LANGUAGE BOTTLENECK MODELS: A FRAMEWORK FOR QUALITATIVE COGNITIVE DIAGNOSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurately assessing student knowledge is central to education. Cognitive Diagnosis (CD) models estimate student proficiency, while Knowledge Tracing (KT) methods excel at predicting performance over time. However, CD models represent knowledge concepts via quantitative estimates on predefined concepts, limiting expressivity, while KT methods often prioritize accuracy at the cost of interpretability. We propose Language Bottleneck Models (LBMs), a general framework for producing textual knowledge state summaries that retain predictive power. LBMs use an encoder LLM to produce minimal textual descriptions of a student's knowledge state, which a decoder LLM then uses to reconstruct past responses and predict future performance. This natural-language bottleneck yields human-interpretable summaries that go beyond the quantitative outputs of CD models and capture nuances like misconceptions. Experiments show zero-shot LBMs rival state-of-the-art CD and KT accuracy on synthetic arithmetic benchmarks and real-world datasets (Eedi and XES3G5M). We also show the encoder can be finetuned with reinforcement learning, using prediction accuracy as reward, to improve summary quality. Beyond matching predictive performance, LBMs reveal qualitative insights into student understanding that quantitative approaches cannot capture, showing the value of flexible textual representations for educational assessment.

1 INTRODUCTION

Knowledge State Modeling A fundamental objective in education is accurately assessing what a student knows, identifying misconceptions, and understanding how their knowledge evolves over time (Posner et al., 1982; Larkin, 2012; Chen et al., 2020). Teachers intuitively achieve this through diagnostic reasoning: by observing students' answers, they infer not merely correctness, but deeper patterns reflecting conceptual mastery or specific misunderstandings.

Limitations of Cognitive Diagnosis and Knowledge Tracing Cognitive Diagnosis (CD) (Templin et al., 2010; Wang et al., 2024) models formalize this by producing diagnostic reports but are often limited by rigid, predefined knowledge concept taxonomies. In parallel, Knowledge Tracing (KT) (Corbett & Anderson, 1994; Shen et al., 2024) models excel at predicting future performance based on observed past responses, but often at the cost of interpretability, representing knowledge as opaque, high-dimensional vectors.

Limitations of Existing LLM-based approaches Recent approaches leveraging large language models (LLMs) have shown that LLMs can produce sensible predictions of students' future behavior when provided with relevant information (Li et al., 2024a; Kim et al., 2024), help mitigate the KT's cold-start problem (Lee et al., 2024) and be finetuned to improve accuracy on KT tasks (Wang et al., 2025). Nonetheless, these LLM-based methods still lack rigorous interpretability, as they either treat the model as a black-box or rely on free-form explanations that are susceptible to hallucination (Bender et al., 2021). From the standpoint of Cognitive Diagnosis, such methods fail to provide grounded, reliable representations of knowledge that can be trusted in educational practice. This gap motivates a principled approach where interpretability is an intrinsic design constraint.

Knowledge State Modeling as an Inverse Problem Assessing student knowledge can be framed as an *inverse problem* (Figure 1A): observable answers are generated by an underlying knowledge state

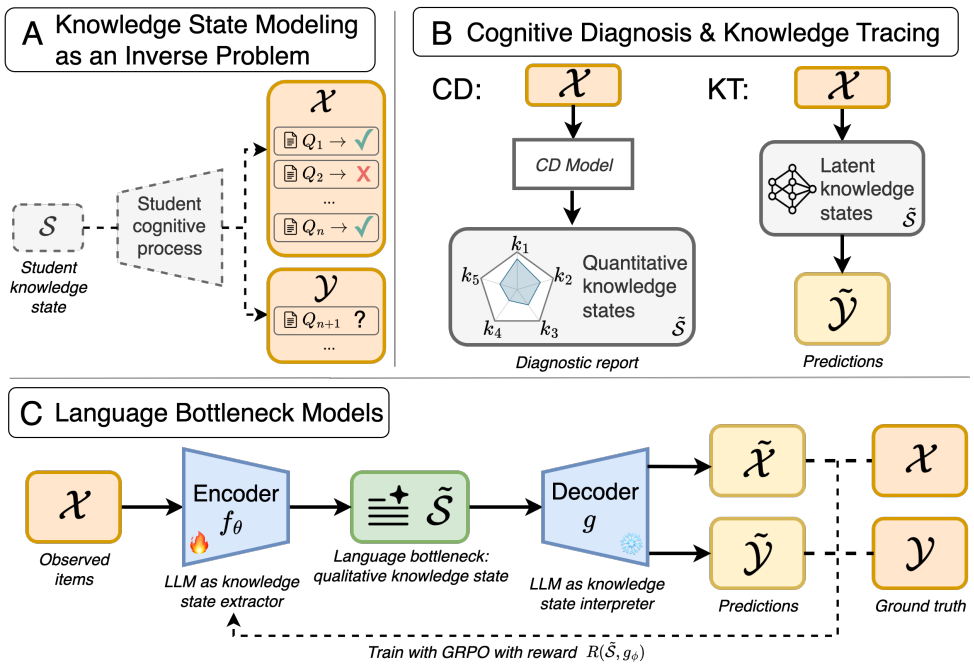

Figure 1: **Language Bottleneck Models for Knowledge Modeling.** (A) Past and future behavior $\mathcal{X}$ and $\mathcal{Y}$ are caused by a certain knowledge state $\mathcal{S}$ held by the student when answering questions. (B) CD and KT models represent the knowledge state via quantitative proficiency vectors or opaque latent embeddings. (C) LBMs approximate the knowledge state using natural language summaries which are used to predict past and future behavior.

$\mathcal{S}$, and the goal is to infer a faithful approximation $\tilde{\mathcal{S}}$. CD models constrain $\tilde{\mathcal{S}}$ to fixed quantitative proficiency estimates over knowledge concepts (Figure 1B), providing interpretability but lacking flexibility to capture nuanced knowledge states. KT models prioritize predictive performance but have uninterpretable latent representations (Figure 1B). We adopt the inverse-problem framing but allow $\tilde{\mathcal{S}}$ to be expressed as concise, free-form natural language, so knowledge states remain predictive while avoiding rigid concept taxonomies.

Language Bottleneck Models We introduce *Language Bottleneck Models (LBMs)* as a general framework for compressing a student's interaction history into a predictive, text-based knowledge state. As shown in Figure 1C, an encoder LLM maps a student's interaction history $\mathcal{X}$ into a natural language summary $\tilde{\mathcal{S}}$, while a frozen decoder LLM must reconstruct past responses and predict future ones using only that summary. In an educational settings, they constitute a language-based approach to Cognitive Diagnosis, not bound to fixed skill vocabularies and can flexibly describe nuanced insights such as misconceptions.

Contributions.

- We cast knowledge state modeling as an inverse problem—building on ideas from Cognitive Diagnosis but replacing rigid concept proficiencies with a flexible natural-language knowledge state.
- To instantiate this, we introduce Language Bottleneck Models (LBMs), which encode observed student behavior into predictive text-based summaries of knowledge state summaries.
- We extensively evaluate LBMs on synthetic and real-world datasets against 14 KT and CD baselines across 7 open- and closed-source LLM backbones, present a detailed case study of qualitative differences with CD knowledge states, and demonstrate that LBMs can be effectively trained and steered via their textual summaries.

2 KNOWLEDGE STATE MODELING AS AN INVERSE PROBLEM

2.1 PRELIMINARIES AND NOTATION

We consider data consisting of student interactions with educational questions. At each time step t, a student is presented with a question $q_t \in \mathcal{Q}$ and optionally knowledge concept information $k_t \in \mathcal{K}$ and provides a response $r_t \in \mathcal{R}$, which is evaluated for correctness $c_t \in \{0, 1\}$. We represent an interaction as $x_t = (q_t, k_t, r_t, c_t)$, and a student's interaction history up to time t as $H_t = (x_1, \ldots, x_t)$.

The standard predictive task, often referred to as *Knowledge Tracing (KT)* (Corbett & Anderson, 1994; Shen et al., 2024), is to estimate $p(c_{t+1} \mid q_{t+1}, H_t)$, the probability that the student will answer a new question q_{t+1} correctly, conditioned on their interaction history. This task definition captures the forward-prediction aspect of knowledge state modeling, but does not yet address how the underlying knowledge that drives these responses should be represented.

2.2 KNOWLEDGE STATE MODELING AS AN INVERSE PROBLEM

An alternative view is to frame knowledge state modeling as an *inverse problem*: observed responses are generated by a latent knowledge state $\mathcal{S}$ through the student's cognitive process, and the goal is to recover an approximation $\tilde{\mathcal{S}}$ of this state from the observed responses.

This perspective is central to *Cognitive Diagnosis (CD)* models (Reckase, 2006; De La Torre, 2009; Templin et al., 2010; Wang et al., 2022; 2024), which produce diagnostic reports of concept mastery from observed responses. However, CD typically restricts $\tilde{\mathcal{S}}$ to quantitative mastery or proficiency vectors based on predefined or inferred concepts, limiting expressivity. Meanwhile, deep learning-based KT methods prioritize predictive accuracy without explicitly recovering $\tilde{\mathcal{S}}$, representing knowledge states as high-dimensional embeddings that lack transparency (Piech et al., 2015; Zhang et al., 2017; Pandey & Karypis, 2019; Ghosh et al., 2020). Recent KT extensions introduce diagnostic reports similar in spirit to CD, but these remain bound to rigid concept taxonomies or rely on post-hoc interpretability (Minn et al., 2022; Chen et al., 2023; Park et al., 2024).

Key assumption: constant knowledge state A practical assumption underlying this formulation is that a knowledge state can be treated as constant within short diagnostic windows (e.g., unit tests, placement exams, or tutoring sessions). This aligns with CD models (see §2.1 in Wang et al. (2024)), and distinguishes them from KT approaches which model the evolution of $\mathcal{S}_t$ across longer periods where the underlying knowledge state is expected to change.

2.3 NATURAL LANGUAGE AS THE INTERFACE

Our formulation follows CD approaches in adopting the inverse problem framing of inferring a diagnostic report from observed responses, but instead of restricting $\tilde{\mathcal{S}}$ to quantitative mastery scores, we model it through concise textual summaries. Natural language provides an interpretable and expressive medium—capable of describing arbitrary reasoning patterns or misconceptions, a key focus in education research (Smith III et al., 1994; Wang et al., 2020; King et al., 2024)—while remaining human-understandable. In the next section, we introduce *Language Bottleneck Models (LBMs)*, which operationalizes this idea by compressing student interaction histories into concise textual representations that preserve predictive information.

3 LANGUAGE BOTTLENECK MODELS

3.1 FORMAL DEFINITION

We propose *Language Bottleneck Models* (LBMs) for Knowledge State Modeling via textual summaries: an LLM-based, two-stage architecture designed to infer a predictive text-based knowledge state from a student's interaction history.

Let $X^{\text{enc}} \subseteq H_t$ denote a subset of observed interactions used by the encoder. An encoder LLM f_θ maps this history to a natural-language summary:

$$\tilde{\mathcal{S}} = f_\theta(X^{\text{enc}}).$$

This summary serves as the sole representation of the student's knowledge state.

A decoder LLM g_ϕ then conditions only on $\tilde{\mathcal{S}}$ to predict the probability that the student will answer a question $q \in \mathcal{Q}$ correctly: $g_\phi(q, \tilde{\mathcal{S}}) = p(c \mid q, \tilde{\mathcal{S}})$, that is, given a question q and a summary $\tilde{\mathcal{S}}$, the decoder predicts the probability that the student will answer correctly.

