Personalized Learning Recommendation with Knowledge Tracing and Tag-Level Mastery

Changxin Chen
Southern University of Science and Technology
Shenzhen, China
chenchangxin2022@mail.sustech.edu.cn

Weijuan Ou
Southern University of Science and Technology
Shenzhen, China
12212252@mail.sustech.edu.cn

Tianrun Qiu
Southern University of Science and Technology
Shenzhen, China
qiutr@mail.sustech.edu.cn

Yang Shu
Southern University of Science and Technology
Shenzhen, China
12211806@mail.sustech.edu.cn

Wenhui Tao
Southern University of Science and Technology
Shenzhen, China
12111744@mail.sustech.edu.cn

ABSTRACT

We present an intelligent learning recommendation system that combines sequential deep learning models and interpretable rule-based algorithms to support personalized online education. Leveraging the large-scale EdNet dataset, we begin with comprehensive data preprocessing and exploratory analysis, including statistical characterization of student behavior, question difficulty, and item discrimination. For knowledge tracing, we implement SAINT+, a Transformer-based model enhanced with temporal features such as elapsed time and lag time, enabling effective prediction of student correctness. To further assist learning, we develop a personalized question recommendation engine that adapts to individual students' strengths and weaknesses by analyzing their tag-level mastery and targeting unattempted but relevant questions based on difficulty and pedagogical value. Experimental results demonstrate that SAINT+ achieves a validation AUC of 0.5658, while the recommendation module provides interpretable and efficient suggestions. The system is deployed via a modern web interface powered by FastAPI and React, offering a responsive and interactive educational experience. Our work highlights the synergy of deep learning and human-centric heuristics for scalable, personalized learning systems.

ACM Reference Format:
Changxin Chen, Weijuan Ou, Tianrun Qiu, Yang Shu, and Wenhui Tao. 2025. Personalized Learning Recommendation with Knowledge Tracing and Tag-Level Mastery. In *Data Mining '25: SUSTech Data Mining Final 2025 Spring, June 04, 2025, Shenzhen, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Data Mining '25, June 04, 2025, Shenzhen, China
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Online education has seen rapid growth in recent years, producing vast amounts of user interaction data. Analyzing such data helps improve learning outcomes by understanding student behavior and providing personalized learning experiences. However, modeling diverse learning behaviors, tracking knowledge evolution, and generating accurate recommendations remain challenging tasks. In this paper, we explore large-scale educational datasets to analyze student behavior, model knowledge tracing, and build a personalized recommendation system. Our contributions are threefold, including: *(1)* a descriptive analysis of student interactions and question difficulty; *(2)* knowledge tracing using sequential models like LSTM and Transformer; *(3)* a recommendation system based on collaborative filtering and deep learning methods.

2 DATA AND PREPROCESSING

2.1 Dataset Overview

We utilized the Riiid! Answer Correctness Prediction dataset [4], which is a dataset derived from the original EdNet dataset [2], which is a large-scale public dataset for educational data mining, containing over 131 million interaction records from 784,309 students answering 13,169 unique questions. Each record includes user ID, question ID, timestamp, and correctness. The dataset enables temporal and personalized analysis of student learning behaviors.

The Riiid! Answer Correctness Prediction dataset further added these four parameters to all records to provide a more robust dataset for knowledge tracing tasks. The four parameters are:

- `user_answer`: (int8) the user's answer to the question, if any. Read −1 as null, for lectures.
- `answered_correctly`: (int8) if the user responded correctly. Read −1 as null, for lectures.
- `prior_question_elapsed_time`: (float32) average response time (ms) per question in the previous bundle (null for first bundle or lecture).
- `prior_question_had_explanation`: (bool) whether the user received feedback after the previous question bundle (null for first bundle or lecture).

