ARC: Agentic Rank and Critique for Knowledge Component

Anonymous ACL submission

Abstract

Identifying the mathematical concepts required to solve a math word problem (MWP) is fundamental for Intelligent Tutoring Systems (ITS), supporting prerequisite diagnosis, curriculum sequencing, and personalized practice recommendation. Prior work has largely cast this as flat multi-label classification, predicting unordered Knowledge Component (KC) sets that ignore dependencies among concepts. We reformulate KC prediction as structured graph construction, where nodes represent required KCs and directed edges encode prerequisite relations. We propose ARC, an Agentic Rank and Critique framework, in which a Rank Agent retrieves candidate KCs, then a Critic Agent evaluates the current graph state and issues construction instructions, and an Architect Agent incrementally builds the KC graph through iterative feedback. Experiments and ablation studies show that ARC is particularly effective on hard MWPs compared to supervised fine-tuned and prompted LLM baselines, and produces interpretable KC dependency graphs suitable for downstream educational use.

1 Introduction

Effective mathematics instruction requires understanding not only whether a student can solve a problem, but also which mathematical concepts the problem involves and how those concepts relate to one another. The Common Core State Standards (CCSS) for Mathematics provide a principled taxonomy for representing such knowledge, each standard Knowledge Component (KC) corresponds to a specific mathematical skill or operation. Identifying the KCs required by a math word problem (MWP) is therefore an important capability for Intelligent Tutoring Systems (ITS), supporting prerequisite diagnosis, curriculum sequencing, and personalized practice recommendation.

Recent efforts have begun to annotate MWP datasets such as KC-Tagging (Tan et al., 2025), it contains ASDiv (Miao et al., 2020), GSM8K (Cobbe et al., 2021), and MATH (Hendrycks et al., 2021) with CCSS-aligned KC labels. However, despite the growing availability of annotated resources, KC prediction for MWPs remains underexplored. Existing approaches largely formulate the task as flat multi-label classification, predicting an unordered set of KCs for each problem. While such labels indicate which concepts are involved, they fail to capture the dependency structure among them. For example, solving a problem may require first performing unit conversion, then multi-digit multiplication, and finally percentage application. These operations form a procedural chain in which later steps depend on earlier ones. In addition, scaling flat classifiers to large KC taxonomies with sparse annotations remains challenging.

To address these limitations, we reformulate KC prediction as a structured graph construction task. For each problem, we construct a KC *graph* in which nodes represent required KCs and directed edges encode prerequisite relations between them. Based on this formulation, we propose ARC, a multi-agent framework consisting of three collaborative components. A **Rank Agent**, assisted by a **Judge Model**, retrieves a focused set of candidate KCs. A **Critic Agent** evaluates the current graph state and generates graph-construction instructions, while an **Architect Agent** incrementally updates the graph according to instructions and sends feedback to the **Critic Agent**.

Our contributions are:

- Introduce *KC graph construction* as a task formulation that jointly models concept identification and dependency structure for MWPs, along with KC-stratified evaluation splits on ASDiv, GSM8K, and MATH.
- Develop a binary problem-KC Judge Model that efficiently narrows large KC taxonomies

into high-quality candidate sets for downstream reasoning.

- Propose ARC, a multi-agent framework that turns flat KC prediction into iterative graph construction and produces interpretable KC dependency graphs.

2 Related Work

The KC-Tagging dataset (Tan et al., 2025) annotates MWPs with CCSS-defined KC labels, casting the task as multi-label prediction over problem-level requirements. Existing content-side works treat KC labelling as flat classification: Reales et al. (Reales et al., 2024) predict which KCs a resource touches via multi-label classification, and the LLM-augmented KT survey (Cho et al., 2024) shows LLMs are increasingly used to tag KCs for problems, but relations *among* required KCs are not modeled in either case. Learner-side methods such as Deep Knowledge Tracing (Piech et al., 2015) and large-scale interaction logs like EdNet (Choi et al., 2020) borrow the KC vocabulary for mastery modeling and exercise mapping, but similarly operate over flat KC sets with no relational structure between them.

Graph-based methods in the educational domain treat graphs as fixed input rather than as a prediction target. Graph-based Knowledge Tracing (Nakagawa et al., 2019) propagates mastery over a concept graph assumed to be given, and Han et al. (Han et al., 2025) transfer knowledge across courses through a heuristically derived graph. Educational knowledge graph pipelines (Abu-Salih and Alotaibi, 2024; Canal-Esteve and Gutiérrez, 2024) construct knowledge graphs over educational content, but their edges are generic ontological or co-occurrence relations and construction is rule- or extraction-driven with no held-out evaluation of predicted structure. Graph representation learning methods (Wu et al., 2021) similarly take an existing graph as input and learn encoders over it.

The agentic paradigm is relevant to the iterative nature of graph construction. ReAct (Yao et al., 2023) shows that interleaving reasoning and action steps improves LLM performance on grounded tasks; Reflexion (Shinn et al., 2023) further demonstrates that self-correction through verbal feedback sharpens multi-step decisions. Multi-agent frameworks such as AutoGen (Wu et al., 2024) and Toolformer (Schick et al., 2023) extend this to collaborative tool-using agents, but these methods have been applied to QA, navigation, and coding, not to the structured construction of educational graphs.

Candidate selection before any agentic loop requires a strong ranking component. Dense retrieval (Karpukhin et al., 2020) paired with neural rerankers (Nogueira et al., 2020; Sun et al., 2023; Adeyemi et al., 2024) and embedding models such as Qwen3-Embedding (Zhang et al., 2025; Yang et al., 2025) are well-established for open-domain retrieval. A naive embedding-based approach achieves high recall but floods the candidate set with noise, which degrades downstream reasoning quality.

3 Method

3.1 Overview of ARC

ARC is a multi-agent framework for KC graph construction that iteratively coordinates three agents: a Rank Agent ($\mathcal{A}_r$) that filters the candidate pool, a Critic Agent ($\mathcal{A}_c$) that evaluates the current KC set and issues instructions to the Architect Agent ($\mathcal{A}_a$) for execution and feedback. The goal of ARC is to produce a KC graph $\mathcal{G}_i^T$ for each problem p_i, where nodes are the required KCs and directed edges encode prerequisite relations. Formally, we define the KC graph construction task as follows:

$$\mathcal{G}_i^T = \texttt{ARC}(p_i, \mathcal{K}, [\mathbf{P}]) \tag{1}$$

where $\mathcal{K}$ is the full KC taxonomy and $\mathbf{P}$ is the optional External Knowledge.

As illustrated in Figure 1, for each problem p_i, $\mathcal{A}_r$ first scores the retrieved KC set, $\mathcal{A}_c$ then performs candidate filtering to reduce the context space and determines the next action instructions to refine and construct the KC graph. $\mathcal{A}_a$ executes the instructed actions and feeds the results back into $\mathcal{A}_c$'s Experience Memory.