In principle, both encoder f_θ and decoder g_ϕ could be trained. However, we show with the following motivating experiments that decoding is not the hard part: when given high-quality knowledge state summaries, off-the-shelf LLMs can achieve near-perfect prediction accuracy. For this reason, in this work we keep g_ϕ fixed and focus on learning an encoder that produces faithful, predictive, and interpretable summaries.

3.2 Motivating Observations

We motivate the design of LBMs by two observations.

Observation 1: Given a good knowledge state summary, strong LLMs can decode with high fidelity. To test this in an idealized setting, we used a synthetic dataset where each student's knowledge state was programmatically generated. Figure 2 evaluates the performance of different decoder models when given direct access to this perfect, "ground-truth" summary of the student's latent knowledge (an example is shown in Figure A1 in the Appendix, and full dataset details are in Section 5). Stronger models like GPT-4o achieve nearly perfect accuracy (98%), indicating that the bottleneck representation is indeed sufficient to drive effective downstream prediction—provided it captures the right information.

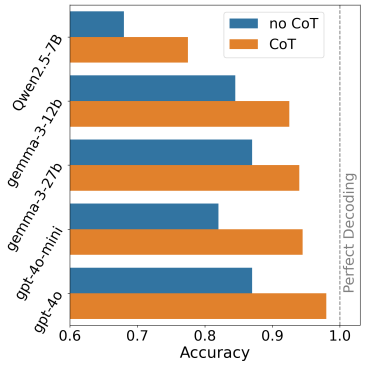

Figure 2: **Accuracy on synthetic dataset given ground truth knowledge state summaries.**

Observation 2: Summarizing knowledge states from raw interactions is non-trivial. Standard LLM summarization approaches can capture high-level skill mastery but often fail to identify crucial latent patterns like student misconceptions (see an example comparison on our synthetic dataset Figure A1 in the Appendix).

Together, these observations suggest that the key challenge lies in learning an encoder that produces faithful summaries, rather than in the decoding step itself.

3.3 Training Objective and Optimization

We propose a reinforcement learning-based approach to train an encoder to produce more faithful and predictive summaries by using downstream decoder accuracy as reward.

Summary generation and prediction Given an interaction history $H_t = (x_1, \ldots, x_t)$, the encoder f_θ maps a subset of interactions $\mathcal{X}_{\text{enc}} \subseteq H_t$ to a textual summary $\tilde{\mathcal{S}} = f_\theta(\mathcal{X}_{\text{enc}})$. The frozen decoder g then conditions on $\tilde{\mathcal{S}}$ to predict responses for two sets $\mathcal{X}$ and $\mathcal{Y}$: interactions used for reconstruction and prediction, respectively. In practice, $\mathcal{X}$ and $\mathcal{Y}$ may be chosen flexibly to include held-out past responses, future responses, or both.

Reward function Given predicted interactions $\tilde{\mathcal{X}} = \{g(q, \tilde{\mathcal{S}}), q \in \mathcal{X}\}$ and $\tilde{\mathcal{Y}} = \{g(q, \tilde{\mathcal{S}}), q \in \mathcal{Y}\}$, the reward for a summary $\tilde{\mathcal{S}}$ is defined as

$$R(\tilde{\mathcal{S}}; g) = \phi\Big(\texttt{Acc}\big(\tilde{\mathcal{X}}, \mathcal{X}\big), \texttt{Acc}\big(\tilde{\mathcal{Y}}, \mathcal{Y}\big),\ |\tilde{\mathcal{S}}|,\ \Omega(\tilde{\mathcal{S}})\Big), \tag{1}$$

where $\texttt{Acc}(\cdot, \cdot)$ measures accuracy, $|\tilde{\mathcal{S}}|$ penalizes overly long summaries, and $\Omega(\tilde{\mathcal{S}})$ enforces optional structural constraints (e.g., inclusion of a `Misconceptions` section). The function Φ balances these components using hyperparameters or indicator functions to enforce constraints as needed.

Optimization via GRPO We optimize the encoder f_θ with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each input $\mathcal{X}_{\text{enc}}$, the encoder generates G candidate summaries $\{\tilde{\mathcal{S}}^1, \ldots, \tilde{\mathcal{S}}^G\}$, each evaluated by $R(\tilde{\mathcal{S}}^i; g)$. We then compute group-relative advantage and update parameters:

$$A(\tilde{\mathcal{S}}^i) = \frac{R(\tilde{\mathcal{S}}^i; g) - \mu}{\sigma}, \quad \nabla_\theta J(\theta) = \frac{1}{G} \sum_{i=1}^{G} A(\tilde{\mathcal{S}}^i) \nabla_\theta \log p_\theta(\tilde{\mathcal{S}}^i \mid \mathcal{X}_{\text{enc}}), \tag{2}$$

where μ and σ are the mean and standard deviation of rewards within the group.

3.4 Steerability of the Estimated Knowledge State

The natural-language summaries generated by LBMs allow for various human-model interactions (detailed in Appendix C):

- **Prompt engineering the encoder.** Since the encoder f_θ is itself an LLM, its behavior can be shaped through prompt design, such as system instructions or in-context examples (Brown et al., 2020).
- **Steering via reward signals.** Rewards to steer the encoder towards human preferences can be incorporated through the $\Omega(S)$ term in Eq. 1.
- **Augmenting with student-specific information.** Educators can supplement the model with additional knowledge not present in observed data—either by augmenting encoder inputs or by editing the generated summary before decoding. This enables integration of recent classroom observations or specific misconceptions identified through in-person interactions.

4 Related Work

We review related works from the Cognitive Diagnosis and Knowledge Tracing literature, as well as concept bottleneck models. See Appendix D for an extended review of related works, and Table D1 for a high-level comparison of LBMs with CD and KT.

Cognitive Diagnosis Cognitive Diagnosis Models (CDMs) infer student knowledge states from observed responses. Classical approaches include Item Response Theory (IRT) and Multidimensional IRT which measure continuous proficiency scores (Rasch, 1993; Reckase, 2006), and the DINA model and its variants which estimate binary mastery of knowledge concepts (De La Torre, 2009). Recent deep learning variants like NeuralCDM (Wang et al., 2022) and RCD (Gao et al., 2021) use neural networks and graph architectures to model complex relationships between students, questions, and knowledge concepts. However, these models typically operate within predefined or inferred knowledge frameworks and provide only quantitative skill mastery estimates.

Knowledge Tracing Knowledge Tracing methods model student learning to predict future performance. Deep learning approaches like DKT (Piech et al., 2015), DKVMN (Zhang et al., 2017) and AKT (Ghosh et al., 2020) employ neural architectures. Despite strong predictive performance, these models represent knowledge as abstract latent vectors, limiting interpretability. Several recent works have proposed more interpretable KT, whether via better question-concept relationship modeling (Minn et al., 2022; Tong et al., 2022; Chen et al., 2023), explainable subsequences (Li et al., 2023) or option tracing (Ghosh et al., 2021). However they remain fundamentally constrained to quantitative concept proficiency estimation or require post-hoc interpretability. Finally, recent LLM-based approaches have shown promise for knowledge tracing tasks (Li et al., 2024a; Wang et al., 2025), but they generally remain opaque, either treating LLMs as black boxes or relying on model-generated explanations susceptible to hallucination.

Concept Bottleneck Models Concept Bottleneck Models (CBMs) (Koh et al., 2020) improve interpretability by using human-understandable concept activations as intermediates between inputs and predictions. However, CBMs typically rely on finite predefined concept sets, limiting applicability to complex tasks like knowledge tracing. Recently, Explanation Bottleneck Models (XBMs) (Yamaguchi & Nishida, 2024) use textual rationales as intermediates for vision classification. While

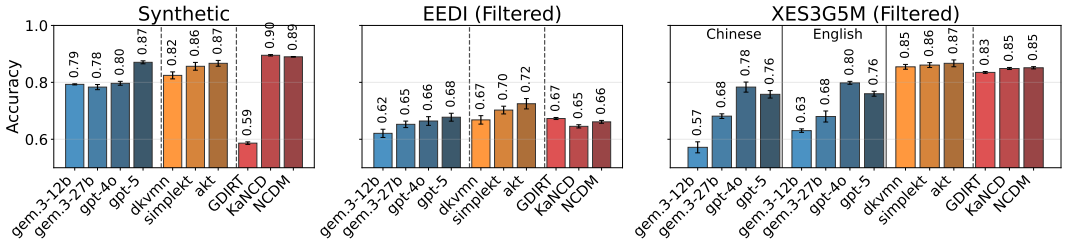

Figure 3: **Accuracy of LBM vs the top-3 KT and CD models across datasets.** Models are grouped by LBMs (blue), KT models (orange) and CD models (red). Top 3 KT and CD models are selected based on average accuracy across all three datasets. Full results for other models are available Table A1 in the Appendix.

our Language Bottleneck Models (LBMs) adopt this language bottleneck concept, they differ fundamentally: unlike XBMs' instance-specific rationales, LBM summaries capture implicit knowledge states that generalize to future, unknown questions, requiring holistic, adaptable summaries rather than task-specific rationales.

5 Experiments

Datasets We evaluate LBMs and baseline models on a synthetic arithmetic benchmark and two real-world datasets (Table 1). Our Synthetic dataset simulates learners answering addition, subtraction, multiplication, and division questions; each student is assigned mastered skills, unmastered skills, and systematic misconceptions. We filter the real-world *Eedi* dataset (Wang et al., 2020) to approximate quasi-static knowledge states. We do a similar filtering for the *XES3G5M* dataset (Liu et al., 2023b), and we evaluate on both Chinese and English versions of the question texts using translations from Ozyurt et al. (2024). More details about datasets and preprocessing are provided in the Appendix B.3.

Table 1: Overview of datasets. AVG#log and STD#log>1 are defined following Wang et al. (2022) as respectively the average number of logs per student per KC, and the mean standard deviation of score per student and per KC.

Dataset	Synthetic	Eedi (Filt.)	XES3G5M (Filt.)
#Students	2,000	623	996
#Questions	5,000	3,401	3,221
#KCs	4	1,284	803
#Logs	420,000	28,947	57,788
#Logs/stud.	210	$\geqslant$40	$\geqslant$34
AVG Acc.	0.55$\pm$0.20	0.68$\pm$0.18	0.85$\pm$0.36
AVG#log	52.5$\pm$0.0	1.65$\pm$0.52	2.02$\pm$0.55
STD#log>1	0.29$\pm$0.12	0.38$\pm$0.08	0.24$\pm$0.14

Models We evaluate LBMs across LLM backbones of different sizes and capabilities, both open-source (Qwen 2.5 3B and 7B (Team, 2024), Gemma 3 12B and 30B (Team, 2025)), and closed-source (GPT-4o-mini, GPT-4o and GPT-5 (Achiam et al., 2023)). Unless noted otherwise, we run the *instruct* variants of each open-source model, use the same backbone LLM for both the encoder and decoder, and prompt all models to provide their response directly without chain-of-thought. We run GPT-5 with `reasoning_effort=minimal` configuration. Hyper-parameters and prompt templates are provided in Appendix B.

Baselines We compare LBMs against 9 Knowledge Tracing methods: DKT (Piech et al., 2015), DKVMN (Zhang et al., 2017), SAKT (Pandey & Karypis, 2019), AKT (Ghosh et al., 2020), Deep IRT (Yeung, 2019), SAINT (Choi et al., 2020), SimpleKT (Liu et al., 2023a), QIKT (Chen et al., 2023), GKT (Nakagawa et al., 2019)) implemented with the PYKT library (Liu et al., 2022) and 5 Cognitive Diagnosis methods (IRT (Rasch, 1993), MIRT (Reckase, 2006), DINA (Junker & Sijtsma, 2001), KaNCD and NeuralCDM (Wang et al., 2022)) implemented with the EduCDM library (bigdata ustc, 2021). We also run each LLM via *direct prompting*, where the LLM predicts answers from the full interaction history without a bottleneck. Training details for all baselines are given in Appendix B.