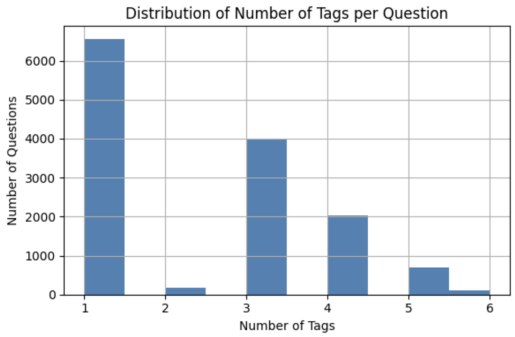

Figure 1: Distribution of question's number of tags

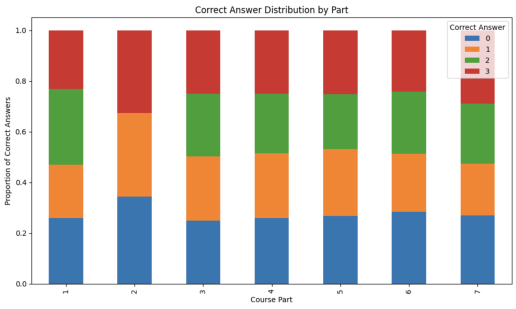

Figure 2: Distribution of answers by part

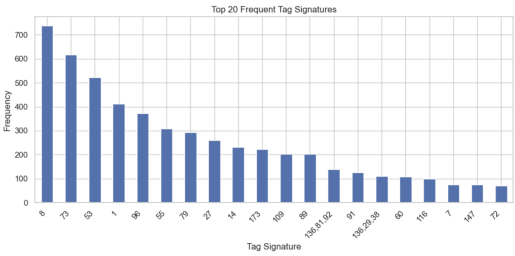

Figure 3: Frequent tags analysis

2.2 Data Descriptive Analysis

2.2.1 Question Metadata Analysis. The `questions.csv` file contains 13,523 records describing individual exercises. We conducted multi-level analysis across structure, consistency, and feature engineering.

Structural Features. We extracted key structural properties to support modeling:

- **num_tags:** Number of knowledge tags per question, capturing topical breadth. Most questions had 2–4 tags.
- **is_single_tag:** Boolean flag indicating whether a question was labeled with only one tag.
- **avg_tags_per_part:** We computed average number of tags per TOEIC part; Parts 6 and 7 had slightly higher tag density.

To support further exploration, we also analyzed the distribution and frequency of answers and tags, as shown in Figure. 1, 2 and 3.

Consistency Checks. We examined data quality and potential biases:

- **Missing tags:** Only one question lacked tag information, indicating a near-complete tag coverage.
- **Bundle size:** Questions were grouped by `bundle_id`; bundle sizes varied from 1 to over 10.
- **Correct answer distribution:** We analyzed answer label distributions per TOEIC part and observed no extreme class imbalance.

Feature Engineering. We created derived features for modeling and recommendation:

- **tag_signature:** Canonical string representation of sorted tags per question, e.g., `12,35,77`.
- **tag_signature_freq:** Frequency of each tag signature, indicating common question patterns.
- **tag_signature_hash:** Optional hashed representation for embedding use.

This analysis provided a foundational understanding of item diversity, structural regularity, and categorical features for subsequent modeling tasks.

2.2.2 Interaction Log Analysis. The `train.csv` file contains the complete student interaction logs. It includes both question-answering and lecture-viewing events, with over 100 million records and 10 core fields.

Feature Overview. Key fields include:

- **user_id, content_id:** Used to track individual user-question interactions.
- **content_type_id:** Binary indicator (0 for question, 1 for lecture).
- **answered_correctly:** Target variable (1 if correct, 0 if incorrect, -1 for lecture).
- **prior_question_elapsed_time, had_explanation:** Context from previous batch, modeling feedback effects.
- **time_since_last:** Temporal data to indicate the time between user actions.

Content Type Distribution. We found that the majority of records (~96%) are question interactions. Lectures are sparse and auxiliary in nature, reinforcing their secondary role in student activity modeling. Therefore, we continue to analyze mainly for question activities.

Behavioral Effects of Explanations. We further grouped records by *prior_question_had_explanation* to observe a modest increase in correct answer rate when prior questions were accompanied by explanations, indicating some learning benefit (0.5007 when `false` and 0.6731 when vice versa).