3.2 Rank Agent

The full KC set with taxonomy and problem text can be large, and using it directly may exceed the model's context capacity. To efficiently support KC graph construction, we need a mechanism to reduce the context space. In this paper, we design $\mathcal{A}_r$ to learns a Judge Model $\texttt{JM}_\theta$ that estimates the compatibility between a problem p_i and a KC k_j:

$$\alpha_{i,j} = \texttt{JM}_\theta(p_i, k_j) \tag{2}$$

The score $\alpha_{i,j}$ reflects how likely k_j is required to solve p_i. At inference time, $\texttt{JM}_\theta$ scores the whole

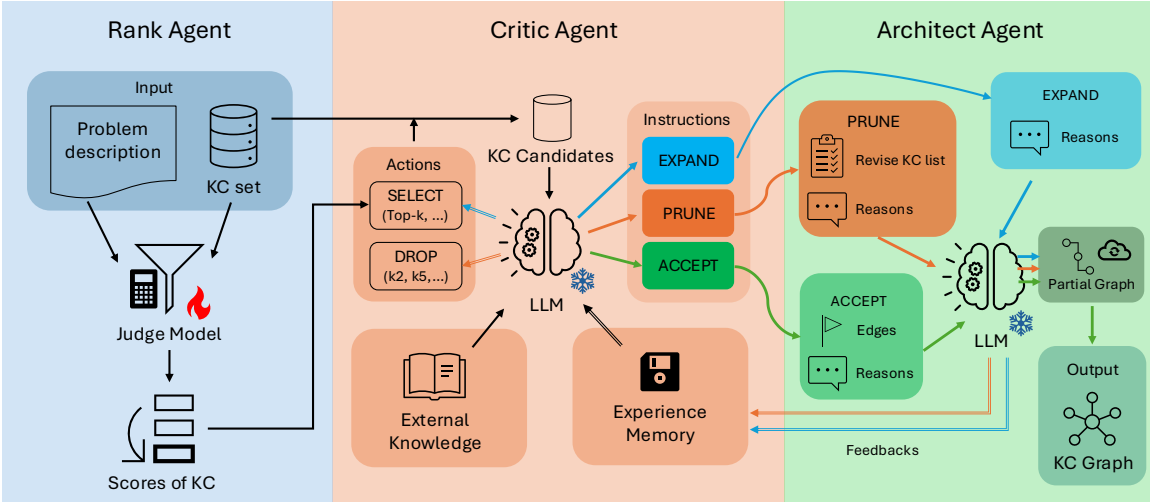

Figure 1: Overview of the ARC framework for iterative KC Graph Construction.

KC set $\mathcal{K}$ and the top-n candidates will be selected as the initial KC set:

$$\mathcal{K}_i^0 = \texttt{SELECT}(\mathcal{K}, \boldsymbol{\alpha}_i, n) \tag{3}$$

To train $\texttt{JM}_\theta$, we construct (problem, KC) pairs from the KC-Tagging dataset. Positive pairs are drawn from human annotations, while negative pairs are sampled from unannotated KCs for the same problem. We keep all data splits disjoint to avoid leakage across train, validation, and test sets. $\texttt{JM}_\theta$ is optimized with binary cross-entropy:

$$\begin{aligned}\mathcal{L}_{\text{BCE}} = -\sum_{(i,j)} \big[& y_{i,j} \log \sigma(\alpha_{i,j}) \\ & +(1-y_{i,j}) \log(1-\sigma(\alpha_{i,j}))\big]\end{aligned} \tag{4}$$

where $y_{i,j} = 1$ indicates k_j is required to solve p_i and 0 otherwise, σ is the `sigmoid` activation.

It should be noted that $\mathcal{A}_r$ accepts various implementations, such as Qwen3's Reranker or Embedding, which can compute compatibility scores between problem text and KC taxonomy entries.

3.3 Critic Agent

With the initial candidates $\mathcal{K}_i^0$, we need to check whether they are sufficient to solve the problem p_i or not. The Critic Agent $\mathcal{A}_c$ is designed to evaluate the current $\mathcal{K}_i^t$ and determine instructions to build the KC graph. It uses an `LLM` to produce a decision and a rationale with memorized experience:

$$\delta_i^t, r_i^t = \texttt{LLM}(p_i, \mathcal{K}_i^t, [\mathbf{P}]) \tag{5}$$

here, $\delta_i^t \in \{\texttt{EXPAND}, \texttt{PRUNE}, \texttt{ACCEPT}\}$ is the command instruction, r_i^t is the explanation. External Knowledge $\mathbf{P}$ can include the Guidebook manual and KC taxonomy text to help LLM to resolve KC ambiguity. If $\mathcal{A}_c$ does not finalize the prediction, it also consults the Experience Memory $\mathbf{E}_i^t$ to decide whether to retrieve more candidates or drop weak ones:

$$\beta_i^t, [\tilde{\mathcal{K}}_i^t] = \texttt{LLM}(\mathbf{E}_i^t), \tag{6}$$

where $\beta_i^t \in \{\texttt{SELECT}, \texttt{DROP}\}$. If $\beta_i^t = \texttt{SELECT}$, the current set will be refreshed by selecting the next top-n candidates from the ranked pool:

$$\mathcal{K}_i^{t+1} = \texttt{SELECT}(\mathcal{K}, \boldsymbol{\alpha}_i, n) \tag{7}$$

If $\beta_i^t = \texttt{DROP}$, the identified KCs will be instructed for pruning:

$$\mathcal{K}_i^{t+1} = \texttt{DROP}(\mathcal{K}_i^t, \tilde{\mathcal{K}}_i^t) \tag{8}$$

The iteration starts from $\mathcal{K}_i^0 = \emptyset$, so the first action is always `SELECT`. $\mathcal{A}_c$'s Experience Memory will be updated after each response back from $\mathcal{A}_a$:

$$\mathbf{E}_i^{t+1} = \texttt{UPDATE}(\mathbf{E}_i^t, f_i^t) \tag{9}$$

3.4 Architect Agent

The Architect Agent $\mathcal{A}_a$ maintains the partial KC graph $\mathcal{G}_i^t = (\mathcal{V}_i^t, \mathcal{E}_i^t)$ and updates it after receiving each instruction δ_i^t from $\mathcal{A}_c$. Given the instruction, rationale, and current graph, $\mathcal{A}_a$ generates the next graph state and optional feedback:

$$\mathcal{G}_i^{t+1}, [f_i^{t+1}] = \texttt{LLM}(\delta_i^t, r_i^t, \mathcal{G}_i^t) \tag{10}$$

The feedback signal f_i^t is produced when $\delta_i^t \in \{\texttt{PRUNE}, \texttt{EXPAND}\}$ and is used to update the Critic Agent's Experience Memory in the next step. The initial graph is empty, i.e., $\mathcal{V}_i^0 = \mathcal{E}_i^0 = \emptyset$.

Dataset	Problems	Avg KCs	Max KCs	Grade Range	Character
ASDiv	2000	∼1.0	2	K, 1-8	∼99% single-KC, sanity check domain
GSM8K	4999	∼1.4	7	K, 1-8	∼80% single-KC, moderate multi-label
MATH	3994	∼2.2	9	3-8, HS	∼61% single-KC, richest multi-label

Table 1: Dataset statistics used in the experiments. Avg KCs denotes the average number of Knowledge Components per problem.

4 Experiments

4.1 Datasets

We evaluate on the KC-tagging dataset (Tan et al., 2025), covering three MWP corpora under a shared CCSS KC taxonomy (e.g., `K.OA.A.1`). ASDiv is the easiest split, GSM8K is moderate, while MATH is the hardest. Detailed statistics are in Table 1.

We created a KC-stratified 70%/15%/15% split over KC combination signatures, reused across all conditions. For fine-tuning $\texttt{JM}_\theta$, we construct (problem, KC) binary pairs: positives from annotated sets, negatives from unannotated KCs for the same problem.

4.2 Implementation Details

We employ the Qwen3-4B (Yang et al., 2025) series models for Supervised Fine-tuning (SFT), and Prompted inference, while using GPT-4o (OpenAI, 2024) for Prompted baseline alternatives.

The $\texttt{JM}_\theta$ in $\mathcal{A}_r$ is fine-tuned with Qwen3-4B-Instruct-2507 as the backbone, serving as a `YES/NO` classifier. For directly Prompted approach and ARC, we use Qwen3-4B-Thinking-2507 and Qwen3-4B-Instruct-2507 locally via vllm. Experiments are conducted on a single NVIDIA H200 machine, on a dual-socket Intel Xeon Platinum CPU system with 128 logical cores.

4.3 Evaluation Metrics

We report three metrics, namely micro-averaged F1 ($\mathcal{M}_F$), Jaccard ($\mathcal{M}_J$), and exact match ratio (EMR). Let G_i be the Ground-truth KC set and P_i be the predicted KC set for problem p_i, over N total problems.