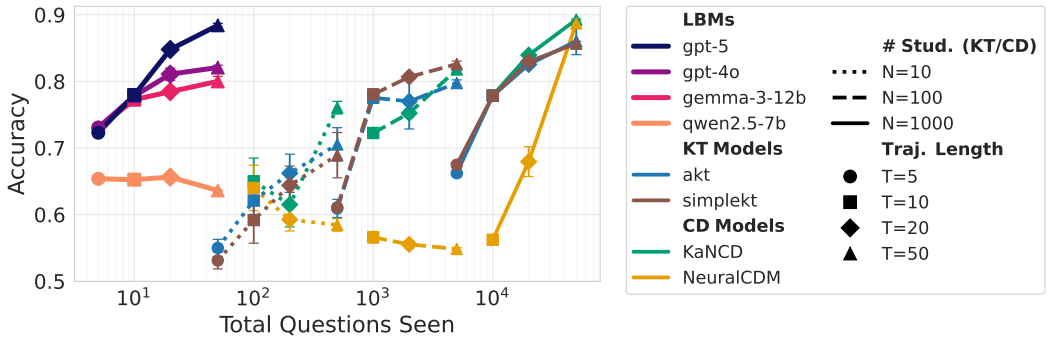

Figure 4: **Accuracy of LBM and the two best KT and CD models on the Synthetic dataset.** The x axis shows the total number of question seen by the model, ie "Traj. Length" x "# Students". Note that #Students $= 1$ for LBMs as they are evaluate zero-shot. CD models are evaluated on held-out test interactions from the same students used during training, while KT models and LBMs are evaluated on 200 unseen students. Results are averaged across N=10 runs for KT and CD models and N=3 for LBMs, with error bars showing the standard error. GPT-5 is ran with `reasoning_effort=minimal` and other models with chain-of-thought prompting.

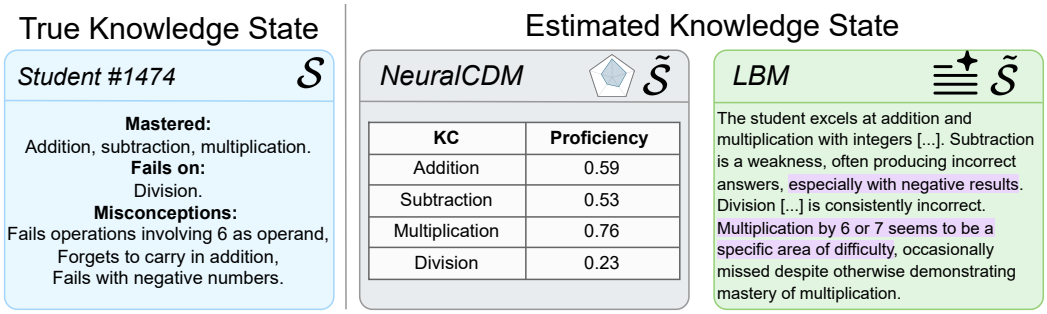

Figure 5: **Case study: comparing CD and LBM knowledge states.** Given a student from the Synthetic dataset, we compare proficiency estimates across knowledge concepts (KCs) obtained from a trained NeuralCDM model to the text-based knowledge state generated by a trained LBM model.

5.1 LBM vs Knowledge Tracing Methods

We present comprehensive results comparing LBMs to baseline methods across all datasets (detailed results in Table A1 in the Appendix). Here we highlight key findings and insights from these experiments.

Performance Figure 3 compares the performance of LBMs against the top three KT and CD models on the Synthetic, EEDI, and XES3G5M datasets. The LBMs are evaluated in a zero-shot setting, whereas the KT/CD models are trained on data from hundreds of students. As expected, LBM performance is strongly tied to the strength of the underlying LLM: with powerful backbones such as GPT-4o and GPT-5, LBMs approach the accuracy of the best KT and CD models across all three datasets. The largest performance gap arises on XES3G5M. However, this dataset has an average accuracy of 85%, implying that even a constant predictor would achieve 85% accuracy. Unlike KT and CD models, LBMs operate zero-shot and thus cannot exploit such dataset-level statistics, which likely explains their lower accuracy but competitive AUC (see Table A1). Finally, thanks to the multilingual capabilities of modern LLMs, LBMs achieve comparable results on both the English and Chinese versions of XES3G5M.

Sample efficiency Figure 4 compares the performance of LBM models to traditional Knowledge Tracing methods on the Synthetic dataset. LBMs with a GPT-5 backbone achieves comparable accuracy to KT methods with significantly less training data. Since CD/KT methods rely on statistical

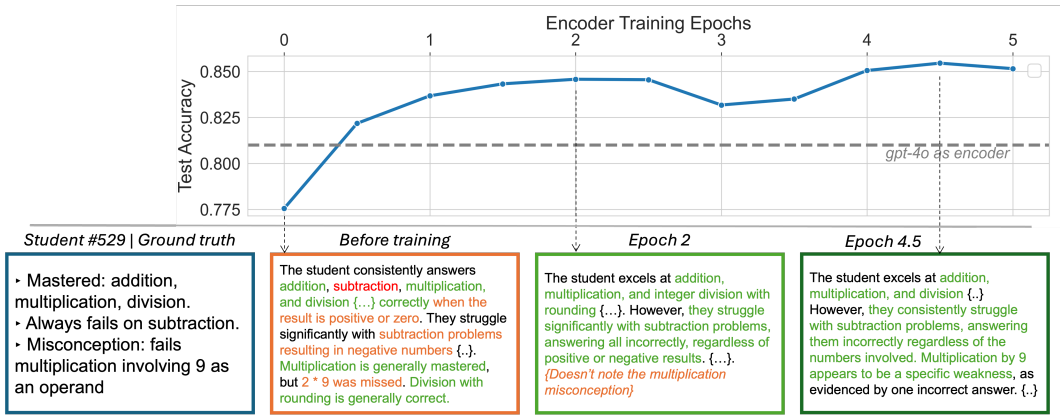

Figure 6: **Training the encoder with the decoding accuracy on the synthetic dataset.** Evolution of the test accuracy as a Gemma3-12B encoder is trained as described in Section 3.3, with a fixed Gemma3-27B decoder. Trained on 800 training students and tested on 200 students, with $|\mathcal{X}| = 50$ questions per input trajectory and $|\mathcal{Y}| = 20$ questions to predict per student. The bottom row show the evolution of the generated summary over the course of training for an example student. Text is colored green (exact), orange (approximate), or red (false) based on ground-truth.

patterns, they require substantially more observations before reaching strong predictive power, while LBMs demonstrate strong zero-shot performance.

Case-study: comparing CD and LBM knowledge state representations We illustrate the interpretability advantages of LBMs over state-of-the-art CD methods in Figure 5. A NeuralCDM model trained on our synthetic dataset achieves strong predictive performance (AUC: 0.96, Acc: 0.90 on the test set). From this model, we extract proficiency estimates across knowledge concepts using the learned student embedding vector. We compare these to the summary generated for the same student by a trained LBM (Gemma-12B encoder; frozen Gemma-27B decoder, cf. Section 5.2).

While NeuralCDM reliably captures overall KC proficiency, its representations are influenced by misconceptions without explicitly identifying them. In contrast, LBMs capture overall proficiency and uncover specific misconceptions (e.g., errors with negative numbers or with operand-6). This ability to provide nuanced, qualitative insights into student knowledge states sets LBMs apart from CD methods.

5.2 Training LBM Encoders

As outlined in Section 3.3, downstream accuracy can serve as a reward signal to train encoders to produce increasingly accurate summaries. We demonstrate this by training a Gemma3-12B encoder with GRPO alongside a frozen Gemma3-27B decoder on 800 students. We set the reward as the decoder accuracy across $|\mathcal{Y}| = 20$ unseen questions $R(\tilde{\mathcal{S}}; g) = \texttt{Acc}\big(\tilde{\mathcal{Y}}, \mathcal{Y}\big)$, train with a LoRA adapter (Hu et al., 2022) and evaluate on 200 unseen test students.

Figure 6 shows the encoder progressively improving summary quality and quickly outperforming GPT-4o. The figure illustrates this through an example student who mastered all constructs except subtraction and fails any multiplication involving 9. The initial summary contains inaccuracies and misses this systematic misconception, while the final summary successfully captures the student's complete knowledge state. Stratifying by knowledge state complexity, we observe larger gains for more complex cases (Appendix A.3).

5.3 Steering LBM behavior

We demonstrate multiple steering strategies described in Section 3.4. Providing explicit misconception information during encoder training produces substantially stronger learning effects than adding the same information at the decoder stage (Appendix A.4.1), suggesting that the encoder uses this additional context to better interpret patterns in student responses. We also show the

encoder can be steered during training to explicitly mention misconceptions through reward signals (Appendix A.4.2), and illustrate how supplementing the summary with information not present in the input can significantly improve decoder accuracy (Appendix A.4.3).

5.4 Ablation Experiments

How much information is lost by the bottleneck? Table 2 compares the accuracy of LBM models to directly predicting new questions from the observed student data. Despite the information bottleneck, LBM accuracy typically remains within 2% of direct prediction—and often surpasses it. Figure 7 shows that this gap decreases for longer bottleneck token limits, highlighting a trade-off between conciseness and predictive accuracy.

Table 2: Accuracy results for Direct and LBM methods. Bold indicates LBM accuracy is no more than 2% below the Direct baseline.

	Synthetic			EEDI (Filtered)		
	Direct	LBM	Δ	Direct	LBM	Δ
Qwen2.5-3B	.56 ± .00	.61 ± .01	**+.06**	.35 ± .00	.38 ± .00	**+.03**
Qwen2.5-7B	.64 ± .00	.65 ± .01	**+.01**	.65 ± .00	.58 ± .01	-.07
gemma-3-12b	.63 ± .00	.79 ± .00	**+.17**	.58 ± .00	.62 ± .01	**+.04**
gemma-3-27b	.62 ± .00	.78 ± .01	**+.16**	.67 ± .00	.65 ± .01	**-.02**
gpt-4o-mini	.81 ± .01	.78 ± .01	-.03	.67 ± .01	.61 ± .02	-.05
gpt-4o	.85 ± .01	.80 ± .01	-.05	.66 ± .00	.66 ± .02	**+.00**
gpt-5	.87 ± .00	.87 ± .01	**+.00**	.71 ± .02	.68 ± .01	-.03

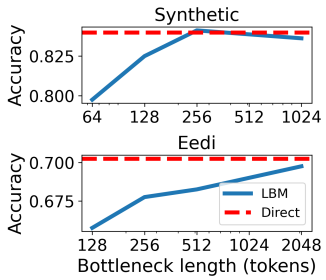

Figure 7: **Evolution of LBM accuracy with bottleneck length (GPT-4o backbone).**

Which of the encoder or decoder is most critical for LBMs? We evaluate LBMs with different encoder–decoder pairings (Appendix A.5.1). Using a strong model (GPT-4o) as encoder with weaker models as decoders yields accuracies $5-10\%$ higher than when the stronger model is used as the decoder. This confirms our hypothesis that extracting relevant information for the summary (encoding) is more challenging than predicting future answers given a summary (decoding).

Do LBMs require knowledge concept information? Table A6 compares LBMs with and without KC information in the input prompt on the Synthetic and EEDI (Filtered) datasets. Performance does not significantly change, demonstrating that LBMs do not fundamentally require KC information.