Batch-wise Lecture Impact. We aggregated statistics by *task_container_id* to evaluate the relationship between the proportion of lecture content and question accuracy. While some lecture-rich batches showed high accuracy, the overall trend suggested limited direct benefit, hinting at complex user behavior or reverse causality.

User Activity Analysis. To analyze student interaction logs, we must also focus on the perspective of users.

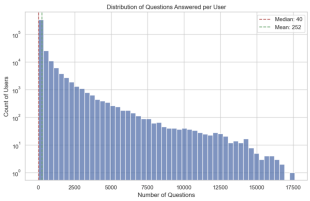

Figure 4: Distribution of questions answered per user

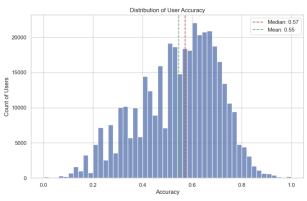

Figure 5: Distribution of user accuracy

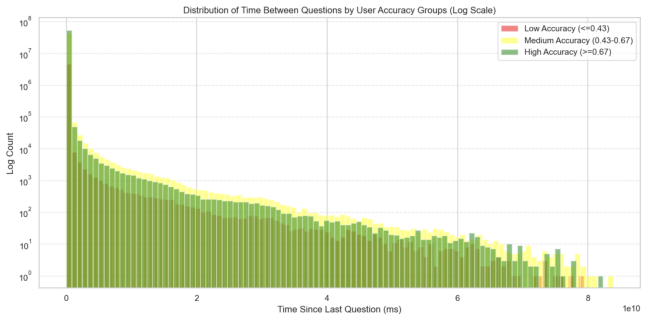

Figure 6: Distribution of time between questions

By analyzing the number of answered questions per user, we observed a long-tailed distribution, which is significant even using log axis in Figure. 4. This insight is useful for identifying cold-start users and setting thresholds for filtering or weighting in downstream modeling.

We also found that there was significant variation in users' accuracy rates, which could be further categorized into low-accuracy, medium-accuracy, and high-accuracy groups in Figure. 5. However, the three groups have similar distribution of time between questions, showing that this long-tail pattern is universally appliable, as shown in Figure. 6.

Item Difficulty and Discrimination. To evaluate item quality, we utilized classical test theory [5]. Each question is characterized by two psychometric properties:

Difficulty (P). is defined as the proportion of correct responses for an item:

$$P = \frac{R}{N} \tag{1}$$

where R is the number of correct responses and N is the number of total attempts. A range of $P \in [0.3, 0.7]$ is considered optimal for moderate difficulty [6].

Discrimination (D). measures how well an item separates strong and weak students. We computed:

$$D = P_{\text{high}} - P_{\text{low}} \tag{2}$$

where P_{high} and P_{low} are the average correctness rates among the top 27% and bottom 27% of students, respectively.

Item Quality Categorization. Each item was annotated with a `quality_flag` based on these two metrics:

- **Recommended:** $P \in [0.3, 0.7]$ and $D > 0.3$
- **Needs Review:** All others

This labeling helped prioritize reliable questions for model input and personalized recommendation.

This preprocessing ensures that the selected questions are both appropriately challenging and informative for learning analytics and personalized recommendation.

3 METHODOLOGY

Our methodology integrates multiple components to comprehensively model student learning behaviors and enhance personalized learning experiences. We first analyze knowledge states using neural knowledge tracing models such as SAINT+, which leverage temporal and contextual information. Then, to provide targeted learning interventions, we build a rule-based recommendation system that incorporates user mastery on different knowledge tags, question difficulty, and benchmark tag preferences. This hybrid pipeline allows both predictive modeling and interpretable, personalized content delivery.

3.1 Knowledge Tracing

We formally define each learning interaction I_t as a comprehensive record containing:

$$I_t = \langle E_t, R_t \rangle \tag{3}$$

Exercise Information (E_t)

- **Content**: The content includes the specific question or problem along with all its components.
- **Knowledge Components**: The knowledge components consist of a primary knowledge tag (e.g., "Quadratic Equations") and optional secondary tags (e.g., "Factoring Methods") for finer categorization.
- **Difficulty Level**: The difficulty level is represented by a pre-calibrated score ranging from 0 to 1, where higher values indicate greater complexity.
- **Exercise Type**: The exercise type specifies the format of the task, such as multiple choice, free response, or interactive simulation.