Micro-F1 $\mathcal{M}_F$ uses label-level counts aggregated across the full test set. Let TP_j, FP_j, and FN_j be the true positives, false positives, and false negatives for KC label $k_j \in L$, L is the label space:

$$\mathcal{M}_F = \frac{2\sum_j \mathrm{TP}_j}{2\sum_j \mathrm{TP}_j + \sum_j \mathrm{FP}_j + \sum_j \mathrm{FN}_j} \quad (11)$$

Jaccard Score $\mathcal{M}_J$ measures overlap between predicted and ground-truth:

$$\mathcal{M}_J = \frac{1}{N}\sum_{i=1}^{N} \frac{|G_i \cap P_i|}{|G_i \cup P_i|} \quad (12)$$

$\mathcal{M}_{emr}$ is 1 only when the predicted KC set exactly matches the ground-truth set:

$$\mathcal{M}_{emr} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[G_i = P_i] \quad (13)$$

4.4 Baseline Performance

To evaluate the baseline performance on the KC-Tagging, namely ASDiv-KC, GSM8K-KC, and MATH-KC (will omit -KC for brevity), we use the metrics defined above. It is important to note that all labels are human-annotated and may contain errors, particularly for harder problems in MATH where multiple KC assignments are plausible and inter-annotator agreement is lower. Consequently, reported metric values should be interpreted as performance relative to the human reference rather than an absolute ceiling of correctness. This partially explains why absolute scores are lower on MATH even for strong models, and why the gap between $\mathcal{M}_J$ and $\mathcal{M}_{emr}$ widens on harder datasets, since models often predict partially correct KC sets but rarely recover the exact annotated combination.

As shown in Table 2, SFT dominates on ASDiv and GSM8K, where its training distribution closely matches test problems and supervised signal is highly effective. On MATH, however, SFT's scores collapse ($\mathcal{M}_F$=0.246, $\mathcal{M}_{emr}$=0.077, the lowest among all methods). The steep EMR drop reveals that SFT tends to predict partial KC sets, it may recall some relevant labels but misses the full combinatorial structure required. We attribute this to MATH's higher label complexity (avg. 2.2 KCs, up to 9), which demands compositional reasoning over KC combinations rather than pattern recall. ARC's iterative expand-and-prune loop handles this setting better, as $\mathcal{A}_c$ can explicitly refine the KC set across iterations.

With 3-shot examples (Table 3), naive Prompted methods and ARC both improve on ASDiv and

Method	ASDiv (easy)			GSM8K (medium)			MATH (hard)		
	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$
SFT (Qwen3-4B)	**0.697**	**0.613**	**0.527**	**0.441**	**0.346**	**0.139**	0.246	0.163	0.077
Prompted (GPT-4o)	0.479	0.465	0.422	0.154	0.152	0.125	0.290	0.241	0.228
Prompted (Qwen3-4B-Instruct)	0.217	0.205	0.175	0.117	0.090	0.038	0.057	0.085	0.073
ARC (Qwen3-4B-Instruct)	0.496	0.480	0.378	0.279	0.219	0.049	**0.296**	**0.299**	**0.232**

Table 2: Baseline results with KC taxonomy and Guidebook text on 0-shot setting.

Method	ASDiv			GSM8K			MATH		
	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$
Prompted (GPT-4o)	0.499	0.467	**0.404**	0.177	0.175	**0.142**	**0.353**	**0.339**	**0.321**
Prompted (Qwen3-4B-Instruct)	0.333	0.323	0.276	0.144	0.127	0.071	0.125	0.136	0.130
ARC (Qwen3-4B-Instruct)	**0.568**	**0.515**	0.400	**0.295**	**0.216**	0.016	0.271	0.256	0.203

Table 3: Baseline results with KC taxonomy and Guidebook text on 3-shot setting.

GSM8K. ARC achieves the highest $\mathcal{M}_F$ (0.568) and $\mathcal{M}_J$ (0.515) on ASDiv, but its $\mathcal{M}_{emr}$ on GSM8K remains very low (0.016 vs. GPT-4o's 0.142), indicating that ARC retrieves more correct KCs but over-predicts by conservatively avoiding pruning. ARC still trails GPT-4o on MATH under 3-shot, where a few demonstrations cannot convey the required reasoning depth. Few-shot evaluation is not applicable to SFT.

To verify that MATH requires deeper reasoning, we switch the backbone to Qwen3-4B-Thinking (Table 4). Doing so consistently improves all three metrics across datasets and settings, with particularly notable EMR gains, the thinking model more often recovers the exact annotated KC set without over- or under-predicting. On MATH, the slight reversal $\mathcal{M}_J > \mathcal{M}_F$ in the 0-shot prompted setting reflects very precise predictions with few false positives, consistent with a model that traces the mathematical structure before committing to any KC.

4.5 Ablation Study

4.5.1 Judge model vs FAISS retrieval on $\mathcal{A}_r$

As mentioned in Sec. 3.2, Rank Agent $\mathcal{A}_r$ accepts various implementations to be functional. We compared two implementations: the fine-tuned Judge Model ($\texttt{JM}_\theta$) and Facebook AI Similarity Search (FAISS) (Douze et al., 2024) using a dense index built with Qwen3-4B-Embedding. Figure 2 shows the Coverage@k and Mean Recall@k curves for both methods across all three datasets. $\texttt{JM}_\theta$ achieves an overall AUC of 0.91–0.99 across datasets, while FAISS achieves 0.90–0.97. $\texttt{JM}_\theta$ outperforms FAISS on ASDiv and GSM8K by a clear margin, as FAISS relies purely on surface-level semantic similarity between problem text and KC descriptions, whereas $\texttt{JM}_\theta$ has been trained to recognize the functional relationship between a problem's mathematical structure and the KC required to solve it. On MATH, the two methods are more competitive in AUC terms, because MATH problems contain richer mathematical vocabulary that aligns more closely with KC descriptions, partially closing the gap between similarity-based and discriminative retrieval.

At the binary relevance threshold $\tau = 0.5$ as listed in Table 5, $\texttt{JM}_\theta$ attains substantially higher precision (0.75–0.85) compared to FAISS (0.33–0.62), while maintaining strong recall (0.59–0.91 vs. 0.96–0.99 for FAISS). FAISS's near-perfect recall comes at the cost of very low precision, it retrieves almost every gold KC but floods the candidate set with a large number of irrelevant ones, since all datasets have approximately 20% positive pairs. This noise is problematic for $\mathcal{A}_c$, which must reason over the provided candidates. The superior balance of $\texttt{JM}_\theta$ is reflected in its F1 scores of 0.66–0.88, significantly outperforming FAISS's 0.49–0.76.

Coverage@k is the critical metric for ARC: a gold KC missed by $\mathcal{A}_r$ can never be recovered downstream, since $\mathcal{A}_c$ can only select from the provided candidate pool. As shown in Table 6, $\texttt{JM}_\theta$ achieves $\geq$99% Coverage@k and Mean Recall@k across all datasets at $k = 10$, effectively guaranteeing that all gold KCs are present in the candidate set passed to $\mathcal{A}_c$. Even at $k = 10$, FAISS only reaches 0.311 on GSM8K, far below $\texttt{JM}_\theta$'s 1.000. This degradation stems from the greater lexical diversity of GSM8K problems, where the surface text

Method	Shot	ASDiv			GSM8K			MATH		
		$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$	$\mathcal{M}_F$	$\mathcal{M}_J$	$\mathcal{M}_{emr}$
Prompted (Qwen3-4B-Thinking)	0	0.600	0.598	0.553	0.285	0.264	0.199	0.337	0.343	0.337
Prompted (Qwen3-4B-Thinking)	3	0.629	0.623	0.571	0.365	0.343	0.271	0.300	0.297	0.280
ARC (Qwen3-4B-Thinking)	0	0.647	0.640	0.531	0.361	0.303	0.144	0.316	0.320	0.280
ARC (Qwen3-4B-Thinking)	3	0.655	0.641	0.527	0.369	0.311	0.156	0.371	0.369	0.313

Table 4: Results with Qwen3-4B-Thinking as backbone for both 0-shot and 3-shot setting.