6 Discussion

Why can't we just prompt GPT-4o directly? The split encoder-decoder architecture of LBMs offers three key advantages over direct LLM prompting: it creates a global student model with a single latent summary shared across all predictions rather than isolated per-question reasoning; it ensures faithful summaries through a closed-loop decoding objective that penalizes non-predictive summaries; and it provides an explicit interface layer that teachers can read, steer and intervene on.

Wider applicability LBMs extend beyond education to any task requiring compact, human-readable summaries with predictive power. The minimal ingredients needed are: (1) a sequence of observations about an entity, (2) a need to predict future behaviors of that entity, and (3) value in having interpretable representations. For example, clinical decision support could distill patient data into textual state descriptions that forecast outcomes while remaining auditable; preventive maintenance could compress sensor logs into explanations predicting machine failure; customer success teams could summarize interaction histories to forecast churn.

Limitations and Future Work LBMs face several constraints including context length limitations, requirements for textual question content, and substantial computational costs. Future extensions could address these through iterative encoding for longer inputs, active sensing for optimal question selection, adaptation for evolving knowledge states, expansion beyond question-answer data, and integration with pedagogical techniques like LearnLM (Team et al., 2024). These limitations and extensions are discussed in detail in Appendix E.

Reproducibility Statement

We are committed to ensuring the reproducibility of our research. All models, datasets, and experimental settings are described in detail in the paper and the appendix.

- Datasets: We use one synthetic and two real-world datasets. The generation process for the synthetic data, along with preprocessing steps for the Eedi and XES3G5M datasets, are detailed in Appendix B.3. The source code for the synthetic data will be made available in an online repository upon publication of this work.
- Implementation Details: Our LBM framework is presented Section 3. The specific LLM backbones, baseline models, training hyperparameters, and software libraries used in the experiments are described in Appendix B. All prompt templates used for the LBM encoder and decoder are provided in Appendix B.5.
- Code: The source code for generating the synthetic data, training the models, and running all experiments will be made publicly available in an online repository upon publication of this work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

bigdata ustc. Educdm. `https://github.com/bigdata-ustc/EduCDM`, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chen Chen, Gerhard Sonnert, Philip M Sadler, Dimitar Sasselov, and Colin Fredericks. The impact of student misconceptions on student persistence in a mooc. *Journal of Research in Science Teaching*, 57(6):879–910, 2020.

Jiahao Chen, Zitao Liu, Shuyan Huang, Qiongqiong Liu, and Weiqi Luo. Improving interpretability of deep sequential knowledge tracing models with question-centric cognitive representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 14196–14204, 2023.

Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the seventh ACM conference on learning@ scale*, pp. 341–344, 2020.

Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, December 1994. ISSN 1573-1391. doi: 10.1007/BF01099821. URL `https://doi.org/10.1007/BF01099821`.

Jimmy De La Torre. Dina model and parameter estimation: A didactic. *Journal of educational and behavioral statistics*, 34(1):115–130, 2009.

Weibo Gao, Qi Liu, Zhenya Huang, Yu Yin, Haoyang Bi, Mu-Chun Wang, Jianhui Ma, Shijin Wang, and Yu Su. Rcd: Relation map driven cognitive diagnosis for intelligent education systems. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 501–510, 2021.

Aritra Ghosh, Neil Heffernan, and Andrew S Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2330–2339, 2020.

Aritra Ghosh, Jay Raspat, and Andrew Lan. Option tracing: Beyond correctness analysis in knowledge tracing. In *International Conference on Artificial Intelligence in Education*, pp. 137–149. Springer, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Brian W Junker and Klaas Sijtsma. Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, 25(3): 258–272, 2001.

JongWoo Kim, SeongYeub Chu, Bryan Wong, and Mun Yi. Beyond right and wrong: Mitigating cold start in knowledge tracing using large language model and option weight. *arXiv preprint arXiv:2410.12872*, 2024.

Jules King, L Burleigh, Simon Woodhead, Panagiota Kon, Perpetual Baffour, Scott Crossley, Walter Reade, and Maggie Demkin. Eedi - mining misconceptions in mathematics. `https://kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics`, 2024. Kaggle.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, 2020.

Douglas Larkin. Misconceptions about "misconceptions": Preservice secondary science teachers' views on the value and role of student ideas. *Science Education*, 96(5):927–959, 2012.

Unggi Lee, Jiyeong Bae, Dohee Kim, Sookbun Lee, Jaekwon Park, Taekyung Ahn, Gunho Lee, Damji Stratton, and Hyeoncheol Kim. Language model can do knowledge tracing: Simple but effective method to integrate language model and knowledge tracing task. *arXiv preprint arXiv:2406.02893*, 2024.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Haoxuan Li, Jifan Yu, Yuanxin Ouyang, Zhuang Liu, Wenge Rong, Juanzi Li, and Zhang Xiong. Explainable few-shot knowledge tracing. *arXiv preprint arXiv:2405.14391*, 2024a.

Jiatong Li, Qi Liu, Fei Wang, Jiayu Liu, Zhenya Huang, Fangzhou Yao, Linbo Zhu, and Yu Su. Towards the identifiability and explainability for personalized learner modeling: an inductive paradigm. In *Proceedings of the ACM Web Conference 2024*, pp. 3420–3431, 2024b.

Qing Li, Xin Yuan, Sannyuya Liu, Lu Gao, Tianyu Wei, Xiaoxuan Shen, and Jianwen Sun. A genetic causal explainer for deep knowledge tracing. *IEEE Transactions on Evolutionary Computation*, 28 (4):861–875, 2023.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, Jiliang Tang, and Weiqi Luo. pykt: A python library to benchmark deep learning based knowledge tracing models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, and Weiqi Luo. simplekt: a simple but tough-to-beat baseline for knowledge tracing. *arXiv preprint arXiv:2302.06881*, 2023a.

Zitao Liu, Qiongqiong Liu, Teng Guo, Jiahao Chen, Shuyan Huang, Xiangyu Zhao, Jiliang Tang, Weiqi Luo, and Jian Weng. Xes3g5m: A knowledge tracing benchmark dataset with auxiliary information. *NeurIPS*, 2023b.

Max Ruiz Luyten and Mihaela van der Schaar. A theoretical design of concept sets: improving the predictability of concept bottleneck models. In *NeurIPS*, 2024.

Haiping Ma, Manwei Li, Le Wu, Haifeng Zhang, Yunbo Cao, Xingyi Zhang, and Xuemin Zhao. Knowledge-sensed cognitive diagnosis for intelligent education platforms. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 1451–1460, 2022.

Sein Minn, Jill-Jênn Vie, Koh Takeuchi, Hisashi Kashima, and Feida Zhu. Interpretable knowledge tracing: Simple and efficient student modeling with causal relations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 12810–12818, 2022.

Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *IEEE/WIC/aCM international conference on web intelligence*, pp. 156–163, 2019.

Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. Label-free concept bottleneck models. In *ICLR*, 2023.

Yilmazcan Ozyurt, Stefan Feuerriegel, and Mrinmaya Sachan. Automated knowledge concept annotation and question representation learning for knowledge tracing. *arXiv preprint arXiv:2410.01727*, 2024.

Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *12th International Conference on Educational Data Mining, EDM 2019*, pp. 384–389. International Educational Data Mining Society, 2019.

Soonwook Park, Donghoon Lee, and Hogun Park. Enhancing knowledge tracing with concept map and response disentanglement. *Knowledge-Based Systems*, 302:112346, 2024.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep Knowledge Tracing. In *NeurIPS*, volume 28, 2015. URL https://papers.nips.cc/paper_files/paper/2015/hash/bac9162b47c56fc8a4d2a519803d51b3-Abstract.html.

George J Posner, Kenneth A Strike, Peter W Hewson, and William A Gertzog. Accommodation of a scientific conception: Toward a theory of conceptual change. *Science education*, 66(2):211–227, 1982.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Georg Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

Mark D Reckase. 18 multidimensional item response theory. *Handbook of statistics*, 26:607–642, 2006.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shuanghong Shen, Qi Liu, Zhenya Huang, Yonghe Zheng, Minghao Yin, Minjuan Wang, and Enhong Chen. A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on Learning Technologies*, 2024.

Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models. In *ICML*, 2023.

John P Smith III, Andrea A DiSessa, and Jeremy Roschelle. Misconceptions reconceived: A constructivist analysis of knowledge in transition. *The journal of the learning sciences*, 3(2): 115–163, 1994.

Gemma Team. Gemma 3. 2025. URL https://goo.gle/Gemma3Report.

LearnLM Team, Abhinit Modi, Aditya Srikanth Veerubhotla, Aliya Rysbek, Andrea Huber, Brett Wiltshire, Brian Veprek, Daniel Gillick, Daniel Kasenberg, Derek Ahmed, et al. Learnlm: Improving gemini for learning. *arXiv preprint arXiv:2412.16429*, 2024.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Jonathan Templin, Robert A Henson, et al. *Diagnostic measurement: Theory, methods, and applications*. Guilford press, 2010.

Jonathan L Templin and Robert A Henson. Measurement of psychological disorders using cognitive diagnosis models. *Psychological methods*, 11(3):287, 2006.

Hanshuang Tong, Zhen Wang, Yun Zhou, Shiwei Tong, Wenyuan Han, and Qi Liu. Introducing problem schema with hierarchical exercise graph for knowledge tracing. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 405–415, 2022.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yu Yin, Shijin Wang, and Yu Su. Neuralcd: a general framework for cognitive diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, 35 (8):8312–8327, 2022.

Fei Wang, Weibo Gao, Qi Liu, Jiatong Li, Guanhao Zhao, Zheng Zhang, Zhenya Huang, Mengxiao Zhu, Shijin Wang, Wei Tong, et al. A survey of models for cognitive diagnosis: New developments and future directions. *arXiv preprint arXiv:2407.05458*, 2024.

Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel Hernández-Lobato, Richard E Turner, Richard G Baraniuk, Craig Barton, Simon Peyton Jones, Simon Woodhead, and Cheng Zhang. Diagnostic questions: The neurips 2020 education challenge. *arXiv preprint arXiv:2007.12061*, 2020.

Ziwei Wang, Jie Zhou, Qin Chen, Min Zhang, Bo Jiang, Aimin Zhou, Qinchun Bai, and Liang He. Llm-kt: Aligning large language models with knowledge tracing using a plug-and-play instruction. *arXiv preprint arXiv:2502.02945*, 2025.

Shin'ya Yamaguchi and Kosuke Nishida. Toward explanation bottleneck models. In *MINT@NeurIPS2024: Foundation Model Interventions*, 2024.

Chun-Kit Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. *arXiv preprint arXiv:1904.11738*, 2019.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pp. 765–774, 2017.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*, 2020.

APPENDIX

A EXTENDED RESULTS

A.1 MOTIVATING OBSERVATION 2

Figure A1 compares summaries produced by different LLMs when prompted to describe a student's knowledge state from 50 question-answer pairs from a synthetic dataset (see Section 5 for details). While all models capture high-level skill mastery, only one correctly identifies a misconception (errors with negative numbers) out of the four existing ones, illustrating that standard summarization approaches often miss crucial latent patterns.