Response Information (R_t)

- **Correctness**:

$$c_t = \begin{cases} 1 & \text{if answered correctly} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- **Response Process**: The response process includes the attempt duration Δt (measured in seconds), records of intermediate steps (particularly for open-ended questions), and revision history when applicable.
- **Temporal Context**: The temporal context comprises the timestamp T_t of each attempt, the time interval $\delta_t = T_t - T_{t-1}$ since the last interaction, and a session identifier s_t for tracking purposes.
- **Confidence Indicators**: Confidence indicators contain both self-reported confidence levels $\omega_t \in [0, 1]$ and measurable answer certainty features such as hesitation patterns in responses.

Knowledge Tracing Objective The model estimates the probability of correct response given history:

$$P\left[c_t = 1 \mid \mathcal{H}_{t-1}, E_t\right] \tag{5}$$

where $\mathcal{H}_{t-1} = (I_1, ..., I_{t-1})$ represents the complete interaction history up to time $t-1$.

3.2 SAINT+

3.2.1 SAINT: Separated Self-Attentive Neural Knowledge Tracing. We used SAINT+, a knowledge tracing model based on the Transformer [8] architecture. The model consists of two main components:

Encoder

- **Input:** Takes a sequence of exercise embeddings that $\mathbf{E}^e = [E_1^e, E_2^e, \ldots, E_T^e]$
- **Operation:** Processes the sequence through multiple layers of:
  - Self-attention mechanisms (multi-head attention)
  - Position-wise feed-forward networks
  - Layer normalization and residual connections
- **Output:** Produces a contextualized exercise representation sequence $\mathbf{O}^{enc} = [O_1^{enc}, O_2^{enc}, \ldots, O_T^{enc}]$

Decoder

- **Inputs:**
  - Receives the encoder's output $\mathbf{O}^{enc}$
  - Takes a shifted response embedding sequence that $\mathbf{R}^e = [S^e, R_1^e, R_2^e, \ldots, R_{T-1}^e]$ (where S^e is a start token)
- **Operation:** For each timestep t:
  - Performs self-attention on the response sequence
  - Computes encoder-decoder attention using $\mathbf{O}^{enc}$ as keys / values
  - Processes through feed-forward networks with normalization
- **Output:** Generates a sequence $\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_T]$, where each $\hat{c}_t$ represents:

$$\hat{c}_t = P(\text{correct response to } E_t \mid E_t, I_1, \ldots, I_{t-1})$$

The model's core attention mechanism uses multi-head attention [8]:

$$\text{Head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad \text{for each } 1 \le i \le h$$
$$\text{Output} = \text{Concat}(\text{Head}_1, \ldots, \text{Head}_h)\mathbf{W}^O$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are weight matrices of query, key and value respectively, as in SAINT [7].

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$
$$\text{where head}_i = \text{Softmax}\left(\text{Mask}\left(\frac{Q_i K_i^T}{\sqrt{d}}\right)\right)V_i$$

where:

- d is the dimension of the query and key vectors
- W^O is the output projection weight matrix
- W_i^Q, W_i^K, W_i^V are projection matrices for each head i as in SAINT+ [7]

The masking mechanism is critical for ensuring temporal causality in knowledge tracing:

- **Implementation**: Overwrites elements above the diagonal of $Q_i K_i^T$ with $-\infty$
- **Effect**: Makes corresponding softmax outputs zero.
- **Purpose**: Prevents current position from attending to future positions [8].

Encoder Architecture

The encoder consists of N identical layers, each containing:

$$L_X = \text{LayerNorm}(X)$$
$$M = X + \text{Multihead}(L_X, L_X, L_X)$$
$$O = M + \text{FFN}(\text{LayerNorm}(M))$$

where FFN is a feedforward network: $\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$, layer normalization [1] stabilizes training and connection skipping [3] preserve gradient flow. The Multihead Transformer is capable of providing more accurate perception over different dimensions with different characteristics.