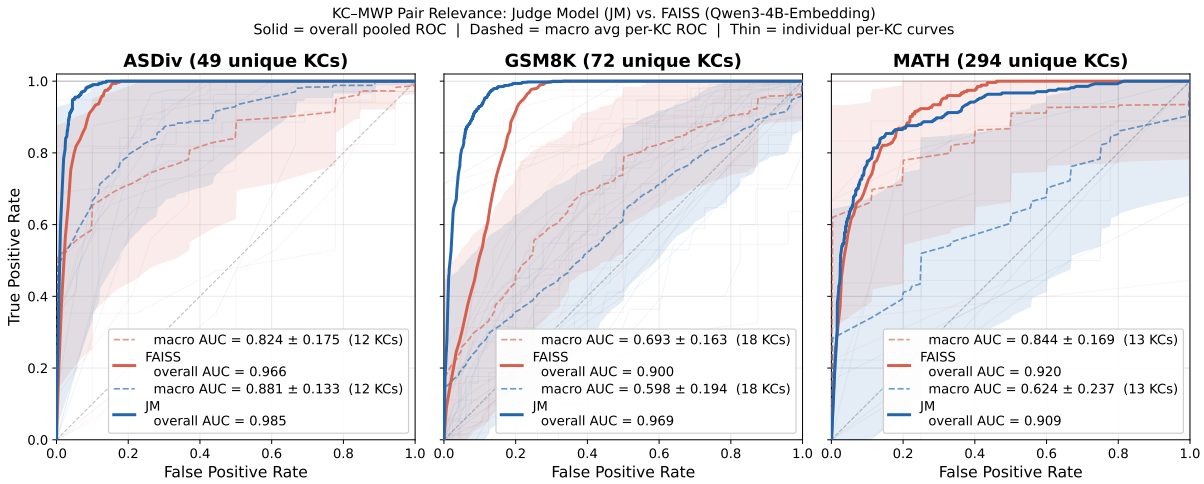

Figure 2: Coverage@k and Mean Recall@k for $\texttt{JM}_\theta$ vs. FAISS (Qwen3-4B-Embedding). $\texttt{JM}_\theta$ achieves near-perfect coverage at $k \leq 10$; FAISS requires far larger k to approach comparable coverage.

Dataset	$\mathcal{A}_r$	Acc.	Prec.	Rec.	F1
ASDiv	FAISS	0.874	0.615	**0.994**	0.760
	$\texttt{JM}_\theta$	**0.951**	**0.852**	0.914	**0.882**
GSM8K	FAISS	0.809	0.512	**0.959**	0.667
	$\texttt{JM}_\theta$	**0.924**	**0.784**	0.855	**0.818**
MATH	FAISS	0.657	0.325	**0.985**	0.489
	$\texttt{JM}_\theta$	**0.899**	**0.750**	0.591	**0.661**

Table 5: Binary relevance at $\tau = 0.5$ ($\alpha_{i,j} \geq 0.5$) vs. FAISS (top-50% of filtered list).

Dataset	$\mathcal{A}_r$	$C@5$	$\bar{R}@5$	$C@10$	$\bar{R}@10$
ASDiv	FAISS	0.724	0.764	0.851	0.868
	$\texttt{JM}_\theta$	**1.000**	**1.000**	**1.000**	**1.000**
GSM8K	FAISS	0.156	0.254	0.311	0.432
	$\texttt{JM}_\theta$	**0.997**	**0.998**	**1.000**	**1.000**
MATH	FAISS	0.553	0.573	0.659	0.686
	$\texttt{JM}_\theta$	**0.992**	**0.994**	**1.000**	**1.000**

Table 6: Coverage@k (all gold KCs in top-k) and Mean Recall@k for $\texttt{JM}_\theta$ vs. FAISS. A missed gold KC is a hard failure in ARC.

does not closely resemble the formal KC description, making dense embedding retrieval unreliable. These results confirm that $\texttt{JM}_\theta$ is the preferred ranking core for ARC, as its near-complete candidate coverage is a prerequisite for meaningful downstream agent reasoning.

4.5.2 The contribution of KC Taxonomy

Since the main results include KC taxonomy text as part of the External Knowledge, we ablate it to isolate its contribution (Table 7–Table 8). Removing KC taxonomy text consistently degrades performance across nearly all method-dataset combinations in both settings. The drop is especially pronounced for ARC: on GSM8K, $\mathcal{M}_F$ falls from 0.279 to 0.182 (0-shot) and from 0.295 to 0.258 (3-shot); on MATH, from 0.296 to 0.247 (0-shot). This is expected as CCSS codes such as `6.EE.A.1` are opaque identifiers with no intrinsic semantic content, and without accompanying definitions a model cannot meaningfully distinguish adjacent KCs in the taxonomy, leading to increased confusion and lower precision.

Interestingly, for Prompted GPT-4o on ASDiv, removing taxonomy text slightly *improves* $\mathcal{M}_F$ (0.516 vs. 0.479, 0-shot), which we attribute to GPT-4o's tendency to over-condition on the taxonomy descriptions and over-predict rare KCs when too much context is provided for a simple domain.

Method	KC Text	ASDiv $\mathcal{M}_F$	ASDiv $\mathcal{M}_J$	ASDiv $\mathcal{M}_{emr}$	GSM8K $\mathcal{M}_F$	GSM8K $\mathcal{M}_J$	GSM8K $\mathcal{M}_{emr}$	MATH $\mathcal{M}_F$	MATH $\mathcal{M}_J$	MATH $\mathcal{M}_{emr}$
Prompted (GPT-4o)	×	0.516	0.503	0.455	0.193	0.180	0.132	0.209	0.201	0.183
Prompted (GPT-4o)	✓	0.479	0.465	0.422	0.154	0.152	0.125	0.290	0.241	0.228
Prompted (Qwen3-4B-Instruct)	×	0.156	0.151	0.127	0.151	0.126	0.068	0.107	0.123	0.102
Prompted (Qwen3-4B-Instruct)	✓	0.217	0.205	0.175	0.117	0.090	0.038	0.057	0.085	0.073
ARC (Qwen3-4B-Instruct)	×	0.448	0.428	0.353	0.182	0.141	0.049	0.247	0.230	0.175
ARC (Qwen3-4B-Instruct)	✓	0.496	0.480	0.378	0.279	0.219	0.049	0.296	0.299	0.232

Table 7: KC Taxonomy Ablation (0-shot, with Guidebook).

Method	KC Text	ASDiv $\mathcal{M}_F$	ASDiv $\mathcal{M}_J$	ASDiv $\mathcal{M}_{emr}$	GSM8K $\mathcal{M}_F$	GSM8K $\mathcal{M}_J$	GSM8K $\mathcal{M}_{emr}$	MATH $\mathcal{M}_F$	MATH $\mathcal{M}_J$	MATH $\mathcal{M}_{emr}$
Prompted (GPT-4o)	×	0.494	0.448	0.378	0.286	0.298	0.251	0.261	0.260	0.248
Prompted (GPT-4o)	✓	0.499	0.467	0.404	0.177	0.175	0.142	0.353	0.339	0.321
Prompted (Qwen3-4B-Instruct)	×	0.265	0.255	0.215	0.207	0.177	0.084	0.128	0.128	0.110
Prompted (Qwen3-4B-Instruct)	✓	0.333	0.323	0.276	0.144	0.127	0.071	0.125	0.136	0.130
ARC (Qwen3-4B-Instruct)	×	0.488	0.422	0.295	0.258	0.199	0.035	0.195	0.180	0.150
ARC (Qwen3-4B-Instruct)	✓	0.568	0.515	0.400	0.295	0.216	0.016	0.271	0.256	0.203

Table 8: KC Taxonomy Ablation (3-shot, with Guidebook).