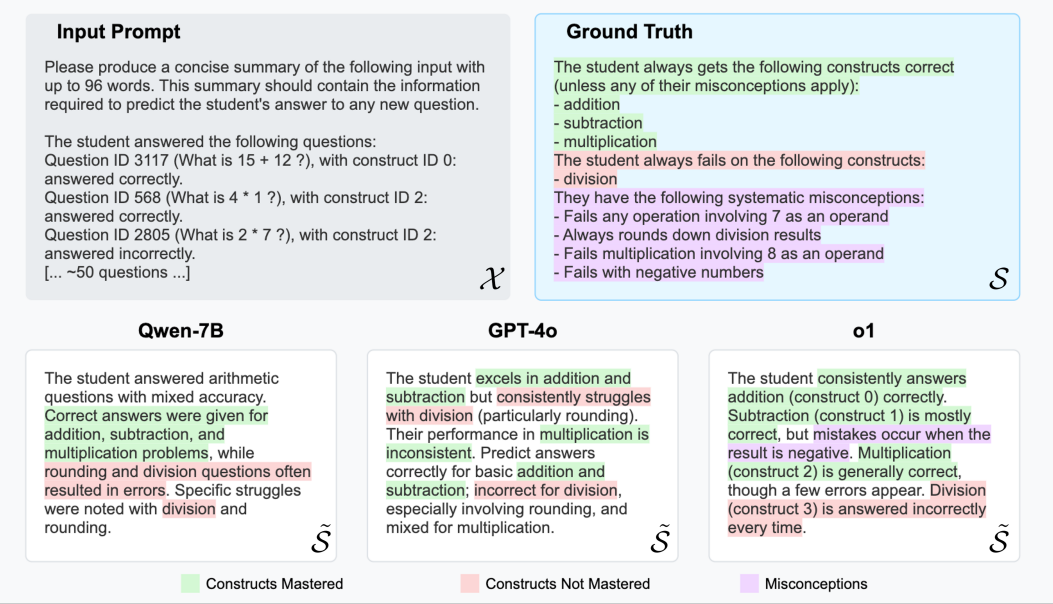

Figure A1: **Zero-shot knowledge state encoding compared to ground truth.** LLM models with different capabilities are prompted to write a summary of the knowledge state (top left panel) of a student, given 50 observed questions and answers provided in text. The ground truth knowledge state (top right panel) describing the student behavior has three main components: constructs mastered, constructs not mastered, and misconceptions. All three models capture the construct mastery correctly, but are not able to capture any misconception, beside o1 which notices the negative numbers misconception (bottom row). Bottom right notations correspond to components in Figure 1.

A.2 FULL RESULTS

Table A1: Results across all datasets and methods. We report the mean and standard deviation across N=10 runs for KT and CD models and N=3 runs for LLM-based models (Direct, LBM). XES3G5M-E and XES3G5M-C denote the English and Chinese versions of the XES3G5M dataset, respectively; note that since the KT and CD models do not use the question content, their results are shared across both versions of the dataset). The KT and CD models are trained on a train set and evaluated on a test set, while LLM Direct and LBM methods are run zero-shot on a test set. AUC unavailable for closed-source LLM models as they require access to the model's output logits. Results for QIKT on Synthetic and MIRT on EEDI are omitted due to implementation issues.

		Synthetic		EEDI (Filt.)		XES3G5M-E (Filt.)		XES3G5M-C (Filt.)	
Model Type	Model Name	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
KT Models (Trained)	akt	.87±.01	.87±.01	.72±.02	.64±.02	.87±.01	.68±.03	.87±.01	.68±.03
	deepIRT	.82±.02	.82±.01	.67±.01	.59±.01	.85±.01	.64±.03	.85±.01	.64±.03
	dkt	.82±.02	.81±.02	.66±.01	.57±.02	.85±.01	.64±.03	.85±.01	.64±.03
	dkvmn	.82±.01	.82±.01	.67±.01	.59±.02	.85±.01	.67±.03	.85±.01	.67±.03
	gkt	.71±.07	.67±.08	.61±.03	.57±.02	.86±.01	.60±.06	.86±.01	.60±.06
	qikt	–	–	.66±.00	.58±.00	.82±.00	.53±.00	.82±.00	.53±.00
	saint	.66±.02	.65±.02	.67±.03	.59±.07	.87±.01	.66±.03	.87±.01	.66±.03
	sakt	.82±.02	.81±.02	.66±.03	.56±.02	.85±.01	.60±.02	.85±.01	.60±.02
	simplekt	.86±.01	.85±.01	.70±.01	.62±.01	.86±.01	.68±.02	.86±.01	.68±.02
CD Models (Trained)	DINA	.57±.01	.61±.01	.51±.00	.55±.01	.50±.01	.55±.01	.50±.01	.55±.01
	IRT	.59±.00	.62±.01	.67±.00	.62±.01	.83±.00	.68±.01	.83±.00	.68±.01
	KaNCD	.90±.00	.95±.00	.65±.01	.64±.01	.85±.00	.81±.00	.85±.00	.81±.00
	MIRT	.61±.00	.65±.01	–	–	.71±.01	.66±.01	.71±.01	.66±.01
	NCDM	.89±.00	.95±.00	.66±.01	.67±.00	.85±.00	.80±.00	.85±.00	.80±.00
LLM Direct (Zero-shot)	Qwen2.5-3B	.56±.00	.79±.00	.35±.00	.54±.00	.19±.00	.56±.00	.19±.00	.55±.00
	Qwen2.5-7B	.64±.00	.73±.00	.65±.00	.62±.00	.76±.00	.63±.00	.75±.00	.63±.00
	gemma-3-12b	.63±.00	.96±.00	.58±.00	.71±.00	.30±.00	.71±.00	.29±.00	.70±.00
	gemma-3-27b	.62±.00	.94±.00	.67±.00	.76±.00	.33±.00	.78±.00	.32±.00	.79±.00
	gpt-4o	.85±.01	–	.66±.00	–	.82±.00	–	.79±.01	–
	gpt-4o-mini	.81±.01	–	.67±.01	–	.74±.08	–	.75±.07	–
	gpt-5	.87±.00	–	.71±.02	–	.81±.01	–	.79±.01	–
LBM (Zero-shot)	Qwen2.5-3B	.61±.01	.65±.01	.38±.00	.55±.01	.27±.01	.58±.01	.30±.01	.63±.02
	Qwen2.5-7B	.65±.01	.69±.01	.58±.01	.55±.01	.75±.01	.63±.00	.74±.01	.62±.02
	gemma-3-12b	.79±.00	.85±.00	.62±.01	.64±.02	.63±.01	.67±.01	.57±.02	.65±.01
	gemma-3-27b	.78±.01	.85±.01	.65±.01	.67±.01	.68±.02	.70±.01	.68±.01	.70±.02
	gpt-4o	.80±.01	–	.66±.02	–	.80±.01	–	.78±.02	–
	gpt-4o-mini	.78±.01	–	.61±.02	–	.70±.08	–	.70±.07	–
	gpt-5	.87±.01	–	.68±.01	–	.76±.01	–	.76±.01	–

A.3 TRAINING LBM ENCODERS: DIFFICULTY STRATIFICATION

To investigate whether the accuracy gains seen during LBM training (Figure 6) vary across students in the dataset, we stratify students according to how many misconception they hold: 0, 1, 2 or 3+. Since each misconception represent additional "irregularities" in the student's response pattern beyond simple mastery of constructs, this effectively stratifies different complexity levels across student knowledge states. Table A2 shows the change in accuracy relative to the GPT-4o baseline across difficulty levels.

Table A2: Relative difference in accuracy between trained Gemma3-12B and the GPT-4o baseline, with students grouped by number of misconceptions.

	Accuracy (mean ± std)			Relative Diff. (%)		
# of Misconceptions	Baseline	Pre-training	Post-training	Pre	Post	Δ
0 ($N = 22$)	0.98 ± 0.05	0.92 ± 0.12	0.99 ± 0.05	-5.80	0.90	**6.70**
1 ($N = 17$)	0.89 ± 0.09	0.87 ± 0.15	0.91 ± 0.09	-2.00	3.00	**5.00**
2 ($N = 23$)	0.81 ± 0.16	0.80 ± 0.12	0.89 ± 0.08	-0.80	10.20	**11.00**
3 ($N = 135$)	0.77 ± 0.12	0.74 ± 0.14	0.82 ± 0.11	-4.80	5.70	**10.50**

Figure A2: **Presence of the word "misconception" in the LBM's summaries during training of the encoder model steered towards mentioning student misconceptions via the reward signal.**

The model shows $10-11\%$ increased accuracy relative to GPT-4o for students with 2 or 3+ misconceptions, compared to just $5-7\%$ for students with 0-1 misconceptions. This suggests the encoder becomes particularly better at handling complex knowledge states.

A.4 Steering LBM behavior

A.4.1 Augmenting with student-specific information to assist LBM training

Finally, we demonstrate how providing additional information to the LBM can assist with the training process. We train two identical LBMs while providing 2 misconceptions either to the encoder as part of the input data, or to the decoder as part of the bottleneck. The models are trained for one epoch on 800 students. We then evaluate the resulting models *without any additional information provided*. Table A3 shows the accuracy before/after training for both models. The model where additional information was provided to the encoder reaches 84% accuracy after one training compared to 80% when it is provided to the decoder. This suggest that the additional information facilitates training of the encoder.

Table A3: Relative difference in accuracy between trained Gemma3-12B and the gpt-4o baseline, with students grouped by number of misconceptions.

	Before training	After 1 epoch training
Information in X	0.802 ± 0.001	**0.840 ± 0.004**
Information in S	0.794 ± 0.002	0.799 ± 0.006

A.4.2 Steering via reward signals

To demonstrate the possibility to steer LBMs' behavior via the reward signal, we consider an example use-case where a teacher would like the model to pay particular attention to potential misconceptions held by the student. Following the reward-shaping framework of Section 3.3, we augment the training objective with an additional term that explicitly encourages the model to surface misconceptions. Concretely, we set $\Omega(S) = \mathbb{1}\big[\text{“misconception”} \in \tilde{S}\big]$, where $\tilde{S}$ is the textual bottleneck emitted by the LBM for student state S. This binary reward is added to the accuracy term and optimized with GRPO. Figure A2 confirms that the policy quickly internalizes this incentive: after only a handful of training steps, the proportion of summaries that explicitly mention a misconception goes from less than 80% to >95%, demonstrating that the reward function provides an effective lever for shaping higher-level pedagogical behavior.

A.4.3 AUGMENTING LBMS WITH STUDENT-SPECIFIC INFORMATION TO COMPLEMENT INPUT DATA

To demonstrate how an LBM can be actively steered with additional information we run the following ablation on the Synthetic dataset: 1. Each student trajectory originally probes four constructs. For a given trajectory, we remove every question linked to one construct leaving the input data intentionally incomplete. 2. We then run the input data through the encoder and inject a single teacher-supplied sentence describing the student's mastery of the missing construct directly into the model's bottleneck representation (e.g., "The student has mastered addition except in the event of misconceptions.).

We repeat this procedure four times—once for each construct—running the LBM both with and without the additional sentence, and report the results Table A4. Naturally, the models complemented with the additional information of the student's mastery of the construct missing in the input data outperform models provided only with the incomplete data. This small experiment illustrates a key advantage of LBMs: because the LBM compresses evidence into a text-based summary, it can be complemented with additional information absent from the input data.

Table A4: Accuracy of a gpt-4o-based LBM on the Synthetic dataset while removing one construct from the input data, with and without providing information about the missing construct in the bottleneck. Each run is repeated across all four constructs; mean and standard deviation are reported.

Without additional bottleneck information	With additional bottleneck information
0.749 ± 0.007	**0.791 ± 0.013**

A.5 ABLATION EXPERIMENTS

A.5.1 ENCODER/DECODER VARIANTS

Table A5 shows the accuracy of different combinations of encoder-decoder models on the Synthetic dataset. The top row shows the result of using the strongest model evaluated (gpt-4o) as both encoder and decoder. Then, we vary either the encoder or decoder part of the LBM across other LLMs, and report the resulting accuracy. A clear pattern which emerges is that the resulting LBMs are stronger when gpt-4o is used as the *encoder* instead of the *decoder*. This implies that the task of accurately capturing knowledge state information in the bottleneck is harder than predicting answers to future questions when provided with a knowledge state summary.

Table A5: Performance of different LBM encoder-decoder combinations on the Synthetic dataset.