Decoder Architecture

Each decoder layer performs three key operations:

$$L_X = \text{LayerNorm}(X)$$
$$M_1 = X + \text{Multihead}_{\text{masked}}(L_X, L_X, L_X)$$
$$M_2 = M_1 + \text{Multihead}(\text{LayerNorm}(M_1), O^{enc}, O^{enc})$$
$$O = M_2 + \text{FFN}(\text{LayerNorm}(M_2))$$

Key features:

- **Masked self-attention**: Enforces temporal causality
- **Encoder-decoder attention**: Incorporates exercise context
- Same FFN structure as encoder

The output of the final decoder layer is passed to a fully connected layer to produce the final prediction sequence $\hat{c}_t$.

3.2.2 Enhancing Knowledge Tracing with Temporal Feature Embeddings. SAINT+ processes two parallel embedding sequences as input:

- **Exercise Embeddings**:

  $$E_t^e = \text{Embed}_{\text{ID}}(e_t) + \text{Embed}_{\text{Category}}(c_t) + \text{Embed}_{\text{Pos}}(t)$$

  where: Embed_{ID} means embedding of unique exercise identifier, $\text{Embed}_{\text{Category}}$: means knowledge component embedding, and $\text{Embed}_{\text{Pos}}$ means positional encoding.
- **Response Embeddings** (which are augmented with temporal features):

  $$R_t^e = \text{Embed}_{\text{Correct}}(r_t) + \text{Embed}_{\text{Pos}}(t) + \text{Embed}_{\text{Elapsed}}(\Delta t_{\text{resp}}) + \text{Embed}_{\text{Lag}}(\Delta t_{\text{lag}})$$

SAINT+ incorporates two critical temporal dimensions (illustrated in Figure 7):

- **Elapsed Time** (Δt_{resp}):

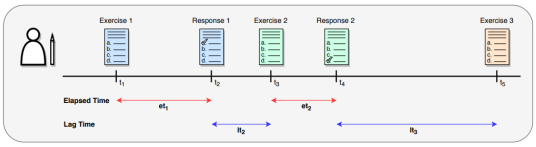

Figure 7: Visualization of elapsed time (response duration) and lag time (interval between consecutive responses) in student interaction sequences.

  - Duration between exercise presentation and student response
  - Short correct responses may indicate fluency
  - Implemented via bucketed embedding: $\text{Embed}_{\text{Elapsed}} : \mathbb{R}^+ \to \mathbb{R}^d$
- **Lag Time** (Δt_{lag}):
  - Interval between current and previous interaction
  - Models forgetting curves and spacing effects
  - $\text{Embed}_{\text{Lag}}$ transforms continuous time into discrete bins

Key features include: (1) shared embedding dimension d; (2) log-binned time features $\text{bin}(t) = \lfloor \log_{10}(t+1) \rfloor$; (3) Transformer-style positional encodings [8]; and (4) full architecture in Figure 8 [7].

Elapsed Time. The elapsed time (et) measures the duration between exercise presentation and student response:

$$et_i = t^{(i)}_{\text{response}} - t^{(i)}_{\text{presentation}} \tag{6}$$

- **Educational Significance**:
  - Short et with correct answers indicates mastery [7]
  - Prolonged et suggests knowledge gaps or uncertainty
  - Helps distinguish fluent recall from random guessing
- **Embedding Methods**:
  - **Continuous**: $\mathbf{v}_{et} = et \cdot \mathbf{w}_{et}, \mathbf{w}_{et} \in \mathbb{R}^d$, where d is the embedding dimension
  - **Categorical**: Time durations are classified into integer-second bins (1-300s) with learned embeddings, and values beyond 300s are clipped