Despite these isolated exceptions, the overall trend is clear.

4.5.3 The contribution of Guidebook Manual

Another important part of External Knowledge is the Guidebook Manual, which was used by human annotators to label the data. As shown in Table 9–Table 10, the effect of Guidebook text is more nuanced than that of KC taxonomy. Not every method-dataset combination benefits from including it. For Prompted Qwen3-4B-Instruct, adding the Guidebook sometimes reduces performance (e.g., $\mathcal{M}_F$ drops from 0.292 to 0.217 on ASDiv, 0-shot), suggesting that the additional context introduces ambiguity or conflicts with the model's internal representations when the model capacity is limited. ARC consistently benefits from the Guidebook, with the most striking gains in EMR ($\mathcal{M}_{emr}$ improves from 0.013 to 0.049 on GSM8K and from 0.142 to 0.232 on MATH). The Guidebook contains human annotator guidance for resolving borderline KC assignments, and $\mathcal{A}_c$ is designed precisely for this kind of disambiguation. When multiple KCs are plausible, the Guidebook provides the tiebreaking criteria needed to recover the exact gold-standard label set, which explains why the benefit shows most clearly in EMR rather than in the more forgiving $\mathcal{M}_F$.

5 Educational Applications

In this paper, KCs are CCSS-defined labels for the mathematical concepts and operations a problem *requires*, not models of learner state. A flat KC set answers *which* concepts a problem touches, but nothing more. It cannot reveal the *order* in which those concepts must be applied, which concept is a *prerequisite* for another, or which is the likely failure point for a student who cannot solve the problem. ARC addresses this by producing a KC *graph*: directed edges encode the computational dependencies among the required KCs, turning an unordered label set into a structured representation of the problem's cognitive demands. Downstream ITS components can then interpret this structure for diagnosis, sequencing, and recommendation, and it uses that relational knowledge, not just label presence.

5.1 Illustrative Example

> *Jerry files a lawsuit . . . asking for damages for loss of a $50,000 annual salary for 30 years, $200,000 in medical bills, and punitive damages equal to triple the medical and salary damages. If he gets 80% of what he's asking for, how much money does he get?* `gsm8k_8077`

ARC is the only method to recover all three gold KCs and arrange them as a directed chain mirroring the computation:

`4.MD.A.2` → `5.NBT.B.5` → `6.RP.A.3c`

The edge rationale produced by $\mathcal{A}_a$: *"Money word problem requires* $\$50{,}000 \times 30$*; needs multi-digit multiplication"* (`4.MD.A.2`→`5.NBT.B.5`),

Method	Guide Text	ASDiv $\mathcal{M}_F$	ASDiv $\mathcal{M}_J$	ASDiv $\mathcal{M}_{emr}$	GSM8K $\mathcal{M}_F$	GSM8K $\mathcal{M}_J$	GSM8K $\mathcal{M}_{emr}$	MATH $\mathcal{M}_F$	MATH $\mathcal{M}_J$	MATH $\mathcal{M}_{emr}$
Prompted (GPT-4o)	×	0.496	0.468	0.415	0.140	0.140	0.117	0.245	0.185	0.175
Prompted (GPT-4o)	✓	0.479	0.465	0.422	0.154	0.152	0.125	0.290	0.241	0.228
Prompted (Qwen3-4B-Instruct)	×	0.292	0.275	0.222	0.110	0.097	0.041	0.073	0.109	0.081
Prompted (Qwen3-4B-Instruct)	✓	0.217	0.205	0.175	0.117	0.090	0.038	0.057	0.085	0.073
ARC (Qwen3-4B-Instruct)	×	0.508	0.437	0.196	0.293	0.205	0.013	0.282	0.268	0.142
ARC (Qwen3-4B-Instruct)	✓	0.496	0.480	0.378	0.279	0.219	0.049	0.296	0.299	0.232

Table 9: Guidebook Ablation (0-shot, with KC Taxonomy).

Method	Guide Text	ASDiv $\mathcal{M}_F$	ASDiv $\mathcal{M}_J$	ASDiv $\mathcal{M}_{emr}$	GSM8K $\mathcal{M}_F$	GSM8K $\mathcal{M}_J$	GSM8K $\mathcal{M}_{emr}$	MATH $\mathcal{M}_F$	MATH $\mathcal{M}_J$	MATH $\mathcal{M}_{emr}$
Prompted (GPT-4o)	×	0.507	0.453	0.353	0.197	0.194	0.153	0.339	0.323	0.309
Prompted (GPT-4o)	✓	0.499	0.467	0.404	0.177	0.175	0.142	0.353	0.339	0.321
Prompted (Qwen3-4B-Instruct)	×	0.318	0.321	0.276	0.159	0.139	0.060	0.142	0.151	0.134
Prompted (Qwen3-4B-Instruct)	✓	0.333	0.323	0.276	0.144	0.127	0.071	0.125	0.136	0.130
ARC (Qwen3-4B-Instruct)	×	0.529	0.443	0.200	0.266	0.184	0.011	0.344	0.338	0.252
ARC (Qwen3-4B-Instruct)	✓	0.568	0.515	0.400	0.295	0.216	0.016	0.271	0.256	0.203

Table 10: Guidebook Ablation (3-shot, with KC Taxonomy).

and *"Punitive damages are triple the total; then 80% settlement requires percent-of-quantity"* (`5.NBT.B.5`→`6.RP.A.3c`). In contrast, SFT appends a spurious Grade-2 addition KC (`2.NBT.B.7`) and returns a flat unordered set; Prompted Qwen3 misses the multiplication KC; GPT-4o picks the wrong percent variant (`6.RP.A.3` instead of `6.RP.A.3c`).

The prerequisite chain reveals cognitive structure that no flat KC set can express. It establishes that `5.NBT.B.5` must precede `6.RP.A.3c`, since computing base damages is a prerequisite for applying the percentage. It also pinpoints multi-digit multiplication as the likely failure point, giving an ITS a concrete entry for diagnosis. Once a student has mastered `5.NBT.B.5`, instruction can be directed to `6.RP.A.3c` alone, avoiding redundant review.

5.2 Downstream Applications

KC graphs produced by ARC support several downstream ITS applications. For prerequisite diagnosis, root KCs in the graph identify the foundational concepts a student must have mastered before attempting a problem. When a student fails, traversing the graph from the output KC back toward the roots surfaces the most likely gap, a more targeted diagnostic signal than inspecting an unordered label set. The topological order of the graph also supports curriculum sequencing: because edges encode which concept must be learned before another, a sequence of problems can be ordered so that each one only introduces KCs whose predecessors have already been covered. This enables data-driven scope-and-sequence design grounded in actual problem dependencies rather than expert heuristics. Beyond diagnosis and sequencing, graph structure offers richer signals for difficulty estimation. Graph depth captures the length of sequential reasoning chains, while branching captures the coordination of multiple independent concepts; together they provide a more informative representation of problem complexity than a scalar difficulty score. Finally, KC graph similarity enables practice recommendation based on conceptual structure. Problems that share subgraphs or closely related dependency patterns can be recommended to reinforce or extend a student's current reasoning trajectory, moving recommendation beyond surface-level keyword matching toward structural alignment of cognitive demands.

6 Conclusion

ARC reformulates KC prediction as a structured graph construction task, capturing both which concepts are required and how they depend on one another. Through iterative retrieval and critic–architect refinement, ARC outperforms supervised and prompted LLM baselines while producing interpretable KC dependency graphs that support prerequisite diagnosis, curriculum sequencing, and adaptive recommendation. Future work will integrate KC graphs into ITS pipelines and use ARC to generate synthetic annotations for low-resource domains.

Limitations

Scope of evaluation. ARC is evaluated exclusively on English math word problems annotated with CCSS-aligned KCs. Generalization to other subjects, languages, or non-CCSS taxonomies remains unvalidated, and porting ARC to a new domain requires re-annotating (problem, KC) pairs to train $\texttt{JM}_\theta$.

