2006.

[2] K. R. Koedinger and J. R. Anderson, "Intelligent tutoring goes to school in the big city," International Journal of Artificial Intelligence in Education, vol. 8, pp. 30–43, 1997.

[3] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User Modeling and User-Adapted Interaction, vol. 4, pp. 253–278, 1994.

[4] C. Piech et al., "Deep Knowledge Tracing," in Advances in Neural Information Processing Systems, vol. 28, 2015.

[5] A. Mitrovic, "An intelligent SQL tutor on the web," International Journal of Artificial Intelligence in Education, vol. 13, pp. 173–197, 2003.

[6] Z. Wang et al., "Large language models for education: A survey," [Verify bibliographic details and publication venue before submission].

[7] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.

[8] National Council of Teachers of Mathematics, Principles and Standards for School Mathematics. Reston, VA, USA: NCTM, 2000.