	Encoder	Decoder	Accuracy
Strongest model	gpt-4o	gpt-4o	0.809
Strongest model as encoder	gpt-4o	gpt-4o-mini	0.821
	gpt-4o	google/gemma-3-12b-it	0.795
	gpt-4o	Qwen/Qwen2.5-7B-Instruct	0.765
	gpt-4o	Qwen/Qwen2.5-3B-Instruct	0.715
Strongest model as decoder	gpt-4o-mini	gpt-4o	0.765
	google/gemma-3-12b-it	gpt-4o	0.775
	Qwen/Qwen2.5-7B-Instruct	gpt-4o	0.666
	Qwen/Qwen2.5-3B-Instruct	gpt-4o	0.649

A.5.2 IMPORTANCE OF KNOWLEDGE CONCEPTS IN THE INPUT

Table A6 compares the accuracy of different LBMs with vs without knowledge conception information in the input prompt. The performance generally remains similar to the KC version, and it even increases for most models on the EEDI dataset. This demonstrates that LBMs do not fundamentally require KC information.

Table A6: Comparison of LBM performance with vs without knowledge concept (KC) information in the input. Bold indicates a difference in accuracy no more than 3% below the full input.

	Synthetic			EEDI (Filtered)		
	w/ KC	w/o KC	Δ	w/ KC	w/o KC	Δ
Qwen2.5-3B	.61 ± .01	.62 ± .05	**+.00**	.38 ± .00	.38 ± .02	**+.01**
Qwen2.5-7B	.65 ± .01	.66 ± .02	**+.01**	.58 ± .01	.58 ± .01	**+.00**
gemma-3-12b	.79 ± .00	.73 ± .00	-.06	.62 ± .01	.60 ± .02	**-.02**
gemma-3-27b	.78 ± .01	.75 ± .01	-.03	.65 ± .01	.69 ± .03	**+.03**
gpt-4o-mini	.78 ± .01	.76 ± .00	**-.01**	.61 ± .02	.65 ± .03	**+.04**
gpt-4o	.80 ± .01	.76 ± .01	-.04	.66 ± .02	.68 ± .01	**+.02**

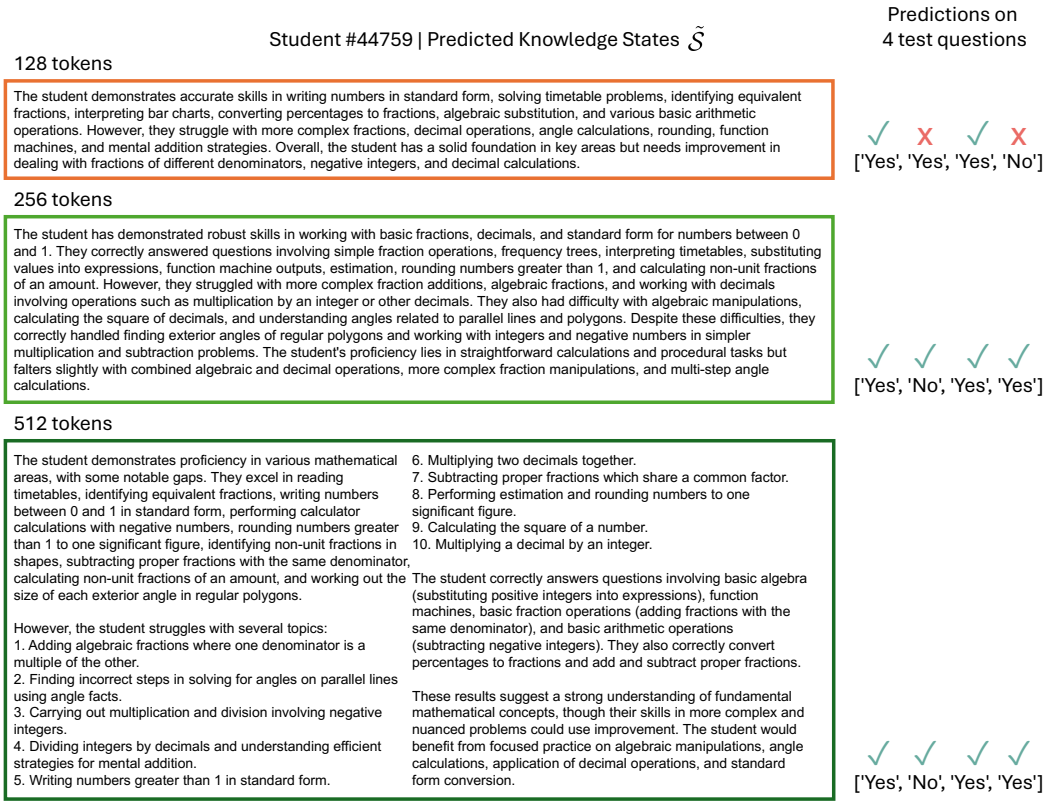

Figure A3: **Example bottlenecks produced by an LBM with gpt-4o as both encoder and decoder for a student in the Eedi dataset, with varying bottleneck lengths.** The input data provided to the encoder is composed of 30 questions across various constructs, and the decoder predicts 4 test questions using the bottleneck.

A.6 Example bottlenecks on the Eedi dataset

Figure A3 shows example bottlenecks produced by a gpt-4o-based LBM for a student in the Eedi dataset. The input data is composed of 30 questions from various constructs, and the decoder predicts 4 test questions. As shown Figure 4 in the main paper, a shorter bottleneck of 128 tokens constrains the expressive power of the model, and in this example the resulting predictions fail on two of the test questions. A longer bottleneck of 256 or 512 tokens allows for more nuance and details in the bottleneck, leading to the decoder correctly predicting all four test questions for this student.

B EXPERIMENTAL DETAILS

B.1 LBMS DETAILS

LLM backbones We evaluate the following closed- and open-source models:

- `GPT-5`, `GPT-4o` and `GPT-4o-mini` (Achiam et al., 2023);
- `Qwen2.5-3B` and `Qwen2.5-7B` (Team, 2024);
- `Gemma3-12B` and `Gemma3-27B` (Team, 2025).

For open-source models we extract the activation logits of the "*Yes*" and "*No*" tokens and return the higher value. The logits for the "*Yes*" and "*No*" tokens are used to compute the AUC. For closed-source models we prompt for a "*Yes*" or "*No*" answer and parse the text output. We run GPT models via the OpenAI API, with the following snapshots: GPT-5: `2025-08-07`; GPT-4o: `2024-08-06`; GPT-4o-mini: `2024-07-18`.

Reward function For training the encoder in Section 6, we set the reward function to the decoder accuracy across $|\mathcal{Y}| = 20$ questions:

$$R(\tilde{\mathcal{S}}; g) = \texttt{Acc}\big(\tilde{\mathcal{Y}}, \mathcal{Y}\big)$$

The RL steering experiment section A.4.2 only rewards the presence of the word "misconception" in the bottleneck:

$$R(\tilde{\mathcal{S}}; g) = \Omega(\tilde{\mathcal{S}}) = \mathbb{1}[\text{'misconception'} \in \tilde{\mathcal{S}}]$$

B.2 KT/CD MODELS

Cognitive Diagnosis models We evaluate the following 5 Cognitive Diagnosis models:

- IRT (Rasch, 1993)
- MIRT (Reckase, 2006)
- DINA (Junker & Sijtsma, 2001)
- KaNCD (Wang et al., 2022)
- NeuralCDM (Wang et al., 2022)

We use the implementation from the EduCDM library (bigdata ustc, 2021). To make sure there is enough data per student to train on, we filter out students students with < 10 interactions in each dataset.

Knowledge Tracing models We evaluate the following 9 Knowledge Tracing models:

- DKT Piech et al. (2015)
- DKVMN Zhang et al. (2017)
- SAKT Pandey & Karypis (2019)
- GKT Nakagawa et al. (2019)
- Deep IRT Yeung (2019)
- AKT Ghosh et al. (2020)
- SAINT Choi et al. (2020)
- SimpleKT Liu et al. (2023a)
- QIKT Chen et al. (2023)

We use the PYKT implementation Liu et al. (2022) for all of these models with default hyperparameters. We only modify the pyKT implementation to additionally compute the accuracy and AUC on $N = |\mathcal{Y}|$ questions.

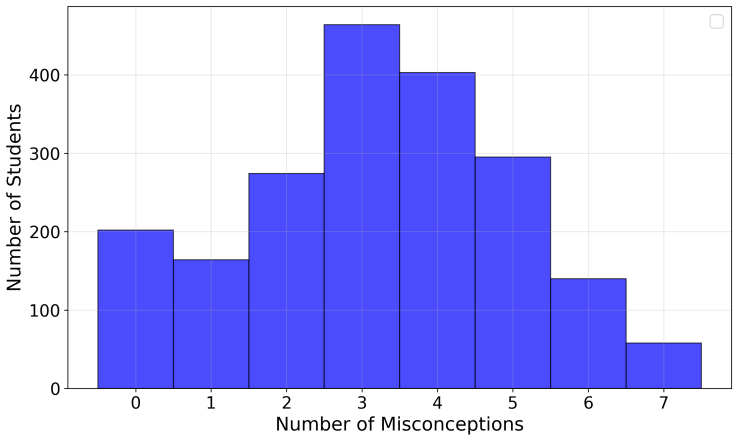

Figure B1: **Distribution of the number of misconceptions per student in the Synthetic dataset.**

B.3 DATASETS

B.3.1 SYNTHETIC

Our synthetic dataset simulates students answering basic arithmetic problems. Each student is characterised by (i) mastered skills, (ii) unmastered skills, and (iii) systematic misconceptions. Questions are arithmetic operations—addition, subtraction, multiplication, or division (rounded to the nearest integer)—between two operands drawn from $[0, 15]$ for addition/subtraction or $[1, 10]$ for multiplication/division.

Misconception pool. For every student we sample misconceptions uniformly at random from:

- forgets to carry in addition;
- fails multiplications involving the number x with $x \sim \mathcal{U}(6, 9)$;
- fails any operation involving the number x with $x \sim \mathcal{U}(6, 9)$;
- fails whenever an operand > 10;
- always rounds division results down;
- fails with negative numbers.

Generation parameters.

- Number of students: 2000;
- Number of questions: 5000;
- Questions answered per student: 210.

Figure B1 shows the histogram of the number of misconceptions per student across the 2000 students of the dataset.

B.3.2 EEDI

The Eedi dataset is analogous to the one publicly shared via the NeurIPS 2020 Education Challenge (Wang et al., 2020) organised by Eedi, but additionally includes the text of each question. While the exact version of the dataset used in this work is not available publicly, a very similar version including question texts is available via the "Eedi - Mining Misconceptions in Mathematics" Kaggle Competition (King et al., 2024).

Preprocessing Although many trajectories in the Eedi dataset span several days or months, we retain only single-session sequences to satisfy the constant knowledge state assumption - a common assumption in Cognitive Diagnosis (Wang et al., 2024). We compute sessions of interactions by grouping answers that are within the follow criterias:

- minimum response time: $5\,\mathrm{s}$, to avoid random answers;
- maximum gap between answers: $3\,\mathrm{min}$;
- minimum trajectory length: 40 questions.

This filtering yields a total of 623 individual trajectories with 40+ questions.

B.3.3 XES3G5M

The XES3G5M dataset contains student interaction data from a large-scale online mathematics learning platform (Liu et al., 2023b). The original dataset has question and construct text in Chinese. We use English translation from Ozyurt et al. (2024) to also run on an English version of the dataset.

Preprocessing The XES3G5M preprocessing follows a similar approach to Eedi, with adjusted parameters to accommodate the characteristics of this dataset. We compute sessions of interactions by grouping answers that meet the following criteria:

- minimum response time: $5\mathrm{s}$, to avoid random answers;
- maximum gap between answers: $10\mathrm{min}$;
- minimum trajectory length: 34 questions.