Edge evaluation. The KC-Tagging dataset provides flat KC set labels, no ground-truth for prerequisite edges exists. Consequently, the dependency structure of ARC's output graphs is evaluated only qualitatively in the application study, and the edge-level accuracy of ARC remains an open question.

References

Bilal Abu-Salih and Salihah Alotaibi. 2024. A systematic literature review of knowledge graph construction and application in education. *Heliyon*, 10(3):e25383.

Mofetoluwa Adeyemi, Akintunde Oladipo, Ronak Pradeep, and Jimmy Lin. 2024. Zero-shot cross-lingual reranking with large language models for low-resource languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–656, Bangkok, Thailand. Association for Computational Linguistics.

Miquel Canal-Esteve and Yoan Gutiérrez. 2024. Educational material to knowledge graph conversion: A methodology to enhance digital education. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 85–91, Bangkok, Thailand. Association for Computational Linguistics.

Yongwan Cho, Rabia Emhamed AlMamlook, and Tasnim Gharaibeh. 2024. A systematic review of knowledge tracing and large language models in education: Opportunities, issues, and future research. *arXiv preprint arXiv:2412.09248*.

Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. 2020. Ednet: A large-scale hierarchical dataset in education. In *Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part II*, page 69–73, Berlin, Heidelberg. Springer-Verlag.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Wenkang Han, Wang Lin, Liya Hu, Zhenlong Dai, Yiyun Zhou, Mengze Li, Zemin Liu, Chang Yao, and Jingyuan Chen. 2025. Contrastive cross-course knowledge tracing via concept graph guided knowledge transfer. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. 2019. Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 156–163.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the association for computational linguistics: EMNLP 2020*, pages 708–718.

OpenAI. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Daniel Reales, Rubén Manrique, and Christian Grévisse. 2024. Core concept identification in educational resources via knowledge graphs and large language models. *SN Comput. Sci.*, 5(8).

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 14918–14937.

Chor Seng Tan, Chengwei Wei, and Jung-Jae Kim. 2025. Towards scalable annotation of math word problems: Knowledge component tagging via llms and sentence embeddings. In *Proceedings of the 33rd International Conference on Computers in Education (ICCE 2025)*. Asia-Pacific Society for Computers in Education.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A Appendix: Examples of KC Graph

We present ten representative cases drawn from the GSM8K and MATH test sets to illustrate how ARC's iterative critic–architect loop produces outputs that are more educationally informative than flat KC predictions. Cases 1–8 are from MATH (difficulty Levels 1 through 5, with Levels 1, 3, and 4 each contributing two cases to illustrate distinct failure modes) using the thinking variant with 3-shot prompting. Cases 9–10 are drawn from GSM8K using the non-thinking 0-shot variant. Three baselines are compared: SFT (Qwen3-4B), Prompted (Qwen3-4B-Thinking), Prompted (GPT-4o) and ARC (Qwen3-4B-Thinking).

A.1 Case 1: Grade-level calibration.

> *Each successive term in the sequence $243, 81, x, y, 3, \ldots$ is obtained by multiplying the previous term by a constant. What is the value of $x + y$?* `math_236`

The sequence $243, 81, x, y, 3$ is a geometric sequence with constant ratio $\frac{1}{3}$, placing it in `HS.F-BF.A.2`. SFT is misled by the algebraic framing (x+y=?) and picks the equation-creation standard `HS.A-CED.A.1`. Prompted regresses to the Grade 4 pattern standard `4.OA.C.5`, which covers simple additive sequences but not geometric sequences defined by a constant multiplicative ratio. GPT-4o associates the constant ratio with exponents, erroneously invoking `8.EE.A.1` (integer exponent properties). ARC's critic recognises the defining characteristic, a constant multiplicative factor of $\frac{1}{3}$, as the signature of a geometric sequence and maps it to the correct high-school standard.

A.2 Case 2: Probability standard precision.

> *What is the probability that if you roll two fair, standard six-sided dice, the difference between the two numbers rolled will be 0? Express your answer as a common fraction.* `math_1306`

Rolling two dice and asking whether the difference equals 0 is a compound event problem at Grade 7 (`7.SP.C.8`: find probabilities via lists, tables, or tree diagrams). SFT maps this to two high-school standards (`HS.S-CP.A.1` and `HS.S-CP.B.9`), treating straightforward enumeration as a formal set-theory probability problem. Prompted correctly stays at Grade 7 but selects the simulation sub-standard (`7.SP.C.8b`) rather than the structured counting approach. GPT-4o picks the long-run frequency interpretation (`7.SP.C.5`). ARC recognises that the problem calls for theoretical probability over an enumerable sample space — precisely the scope of `7.SP.C.8`.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.F-BF.A.2 — Write arithmetic/geometric sequences recursively and explicitly	—
ARC (Qwen3-4B-Thinking) ✓	HS.F-BF.A.2	1.00
SFT (Qwen3-4B) ×	HS.A-CED.A.1 — Create equations in one variable (HS)	0.00
Prompted (Qwen3-4B-Thinking) ×	4.OA.C.5 — Generate number or shape patterns (Gr. 4)	0.00
Prompted (GPT-4o) ×	8.EE.A.1 — Properties of integer exponents (Gr. 8)	0.00

Table 11: Case 1 method comparison on KC prediction.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	7.SP.C.8 — Probability of compound events (Gr. 7)	—
ARC (Qwen3-4B-Thinking) ✓	7.SP.C.8	1.00
SFT (Qwen3-4B) ×	HS.S-CP.A.1 — Events as subsets of a sample space (HS) HS.S-CP.B.9 — Permutations and combinations (HS)	0.00
Prompted (Qwen3-4B-Thinking) ×	7.SP.C.8b — Represent sample spaces for compound events (Gr. 7)	0.00
Prompted (GPT-4o) ×	7.SP.C.5 — Probability as long-run relative frequency (Gr. 7)	0.00

Table 12: Case 2 method comparison on KC prediction.

A.3 Case 3: Abstention recovery with domain grounding.

> *Expand and simplify* $(x^2-5x+7)-(x-3)(x-2)$. `math_611`

Expanding $(x^2-5x+7)-(x-3)(x-2)$ requires distributing and combining like polynomial terms — the core operation defined by HS.A-APR.A.1. The SFT model produces no prediction. Prompted conflates the task with "rewriting an expression in equivalent form" (HS.A-SSE.B.3), missing that the required operation is explicit polynomial arithmetic rather than algebraic restructuring. GPT-4o dramatically underestimates the difficulty, assigning a Grade 6 numerical expression standard (6.EE.A.1). ARC's iterative loop recovers the correct answer where the SFT baseline abstains, the critic identifies that the task requires explicit polynomial multiplication and combination, mapping it to HS.A-APR.A.1.

A.4 Case 4: Prerequisite surfacing via dependency edge.

> *Simplify* $\sqrt{180}$. `math_11603`

Simplifying $\sqrt{180}$ requires factoring $180 = 36\times5$, then applying $\sqrt{36\times5} = 6\sqrt{5}$ — a radical rewriting task governed by HS.N-RN.A.2. Both Prompted and GPT-4o assign 8.EE.A.2, which concerns solving equations of the form $x^2 = p$ and introduces root notation, a conceptually different skill from simplifying a radical expression by factoring. The SFT model returns an empty prediction. ARC identifies the correct target KC and additionally surfaces a prerequisite: factoring 180 into 36×5 relies on finding factor pairs (4.OA.B.4). The directed edge makes explicit what must be mastered before the high-school radical standard: a student who cannot factor 180 cannot proceed to the simplification step regardless of whether they know the radical properties.

A.5 Case 5: Expression versus function framing.