Lag Time. The lag time (lt) captures intervals between consecutive interactions:

$$lt_i = t^{(i)}_{start} - t^{(i-1)}_{response} \tag{7}$$

- **Cognitive Basis**:
  - Models the Ebbinghaus forgetting curve
  - Short intervals strengthen memory retention
  - Long intervals correlate with knowledge decay
- **Embedding Approaches**:
  - **Continuous**: $\mathbf{v}_{lt} = lt \cdot \mathbf{w}_{lt}, \quad \mathbf{w}_{lt} \in \mathbb{R}^d$
  - **Categorical**:
    * 150 logarithmic bins (0-1440 minutes)
    * Special handling for first interaction ($lt = 0$)

3.3 Personalized Tag-Based Recommendation

In addition to sequence-based knowledge tracing, we design a lightweight, interpretable recommendation engine based on students' tag-wise mastery profiles. This engine operates independently of time-series models, making it easier to deploy and scale. It relies solely on students' past interactions and the semantic structure of the questions (i.e., tag annotations and difficulty), while incorporating optional benchmark tags to guide learning focus.

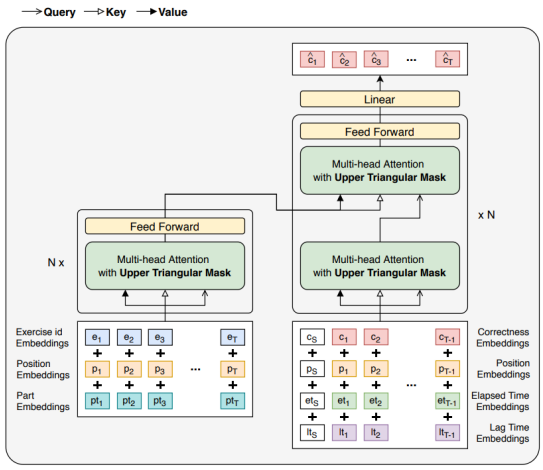

Figure 8: SAINT+ model architecture with temporal feature embeddings [7].

Tag Mastery Estimation. We define a tag-wise mastery score for each student to reflect their conceptual understanding across different topics. Let C_t and I_t denote the number of correct and incorrect responses the student has given to tag t, respectively. To avoid bias from low-frequency interactions and prevent overconfidence on rarely attempted tags, we adopt a Bayesian-smoothed estimation:

$$M_t = \frac{1 + C_t}{2 + C_t + I_t}$$

This formulation guarantees the mastery score $M_t \in (0, 1)$, and is especially robust when data is sparse. A score closer to 1 indicates stronger proficiency in the associated tag.

Candidate Question Filtering. Given the user's past interaction history, we extract a set of candidate questions by removing those already attempted. This is done by checking whether a question's `content_id` appears in the training logs for the user, considering only actual question attempts (i.e., `content_type_id = 0`).

Scoring Candidate Questions. For each unattempted question q, we compute a personalized score that reflects both the user's knowledge gaps and the question's properties. Let T_q be the tag set for question q, and D_q its difficulty score. We define the recommendation score as:

$$\text{score}(q) = \frac{\sum_{t \in T_q} (1 - M_t)}{1 + D_q} + \beta \cdot \mathbb{1}_{\text{recommended}} + 2 \cdot |T_q \cap B|$$

The formulation includes: $1 - M_t$ (lack of mastery for tag t); question difficulty D_q (higher = harder); $\mathbb{1}_{\text{recommended}}$ (1 if marked as high-quality); boost coefficient $\beta = 1.2$; and user-prioritized benchmark tags B.

This design ensures that questions targeting the learner's weakest areas (low M_t) are prioritized, while also favoring easier questions (through division by $1 + D_q$) to avoid demotivating the user with overly difficult content. Questions labeled as *recommended* receive a moderate bonus, and further priority is given to those overlapping with the benchmark tags.

Interpretability and Control. Unlike deep learning models, this rule-based recommendation engine is fully transparent and allows direct control over its behavior through benchmark tags. Educators or learners can guide the system's focus, e.g., by emphasizing certain curriculum topics or remediation areas.

Two Variants. We implement two tag-based variants for comparison:

(1) **Basic version:** Computes tag-level overlaps between previously correct/wrong answers and candidate questions. Scores are weighted accordingly to promote content similar to incorrect attempts and suppress those similar to correctly answered ones.
(2) **Advanced version:** Uses mastery scores and question difficulty to compute a more refined and personalized score. This version supports benchmark tag boosts and integrates a recommendation-quality flag.