The increased maximum gap between answers (10 minutes vs. 3 minutes for Eedi) is to ensure that a sufficient number of sessions are available for training. More stringent filtering make it easier to satisfy the constant knowledge state assumption, but might not produce enough data points for effective training of KT and CD models. This filtering yields a total of $1,245$ student-sessions trajectories with 34+ questions, across 996 individual students.

B.4 Experiment Parameters

B.4.1 Table A1, Figure 3, Table 2, Figure 7

Table B1 summarize experimental settings used for Table A1, Figure 3 and Figure 7. Runs are aggregated across N=3 for LBMs/Direct LLMs and N=10 for KT/CD models. For Table 2 only the bottleneck size varies along the plot's x axes.

	Synthetic	Eedi (Filtered)	XES3G5M (Filtered)
$\lvert\mathcal{X}\rvert$	50	30	30
$\lvert\mathcal{Y}\rvert$	4	4	4
Test students	200	100	200
Bottleneck size	128 tokens	512 tokens	512 tokens
CoT prompting	No	No	No

Table B1: Experimental settings for the different datasets.

B.4.2 Figure 4

Latent Bottleneck Models (LBMs)

- $\lvert\mathcal{X}\rvert = N$, $N \in \{5, 10, 20, 50\}$, $\lvert\mathcal{Y}\rvert = 4$
- Test students: 200
- Bottleneck size: 256 tokens

- Chain-of-thought prompting: **Yes**; `reasoning_effort=minimal` for GPT-5
- Encoder & decoder: same backbone LLM (varies by row in the table)

CD baselines The CD models were run using the EduCDM implementation (bigdata ustc, 2021). As these models require training on a per-student basis, we evaluated them on a held-out set of test interactions from the same students seen during training. We used an 80/20 train/test split of each student's interaction history. While this evaluation setup differs from the unseen-student protocol used for KT and LBMs, it allows for a fair comparison of sample efficiency. To ensure stable and reliable accuracy estimates, especially with limited interactions, all results are aggregated and averaged over N=10 independent runs.

KT baselines KT models were evaluated using the PYKT implementation Liu et al. (2022), modified to compute accuracy on $|\mathcal{Y}| = 4$ questions, in order to be comparable with our LBM models. For each method we keep the pyKT default hyperparameters. Test accuracy is computed across 200 unseen test students.

B.4.3 FIGURE 6

- Dataset: Synthetic
- $|\mathcal{X}| = 50$, $|\mathcal{Y}| = 20$
- Train / test students: $800/200$
- Bottleneck size: 128 tokens
- Chain-of-thought prompting: No
- Encoder (trained): `Gemma3-12B`
- Decoder (frozen): `Gemma3-27B`

Training hyper-parameters.

- Batch size : 5
- $G = 5$
- Learning rate $= 1 \times 10^{-4}$
- $\beta = 0.04$
- Optimiser: Adam (`adamw_torch`, default settings)
- **LoRA configuration**
  - $r = 16$, $\alpha = 16$
  - `target_modules = ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]`
  - Dropout: 0.05

B.4.4 ENCODER–DECODER VARIANTS

Identical to the Table A1 set-up, except for the choice of encoder and decoder LLMs.

B.4.5 STEERING EXPERIMENTS

Same dataset and model parameters as for Figure 6.

Training hyper-parameters.

- Batch size : 4
- $G = 4$
- Learning rate $= 1 \times 10^{-4}$
- $\beta = 0.04$
- Optimiser: Adam (`adamw_torch`, default settings)

- **LoRA configuration**
  - $r = 16, \alpha = 16$
  - `target_modules = ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]`
  - Dropout: 0.05

B.5 PROMPTS

Below are the different prompts used to query LLMs in our experiments.

```
template: |
  Please produce a concise summary of the following input with up to {max_words} words.
  This summary should contain the information required to predict the student's answer to any new question.
  {cot_instruction}

  Input:
  {input_text}

  Once again, please provide a **concise** summary of the student's knowledge state.
  {cot_instruction}
  Keep your summary under {max_words} words.

# with CoT
cot_instruction: |
  Think step by step and lay out your reasoning before you write the final summary. Then, enclose the final summary in <info>...</info>.
# without CoT
cot_instruction: |
  Enclose your entire summary in <info>...</info> and do not include anything else.

input_text: |
  The student answered the following questions:
  {question_1_text}
  ...
  {question_n_text}

# with construct information
question_i_text: |
  Question {question_txt, question_ID}, with construct{construct_txt, construct_ID}: answered {correctness}.

# without construct information
question_i_text: |
  Question {question_txt, question_ID}: answered {correctness}.
```

Listing 1: Base Prompt for the Encoder.

```
template: |
  Here is some information about a student:
  {bottleneck}

  Predict whether the student will answer {new_question} correctly or not.
  Answer with "Yes" or "No" and nothing else.
```

Listing 2: Base Prompt for the Decoder.

```
template: |
  Please produce a concise summary of the following input with up to
       {max_words} words.
  This summary should contain the information required to predict
      the student's answer to any new question.

  Make sure to mention any misconception held by the student.
  {cot_instruction}

  Input:
  {input_text}

  Once again, please provide a **concise** summary of the student's
      knowledge state.
  {cot_instruction}
  Keep your summary under {max_words} words.
```

Listing 3: Base Prompt for the Encoder when steering for mentioning "misconceptions" (Section A.4.2).

B.6 Code and hardware

Experiments were ran on four NVIDIA A100 GPUs with 80GB VRAM each, and 880GB of total RAM.

C Steerability of the Estimated Knowledge State

A key advantage of LBMs is the ability for humans to interact with the model to steer the estimated student knowledge states and complement the model with additional information. Here, we further discuss the three mechanisms for human-model interaction in the LBM framework mentioned in Section 3.4.

Prompt engineering the encoder. The most straightforward approach is directly shaping how the encoder generates summaries through prompt engineering, for example via system instructions or in-context examples (Brown et al., 2020). By modifying the instruction prompt given to the encoder, educators can influence the format, emphasis, and level of detail in the generated knowledge state summaries. For instance, instructing the encoder to highlight specific types of misconceptions or to focus on particular subject areas can yield more targeted summaries. Examples of good and bad knowledge states can also be provided to the encoder to steer its behavior via in-context learning.

Steering via reward signals during training. When training the encoder, human preferences can be incorporated through the reward function by using the $\Omega(S)$ term from Eq. 1. This allows for systematic enforcement of desirable summary properties across the model's outputs. For example, if educators find that explicit enumeration of misconceptions is particularly valuable for intervention planning, the reward function can be designed to favor summaries that consistently identify and articulate student misconceptions – as demonstrated Figure A.4.2.

Augmenting with student-specific information. Perhaps the most powerful form of interaction involves supplementing the model with additional student-specific information not present in the observed interaction data X. This can occur in two ways:

- Augmenting encoder input: Educators can provide supplementary information alongside the observed interactions X, such as notes about recent classroom activities not yet reflected in assessment data, or observations about a student's learning process not captured in their answers.
- Modifying the generated summary: After the encoder produces a knowledge state summary S, educators can directly edit this summary based on their domain expertise and student-specific knowledge before passing it to the decoder. For example, a teacher who noticed

a student struggling with negative numbers during an in-class exercise could add this observation to the generated summary, even if the available assessment data contains few questions involving negative numbers.

D Extended Related Work

This section provides an extended review of the literature related to our proposed approach, spanning traditional knowledge tracing, recent advances in LLM-based student modeling, and concept bottleneck models (CBMs), and text summarization models.

D.1 Cognitive Diagnosis

Cognitive Diagnosis Models (CDMs) aim to infer a student's latent knowledge state from their observed responses to test questions Wang et al. (2024).

D.1.1 Psychometrics-based CD models

Classical models for Cognitive Diagnosis originate from psychometrics, including Item Response Theory (IRT) and its multidimensional variant (MIRT), which model continuous scores of knowledge proficiency using logistic functions Rasch (1993); Reckase (2006). The DINA model estimates binary mastery variables of knowledge concepts, assuming that students must master all required skills to answer correctly, while accounting for slips and guesses Junker & Sijtsma (2001). Other variants have been proposed with different assumptions, such as DINO (Templin & Henson, 2006) which considers that students will correctly answer the item if at least one required knowledge concept is mastered. A critical input for these methods is the Q-matrix, which describe which knowledge concept is required for each question.

D.1.2 Deep Learning-based CD models

More recent deep learning approaches offer greater flexibility in modeling complex relationships. The Neural Cognitive Diagnosis Model (NCDM) uses neural networks to learn interaction functions between student proficiency vectors and item characteristics Wang et al. (2022). Extensions include Kernel-based Neural Cognitive Diagnosis (KaNCD), which models latent associations between knowledge concepts Wang et al. (2022), and Knowledge-Sensitive Cognitive Diagnosis (KSCD), which learns intrinsic relations between knowledge concepts from student responses Ma et al. (2022). Graph neural networks have also been incorporated, with frameworks like RCD capturing relationships between students, questions, and knowledge concepts Gao et al. (2021). Recent encoder-decoder architectures, such as ID-CDF, enable inductive diagnosis by directly encoding student responses into ability vectors Li et al. (2024b). While these deep learning models provide enhanced predictive power and can handle diverse data types, they still typically operate within predefined knowledge concept frameworks and are limited to quantitative estimates of skill mastery.

D.2 Knowledge Tracing

Compare to Cognitive Diagnosis which assumes a constant knowledge state, Knowledge Tracing method aim at estimating evolving knowledge states as students answer questions. We similarly review

D.2.1 Traditional Knowledge Tracing

Traditional knowledge tracing (KT) methods have long been used to model student learning and predict future performance Shen et al. (2024). Early works such as Bayesian Knowledge Tracing (BKT) Corbett & Anderson (1994) model student mastery as a probabilistic process, while more recent methods like Deep Knowledge Tracing (DKT) Piech et al. (2015), Dynamic Key-Value Memory Networks (DKVMN) (Zhang et al., 2017) and Attentive Knowledge Tracing (AKT) Ghosh et al. (2020) uses neural networks or attention-based architecture to learn contextual representation of questions and student knowledge states. Despite their strong predictive performance, these models

represent a student's knowledge state as an abstract, high-dimensional latent vector, which poses significant challenges in interpretability and actionable feedback for educators.

D.2.2 INTERPRETABLE KNOWLEDGE TRACING

Several recent works have proposed more interpretable Knowledge Tracing methods.

Enhanced Concept Proficiency Modeling Interpretable Knowledge Tracing (IKT) (Minn et al., 2022) introduces a causal probabilistic student model based on skill mastery, ability profiles, and problem difficulty. While this approach provides clearer connections between model components and predictions, interpretability remains tied to quantitative skill mastery estimates. Similarly, QIKT (Chen et al., 2023) employs IRT functions as the final prediction layer, combining question-centric knowledge acquisition, knowledge mastery scores, and knowledge application scores through encoder modules. Despite enhanced modeling of question-concept relationships, the diagnostic output remains in the form of proficiency reports across predefined knowledge concepts.

Learned Questions Relationships HGKT (Tong et al., 2022) addresses limitations of concept-based proficiency by modeling hierarchical relationships between questions and introducing problem schemas as additional links between questions. Problem schemas are discovered through hierarchical clustering of question embeddings via BERT encodings, providing a means of grouping questions orthogonal to knowledge concepts and therefore enabling more detailed diagnostic reports than at the concept proficiency level. However, the interpretability of these learned schemas requires post-hoc interpretation using TextRank to infer schema descriptions from clusters of questions. While this approach moves beyond simple concept proficiency, it still relies on learned embeddings that cannot directly articulate student reasoning patterns without post-hoc manipulations.