> *What is the maximum value of* $-4z^2+20z-6$*?* `math_5910`

Both Prompted Qwen3 and Prompted GPT-4o converge on HS.F-IF.C.8a, which involves completing the square *in a quadratic function* to reveal extrema and symmetry on a graph. While conceptually adjacent, the gold standard is HS.A-SSE.B.3: choosing an equivalent form of an *algebraic expression* to reveal properties of the quantity it represents. The distinction is pedagogically meaningful: HS.F-IF.C.8a targets function analysis and graphical interpretation, while HS.A-SSE.B.3 targets the algebraic manipulation of choosing a strategically useful form. The problem asks for the maximum *value* of an expression, a purely algebraic question that does not require graphing, making HS.A-SSE.B.3 the correct framing. The SFT model offers no prediction; both prompted baselines commit the same function-framing error. ARC's critic resolves the distinction and identifies the expression-structure perspective.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.A-APR.A.1 — Add, subtract, and multiply polynomials (HS)	—
ARC (Qwen3-4B-Thinking) ✓	HS.A-APR.A.1	1.00
SFT (Qwen3-4B) ×	(empty prediction)	0.00
Prompted (Qwen3-4B-Thinking) ×	HS.A-SSE.B.3 — Choose equivalent form of expression to reveal properties (HS)	0.00
Prompted (GPT-4o) ×	6.EE.A.1 — Write and evaluate numerical expressions (Gr. 6)	0.00

Table 13: Case 3 method comparison on KC prediction.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.N-RN.A.2 — Rewrite expressions with radicals and rational exponents (HS)	—
ARC (Qwen3-4B-Thinking) ✓	HS.N-RN.A.2 4.OA.B.4 — Find factor pairs; prime and composite numbers (Gr. 4) *Edge:* 4.OA.B.4→HS.N-RN.A.2 *Reason:* "Factor 180 into 36×5"	0.67
SFT (Qwen3-4B) ×	(empty prediction)	0.00
Prompted (Qwen3-4B-Thinking) ×	8.EE.A.2 — Square and cube roots; solving $x^2 = p$ (Gr. 8)	0.00
Prompted (GPT-4o) ×	8.EE.A.2	0.00

Table 14: Case 4 method comparison on KC prediction.

A.6 Case 6: Polynomial division taxonomy.

> *When $3z^3 - 4z^2 - 14z + 3$ is divided by $3z + 5$, the quotient is $z^2 - 3z + \frac{1}{3}$. What is the remainder?* math_8844

Both Prompted Qwen3 and Prompted GPT-4o select HS.A-APR.B.2, the Remainder Theorem: the remainder when $p(x)$ is divided by $(x-a)$ equals $p(a)$. While related, the Remainder Theorem applies specifically to linear divisors of the form $(x-a)$; here the divisor is $3z+5$ and the full quotient is given, so the task is to verify that dividend $=$ (divisor)(quotient) $+$ remainder — the algebraic structure of polynomial long division encoded in HS.A-APR.D.6. SFT misses the polynomial domain entirely, predicting the equation-creation standard HS.A-CED.A.1. ARC correctly identifies the relevant skill as the formal structure of expressing a polynomial quotient: writing $a(z)/b(z) = q(z) + r(z)/b(z)$ with $\deg r < \deg b$.

A.7 Case 7: Cross-domain robustness.

> *For triangle ABC, points D and E are the midpoints of sides AB and AC, respectively. Side BC measures six inches. What is the measure of segment DE in inches?* math_2006

The Triangle Midsegment Theorem gives $DE = BC/2 = 3$ inches via similar triangle ratios, so the correct standard is HS.G-SRT.B.5, using similarity criteria to *solve problems*. SFT makes a dramatic cross-domain error, assigning an *algebraic* equation-creation standard (HS.A-CED.A.1) to a pure geometry task. Both Prompted and GPT-4o correctly identify the geometry subdomain but select proof standards (HS.G-CO.C.10 and HS.G-SRT.B.4) rather than the application standard: the task asks a student to *compute* a measurement, not to prove a theorem. ARC's critic resolves the prove-vs-solve distinction and identifies the correct application standard.

A.8 Case 8: Focused prerequisite graph versus over-prediction.

> *12 balls numbered 1 through 12 are placed in a bin. In how many ways can 3 balls be drawn, in order, from the bin, if the first ball is replaced after it is drawn but the second ball remains outside the bin?* math_4313

The problem asks for the count of ordered draw sequences under mixed replacement rules: $12 \times 12 \times 11$ outcomes. SFT includes the correct KC but appends three irrelevant standards (HS.A-CED.A.1, HS.F-BF.A.2, HS.S-CP.B.9), reducing F1 to 0.40. Prompted undershoots to a Grade 4 multi-step word problem standard, and GPT-4o overshoots to the high-school combinations standard. ARC produces a focused two-KC graph: 6.RP.A.3 (ratio and rate reasoning for multiplicatively enumerating ordered outcomes) as a prerequisite for 7.SP.C.8 (compound event probability). Unlike SFT's unstructured four-KC output, ARC's

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.A-SSE.B.3 — Choose equivalent form of expression to reveal properties (HS)	—
ARC (Qwen3-4B-Thinking) ✓	HS.A-SSE.B.3	1.00
SFT (Qwen3-4B) ×	(empty prediction)	0.00
Prompted (Qwen3-4B-Thinking) ×	HS.F-IF.C.8a — Write quadratic in vertex form to show extrema (HS)	0.00
Prompted (GPT-4o) ×	HS.F-IF.C.8a	0.00

Table 15: Case 5 method comparison on KC prediction.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.A-APR.D.6 — Rewrite rational expressions via long division (HS)	—
ARC (Qwen3-4B-Thinking) ✓	HS.A-APR.D.6	1.00
SFT (Qwen3-4B) ×	HS.A-CED.A.1 — Create equations in one variable (HS)	0.00
Prompted (Qwen3-4B-Thinking) ×	HS.A-APR.B.2 — Remainder Theorem (HS)	0.00
Prompted (GPT-4o) ×	HS.A-APR.B.2	0.00

Table 16: Case 6 method comparison on KC prediction.

graph encodes an instructional insight: students must apply multiplicative counting before they can reason about the probability of compound events.

A.9 Case 9: Perfect three-KC prediction with computational pipeline graph.

> *Jerry files a lawsuit against the convenience store where he works and slipped and fell. He's asking for damages for loss of a $50,000 annual salary for 30 years, $200,000 in medical bills, and punitive damages equal to triple the medical and salary damages. If he gets 80% of what he's asking for, how much money does he get?* gsm8k_8077

ARC is the only method to recover all three gold KCs and return them as a directed chain mirroring the computation: set up the money word problem (4.MD.A.2) → execute multi-digit multiplication for salary damages (5.NBT.B.5) → apply the 80% percentage (6.RP.A.3c). SFT gets the right KC set but appends a spurious Grade 2 addition standard (2.NBT.B.7) and returns a flat, unordered prediction. Prompted Qwen3 misses 5.NBT.B.5, substituting a generic ulti-step reasoning standard (4.OA.A.3); GPT-4o picks the wrong percent variant (6.RP.A.3 instead of 6.RP.A.3c) and similarly adds 4.OA.A.3. The KC chain produced by ARC maps directly onto lesson sequencing: a student weak at multi-digit multiplication will fail before reaching the percentage step.

A.10 Case 10: Complex branching graph for a multi-step financial problem.

> *Nick is choosing between two jobs. Job A pays $15 an hour for 2,000 hours a year in a state with a 20% total tax rate. Job B pays $42,000 a year in a state that charges $6,000 in property tax and a 10% tax rate on net income after property tax. How much more money will Nick make at the job with the higher net pay?* gsm8k_332

This problem requires parallel multi-step calculations for two different tax structures, making it one of the most KC-rich examples in the dataset. ARC recovers all three gold KCs and adds one contextual KC (4.MD.A.2), forming a branching graph with three directed edges: subtract property tax (4.NBT.B.4) → apply the 10% rate on net income (6.RP.A.3c) → money context (4.MD.A.2); in parallel, multiply wage×hours (5.NBT.B.5) → money context (4.MD.A.2). SFT includes two erroneous KCs: 4.NBT.B.5 (multiplication) instead of 4.NBT.B.4 (subtraction), and 6.RP.A.3 instead of 6.RP.A.3c — errors that would direct a teacher to the wrong lesson. GPT-4o reduces the problem to a single percentage standard, missing the multiplication and subtraction operations entirely.