Both approaches offer interpretable, tag-aware recommendations and complete within milliseconds, making them ideal for real-time feedback in educational applications. Further, low-discrimination or noisy items can be excluded to enhance the overall quality of recommendations.

4 EXPERIMENTS AND RESULTS

4.1 System Implementation

To support intelligent educational services, we designed and implemented a modular learning recommendation system composed of three tightly integrated components: a deep learning-based knowledge tracing module, a personalized recommendation engine, and a modern web-based user interface, which is shown in Fig 9.

4.1.1 Backend Design. The backend system is built in Python using the FastAPI [1] framework, offering RESTful endpoints for prediction and recommendation tasks. It interfaces with preprocessed educational data and trained model checkpoints to serve inference requests.

At the core of the backend is a deep knowledge tracing module based on the SAINT+ architecture, implemented using PyTorch Lightning [2]. The model incorporates exercise ID, category, position, elapsed time, and lag time as input features, and outputs the probability that a student will correctly answer the next question. The training and evaluation routines are designed for modular experimentation and reproducibility.

4.1.2 Recommendation Logic. We designed a standalone recommendation engine to complement deep modeling with interpretable logic. The engine operates on tag-level mastery, which is dynamically computed from a student's interaction history. It also incorporates item metadata such as question difficulty, quality labels, and benchmark tag overlap. We incorporated the thinkings of the two variants of recommendation algorithms mentioned in Section 3.3 to provide robust recommendation results.

These components are exposed through a unified API, making them easily accessible from the frontend or external systems.

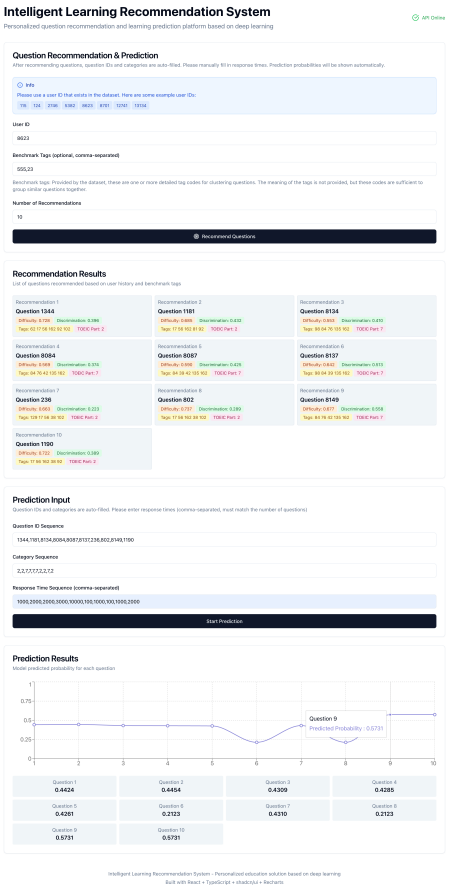

Figure 9: System implementation overview.

4.1.3 Frontend Architecture. The frontend is developed using React [3] and TypeScript [4] with Tailwind CSS [5] and ShadCN UI [6] component library. It serves as the interaction layer, allowing users to query predictions or request personalized recommendations. Data visualizations, such as predicted probabilities and recommended items, are rendered using Recharts [7] for immediate feedback and analysis.

4.1.4 System Integration. The entire system is modular by design. The backend and frontend communicate via JSON over HTTP.

[1] https://fastapi.tiangolo.com/
[2] https://lightning.ai/docs/pytorch/
[3] https://react.dev/
[4] https://www.typescriptlang.org/
[5] https://tailwindcss.com/
[6] https://ui.shadcn.com/
[7] https://recharts.org/

Model training, inference, and recommendation are encapsulated to allow for independent testing and deployment. Logs and model artifacts are stored in designated directories, supporting both offline analysis and live serving.