Explainable subsequences Explainable subsequences provide an alternative to interpretability via concept proficiency by identifying which past questions are most relevant for predicting future responses. For example, Li et al. (2023) proposes a genetic algorithm to identify explainable subsequences in student interaction histories better than with standard Deep Learning explanation methods such as Shapley values or gradient-based saliency maps. While this allows for a different kind of interpretability from concept mastery, the explanations remain at the level of question relevance rather than underlying reasoning processes. This approach can indicate which questions matter but cannot explain why they matter or what misconceptions they reveal.

Option Tracing Option Tracing (Ghosh et al., 2021) moves beyond binary correctness by modeling which specific option a student selects in multiple-choice questions, enabling finer-grained analysis of misconceptions through patterns in distractor choices. Similarly, Park et al. (2024) leverages MCQ responses and concept maps to disentangle student understanding at the concept level. While their approach is motivated by misconception detection, their IRT-based predictions remain grounded in concept-level proficiency prediction and do not explicitly validate misconception identification.

D.2.3 LLM-BASED KNOWLEDGE TRACING

Recent studies have begun integrating Large Language Models (LLMs) into the KT framework. For example, Li et al. (2024a) demonstrated that LLMs are able to make sensible predictions about student responses when prompted with adequate information. Other works Lee et al. (2024); Kim et al. (2024) have studied how LLMs can help mitigate the cold-start problem compared to traditional KT approaches, while Wang et al. (2025) demonstrated state-of-the-art performances in KT by combining LLMs with sequence interaction models. However, these methods generally remain opaque: they either treat the LLM as a black-box, or rely on model-generated explanations that are susceptible to hallucination (Bender et al., 2021). KCQRL (Ozyurt et al., 2024) leverages language models to improve the embedding of any deep learning KT method by encoding semantic information about question content. While this leads to improved predictive accuracy, its interpretability remains limited to knowledge concept proficiency.

D.3 Comparison of LBMs to KT and CD

Table D1 gives a high-level comparison of LBMs and KT/CD models. We acknowledge that this summary table is a simplification of the KT and CD fields, as each contains many works that have been proposed to tackle these individual limitations (for example, using textual question contents to improve embeddings (Ozyurt et al., 2024), or using specialized Multiple Choice Questions to reveal misconceptions via KT-type methods(King et al., 2024)). Nevertheless, it clarifies our paper's similarities with these two fields, while contrasting key differences that enable LBMs to address educational scenarios where neither traditional KT nor CD methods are well-suited —particularly for misconception identification or when working with limited data.

Table D1: Comparison of Cognitive Diagnosis, Knowledge Tracing and our proposed Language Bottleneck Models.

Aspect	**Knowledge Tracing (KT)**	**Cognitive Diagnosis (CD)**	**Language Bottleneck Models (LBMs)**
		Interpretability Capabilities	
Interpretable Output	Quantitative proficiency across knowledge concepts	Quantitative proficiency across knowledge concepts	Qualitative text summaries
Misconception Detection	No	No	Yes
Requires Predefined or Inferred Concepts for Interpretability	Yes	Yes	No
		Data Requirements	
Primary Input Modality	Question and/or Construct IDs	Question and Construct IDs (Q-matrix)	Any textual information
Training Data Requirements	High	High	Low/Zero-shot
		Modeling Characteristics	
Knowledge State Assumption	Dynamic	Static	Static
Human-in-the-Loop	Limited	Limited	High (steerable & editable)
		Fundamental Distinctions	
Core Question Addressed	"Can we predict a student's responses over time?"	"Can we quantitatively estimate a student's proficiency across concepts?"	"Can we qualitatively estimate a student's knowledge state, including knowledge concepts and misconceptions?"
Case Study (Figure 5)	*Similar to CD*	*Proficiency vector: KC1 (Add): 0.59, KC2 (Sub): 0.53, KC3 (Mul): 0.76, KC4 (Div): 0.23*	*"...excels at addition and multiplication... Subtraction is a weakness... Multiplication by 6 or 7 seems to be a specific area of difficulty..."*

D.4 Concept Bottleneck Models

Concept Bottleneck Models (CBMs) Koh et al. (2020) improve interpretability through human-understandable concept activations as intermediates, with extensions exploring unsupervised concept

learning Oikarinen et al. (2023), test-time interventions Shin et al. (2023), and theoretical analyses of concept set design Luyten & van der Schaar (2024). However, CBMs typically rely on finite predefined concept sets, limiting their applicability to complex tasks like knowledge tracing. Recently, Yamaguchi & Nishida (2024) introduced Explanation Bottleneck Models (XBMs), which use textual rationales as intermediates for vision classification, justifying a single known label per input. While our Language Bottleneck Models (LBMs) adopt this language bottleneck concept, they differ fundamentally. Unlike XBMs' instance-specific, task-specific rationales, LBM summaries aim to capture an *implicit* knowledge states—as emphasized by our inverse problem formulation—and generalize to future, unknown questions. These requirements necessitate holistic, adaptable summaries rather than XBMs' task-specific rationales, leading us to introduce Language Bottleneck Models (LBMs) as a distinct framework with broader applicability.

D.5 Text Summarization Models

Recent text summarization models such as BART Lewis et al. (2019), T5 Raffel et al. (2020), and PEGASUS Zhang et al. (2020) effectively produce concise summaries based on explicitly available textual content. However, these approaches differ fundamentally from knowledge tracing, where summaries must infer implicit student knowledge states not directly observable in the input. Standard summarization metrics (e.g., ROUGE, BLEU) rely on explicit reference summaries, making them unsuitable for evaluating latent knowledge inference tasks. In contrast, our Language Bottleneck Models generate textual summaries of the student's implicit knowledge state, optimized for predictive accuracy on downstream questions rather than syntactic overlap with the observed input.

E Extended Discussion

E.1 Extended Discussion

Parallel with Kolmogorov Complexity The inverse problem formulation of Knowledge State Modeling draws a natural parallel with algorithmic information theory. The Kolmogorov complexity $K(x)$ of a string x is defined as the length of the shortest program that outputs x when run on a universal Turing machine. Traditional KT methods focus on learning statistical regularities between questions and responses, without looking for parsimonious explanations for observed behavior. In contrast, LBMs explicitly search for minimal natural language descriptions that can both reconstruct past interactions and predict future responses. The knowledge state summary $\mathcal{S}$ is akin to the minimal program, and the decoder LLM functions is akin to the universal interpreter that unpacks $\mathcal{S}$ to generate the observed question-answer patterns. This connection suggests that effective knowledge state summaries should capture the algorithmic essence of student behavior—the underlying "program" of knowledge and misconceptions that generates observable responses—rather than merely fitting surface-level patterns. While true Kolmogorov complexity is uncomputable, LBMs approximate this ideal through the natural language bottleneck constraint, encouraging summaries that balance compression with predictive power.

Computational Architecture and Design Choices The encoder-decoder architecture reflects a deliberate separation of concerns: extraction versus interpretation of knowledge states. Our experiments demonstrate that these two tasks have different intrinsic complexity, with encoding (summary generation) being more challenging than decoding (prediction from summaries). This finding has practical implications for model selection and computational resource allocation. The use of GRPO for encoder training represents a novel application of reinforcement learning to interpretable AI, where the reward signal directly measures the downstream utility of explanations rather than their surface-level quality.

E.2 Extended Limitations

Context Length Constraints A practical limitation of LBMs is the context length restrictions of current LLMs. Eventhough modern models can handle thousands of tokens, comprehensive student histories in real educational settings can easily exceed these limits. As student trajectories grow more complex and include more information, the context required for the encoder might go beyond what

current LLMs are capable of. A natural solution could be an iterative encoding process, where the encoder iterates over the text bottleneck while going over the input data by windows.

Computational Cost and Resource Requirements Since LBMs involve two LLMs as the encoder and decoder, they are typically more computationally intensive to run than traditional KT methods. Each inference call with LBMs requires two LLM calls (encoding and decoding) with extensive context windows. Training LBMs with GRPO is particularly costly, as it requires iteratively generating and evaluating multiple candidate summaries per training example. This cost structure may limit the practical deployment of LBMs to high-stakes educational contexts where the interpretability benefits justify the computational expense.

Dependence on Textual Question Content Many existing CD/KT datasets provide only question identifiers rather than full question text. This limits the direct applicability of LBMs, as they require the questions content in order to generate meaningful summaries about student knowledge. While question content is increasingly available in modern educational platforms, this dependency creates a barrier to applying LBMs to historical datasets or systems that rely primarily on item response theory frameworks.

Constant Knowledge State Assumption The constant knowledge state assumption underlying the inverse problem formulation, while reasonable for short diagnostic sessions, does not cover longer time horizons. Real learning involves continuous knowledge acquisition, forgetting, and misconception evolution that our current framework cannot capture. This limitation restricts current LBMs to diagnostic rather than contexts of assessment throughout the learning process. The assumption also fails to account for contextual factors (fatigue, motivation, external stressors) that can significantly impact student performance within even short sessions.

E.3 Extended Future Work

Iterative Encoding Strategies A natural extension to address context length limitations involves iteratively applying the encoder while chunking input data. At each step, the encoder would process a portion of the interaction history X along with one or more previously generated bottleneck summaries, creating an updated summary that incorporates new information from the latest chunk. This approach could maintain both detailed recent context and compressed historical patterns, enabling LBMs to handle arbitrarily long student trajectories through sequential refinement of knowledge state representations.

Active Learning and Question Selection Building on iterative encoding, active learning strategies could provide encoders with the most informative input questions. Rather than processing all available interactions, the system could strategically select questions that maximize information gain about uncertain aspects of student knowledge. This could leverage uncertainty quantification methods or use the encoder LLM itself to recommend the most diagnostically valuable questions. Such active sensing would improve both efficiency and diagnostic power by focusing computational resources on the most insightful student responses.

Dynamic Knowledge State Modeling Extending LBMs to handle evolving knowledge states represents a significant next step necessary to account for progressive learning and forgetting effects. A natural relaxation of the constant knowledge state assumption involves "piecewise-constant" knowledge states that can evolve between question sessions but remain static within sessions. This extension poses exciting challenges in developing training objectives that balance within-session consistency with between-session learning dynamics, potentially requiring new approaches to temporal modeling in natural language representations.

Multimodal Input Integration Question-answer pairs represent a relatively narrow information source for inferring student knowledge states. Richer data sources such as student-tutor interactions, self-reported explanations for answers, timing patterns, or hint usage could provide deeper insights into student understanding. While LBMs naturally extend to text-based inputs, future work should investigate optimal strategies for combining these varied data sources and evaluate their relative contributions to knowledge state inference accuracy and interpretability.

Pedagogical Alignment Recent work like LearnLM Team et al. (2024) demonstrates the potential for making LLMs more pedagogically aligned. Incorporating similar pedagogical principles into LBM components could help encoders better interpret educational interactions and generate summaries that align with expert teaching practices. This might involve training on educator-annotated examples, incorporating educational taxonomies into summary structure, or using reward functions that emphasize pedagogically relevant aspects of student knowledge states. Such alignment could bridge the gap between computational convenience and educational validity.

STATEMENT: USE OF LARGE LANGUAGE MODELS

Large Language Models (LLMs) were used as an assistive tool in the preparation of this manuscript and the development of the accompanying code.

- LLMs were used to improve the clarity, grammar, and flow of the text. This included rephrasing sentences, correcting typographical errors, and ensuring a consistent tone.
- LLMs were used to generate boilerplate code for data processing scripts, assist in debugging, and suggest implementations for standard machine learning components.

The authors take full responsibility for the final content of this paper.