B Prompt templates used in this paper

B.1 Prompted method, System Prompt

```
You are an expert mathematics education specialist for
↪ Knowledge Component (KC) tagging.
Given a math question, return only a JSON object with key
↪ `labels`.
The value must be a JSON array of KC labels (order does not
↪ matter).
Use only labels from this list and their definitions:
```

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	HS.G-SRT.B.5 — Use similarity/congruence criteria to solve problems (HS)	—
ARC (Qwen3-4B-Thinking) ✓	HS.G-SRT.B.5	1.00
SFT (Qwen3-4B) ×	HS.A-CED.A.1 — Create equations in one variable (HS)	0.00
Prompted (Qwen3-4B-Thinking) ×	HS.G-CO.C.10 — Prove theorems about triangles (HS)	0.00
Prompted (GPT-4o) ×	HS.G-SRT.B.4 — Prove theorems about similar triangles (HS)	0.00

Table 17: Case 7 method comparison on KC prediction.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	7.SP.C.8 — Probability of compound events (Gr. 7)	—
ARC (Qwen3-4B-Thinking) ✓	7.SP.C.8 6.RP.A.3 — Ratio and rate reasoning for real-world problems (Gr. 6) *Edge:* 6.RP.A.3→7.SP.C.8 *Reason:* "Multiplicative outcome counting underpins the compound event probability."	0.67
SFT (Qwen3-4B) ×	7.SP.C.8, HS.A-CED.A.1, HS.F-BF.A.2, HS.S-CP.B.9	0.40
Prompted (Qwen3-4B-Thinking) ×	4.OA.A.3 — Solve multi-step word problems (Gr. 4)	0.00
Prompted (GPT-4o) ×	HS.S-CP.B.9 — Permutations and combinations (HS)	0.00

Table 18: Case 8 method comparison on KC prediction.

```
{definition_block}

Output format exactly: {"labels": ["label1", "label2"]}
If no label is appropriate, return an empty array.
{guidebook_block}
{few_shot_examples}
```

B.2 Prompted method, User Prompt

```
Question:
{text}
```

B.3 ARC, Critic Agent, System Prompt

```
You are the Critic in a two-role KC pipeline (Critic +
↪  Architect).
Goal: identify exactly which KCs are required to solve the math
↪  problem from ranked candidates.

Follow these steps:
1. REASONING — Think through the solution approach: what
↪  mathematical operations and concepts are needed?
2. EVALUATE — For each candidate KC (shown with its definition
↪  and judge score), decide whether your solution requires it.
3. CONFIRM each required KC with a brief reason; DROP each
↪  unrelated KC.
4. Decide: drop (list drop_kcs), extend (set request_more=true
↪  for more candidates), or finalize (set finalize=true).
5. Follow the guidebook rules — they define the annotation
↪  standard.
Every action must include a short reason.
{few_shot_section}

Return JSON only:
{
  "actions": [
    {"action": "CONFIRM", "kc": "4.NBT.B.5", "reason": "..."},
    {"action": "DROP", "kc": "3.OA.A.1", "reason": "..."}
  ],
  "decision": "extend",
  "reason": "Need additional candidates for missing
  ↪  operations.",
  "request_more": true,
  "request_count": 10,
  "drop_kcs": ["3.OA.A.1"],
  "finalize": false,
  "final_graph": {"kcs": [], "edges": []}
}
Rules:
- decision=drop: explicitly list KCs in drop_kcs and include
↪  DROP actions.
- decision=extend: set request_more=true.
- decision=finalize: set finalize=true and optionally provide
↪  final_graph.
```

B.4 ARC, Critic Agent, User Prompt

```
Problem:
{problem_text}

{guide_section}

{architect_section}Confirmed KCs:
{confirmed_kc_block or "(none yet)"}

Dropped KCs:
{dropped_kc_block or "(none yet)"}

Candidate KCs (ranked by judge_score):
{candidate_lines_with_def or "(none)"}

Return JSON.
```

B.5 ARC, Architecture Agent, System Prompt

```
You are the Architect in a two-role KC pipeline.
Input: Critic decision/actions + current graph state.
Tasks:
1) Build/adjust dependency edges between confirmed KCs.
2) Decide one of: drop, extend, finalize.
3) Provide concise reasons.
Return JSON only in this schema:
{
  "decision": "extend",
  "reason": "Need one more operation-level KC.",
  "drop_kcs": [{"kc": "3.OA.A.1", "reason": "Not required"}],
  "edges": [{"from": "4.NBT.B.4", "to": "4.NBT.B.5", "reason":
  ↪  "..." }],
  "final_graph": {"kcs": [], "edges": []}
}
```

B.6 ARC, Architecture Agent, User Prompt

```
Problem:
{problem_text}

{guide_section}

Confirmed KCs:
{confirmed_kc_block or "(none yet)"}

Dropped KCs:
{dropped_kc_block or "(none yet)"}

Current partial graph:
{json(partial_graph)}

Critic payload (this round):
{json(critic_payload)}
```

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	4.MD.A.2 — Word problems with money and operations (Gr. 4) 5.NBT.B.5 — Fluently multiply multi-digit whole numbers (Gr. 5) 6.RP.A.3c — Find a percent of a quantity (Gr. 6)	—
ARC (Qwen3-4B-Thinking) ✓	4.MD.A.2, 5.NBT.B.5, 6.RP.A.3c *Edge 1:* 4.MD.A.2→5.NBT.B.5 *Reason:* "Money word problem requires $50,000×30; needs multi-digit multiplication." *Edge 2:* 5.NBT.B.5→6.RP.A.3c *Reason:* "Punitive damages are triple the total; then 80% settlement requires percent-of-quantity."	1.00
SFT (Qwen3-4B) ×	4.MD.A.2, 5.NBT.B.5, 6.RP.A.3c, 2.NBT.B.7 (extra)	0.86
Prompted (Qwen3-4B-Thinking) ×	4.OA.A.3, 6.RP.A.3c	0.50
Prompted (GPT-4o) ×	6.RP.A.3, 4.OA.A.3	0.00

Table 19: Case 9 method comparison on KC prediction.

Method	Predicted KC (set, graph)	F1
GT (Human Annotator)	5.NBT.B.5 — Fluent multi-digit multiplication (Gr. 5) 6.RP.A.3c — Find a percent of a quantity (Gr. 6) 4.NBT.B.4 — Fluent multi-digit addition/subtraction (Gr. 4)	—
ARC (Qwen3-4B-Thinking) ✓	4.MD.A.2, 5.NBT.B.5, 6.RP.A.3c, 4.NBT.B.4 *Edge 1:* 4.NBT.B.4→6.RP.A.3c *Reason:* "Subtract property tax first; then apply 10% rate on net income." *Edge 2:* 5.NBT.B.5→4.MD.A.2 *Reason:* "Multiply hourly wage×hours for gross income in a money word-problem context." *Edge 3:* 6.RP.A.3c→4.MD.A.2 *Reason:* "Percent tax on net income applied within money problem context."	0.86
SFT (Qwen3-4B) ×	4.MD.A.2, 4.NBT.B.5, 5.NBT.B.5, 6.RP.A.3, 6.RP.A.3c	0.57
Prompted (Qwen3-4B-Thinking) ×	4.OA.A.3, 6.RP.A.3c	0.40
Prompted (GPT-4o) ×	6.RP.A.3	0.00

Table 20: Case 10 method comparison on KC prediction.

Return JSON.