4.2 Predict Result

4.2.1 Evaluation Setup. We evaluate our models on a processed subset of the EdNet dataset. The student interaction sequences are chronologically split into training and validation sets. Evaluation is conducted in a full-batch manner using all available validation samples (`val_samples`), without truncation or early stopping. The primary metric used is the Area Under the Receiver Operating Characteristic Curve (AUC), which measures the model's ability to distinguish between correct and incorrect student responses.

4.2.2 Knowledge Tracing Performance. We trained the SAINT+ model with position, content, and temporal feature embeddings for binary correctness prediction. The model achieved an evaluation AUC of **0.5658** and on the validation set (note that this dataset is different from the original EdNet dataset). While not high, this score aligns with previously reported results on real-world, large-scale educational datasets, where label noise and concept sparsity present significant modeling challenges.

The result demonstrates that SAINT+ can effectively capture temporal dependencies such as elapsed time and lag time, but further improvements may require more advanced architectures, richer feature sets, or fine-tuned pretraining strategies.

4.2.3 Recommendation System Case Study. We implemented a rule-based recommendation system that scores unattempted questions based on tag-level mastery and question difficulty. For a test user with weaknesses in benchmark tags (e.g., tags 51 and 131), the system successfully retrieved 10 unseen questions that emphasize weak tags while adjusting for difficulty and metadata flags.

This approach, while simple, offers transparent logic, domain relevance, and real-time responsiveness, making it suitable for practical use in online learning platforms.

4.3 Summary and Insights

The SAINT+ model provides a flexible framework for knowledge tracing using attention and temporal features. However, its predictive power remains moderate on noisy real-world data. The recommendation module complements this by offering interpretable and adaptive content selection based on student mastery, illustrating the potential of hybrid modeling in educational data mining.

Through this project, we have also gained some insights into learning prediction and knowledge tracing tasks, which will guide future practical applications. We recognize the importance of temporal data in knowledge tracing tasks and observe that filtering out questions with lower discrimination scores can further enhance the effectiveness of recommendation algorithms. Additionally, we have verified the efficacy of Transformer models in this task. While acknowledging the current limitations in task performance, we anticipate future improvements through the development of superior models.

5 CONCLUSION

In this work, we presented an intelligent learning recommendation system that integrates deep knowledge tracing with interpretable, tag-based personalized recommendation. Leveraging the SAINT+ model, we captured temporal and contextual features in student interaction sequences to predict future performance. Additionally, we designed a lightweight recommendation engine that ranks unattempted questions based on tag-level mastery, difficulty, and educational value.

Our experimental results demonstrate that SAINT+ can model temporal dynamics in learning behavior, achieving a validation AUC of 0.5658. The rule-based recommendation system complements this by providing fast, transparent, and personalized question suggestions, which are suitable for real-world deployment. An usable system is also implemented to showcase our efforts. We hope these insights can be practically implemented in real-world educational settings in the future.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Yeonjeong Choi, Seewoo Lee, Dongmin Shin, Alice Oh, and Neil Heffernan. 2020. EdNet: A Large-Scale Hierarchical Dataset in Education. *arXiv preprint arXiv:1912.03072* (2020).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[4] Addison Howard, bskim90, Cheehyun Lee, DongminShin(Min), Hoon Pyo (Tim) Jeon, Jineon (Jin) Baek, Karis Chang, kiyoonkay, NHeffernan, seonwooko, Sohier Dane, and Yohan Lee. 2020. Riiid Answer Correctness Prediction. https://kaggle.com/competitions/riiid-test-answer-prediction. Kaggle.

[5] Frederic M. Lord. 1980. *Applications of Item Response Theory to Practical Testing Problems.* Routledge.

[6] Richa Sharma and A. P. Singh. 2022. Psychometric item analysis for e-learning assessment using classical test theory. *Education and Information Technologies* 27, 5 (2022), 6987–7005.

[7] Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. 2021. SAINT+: Integrating Temporal Features for EdNet Correctness Prediction. In *Proceedings of the 11th International Conference on Learning Analytics & Knowledge (LAK21)*. ACM, 485–496. https://doi.org/10.1145/3448139.3448188

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.