K-MASTER: A Training-Free Multi-Agent Framework of Knowledge Concept Tagging for Course Questions

Anonymous ACL submission

Abstract

Automatic knowledge concept tagging assigns educational questions to one or more concepts in a curriculum knowledge system. It is a core component of question-bank management, personalized recommendation, and adaptive tutoring. Supervised tagging models can achieve strong performance when large annotated corpora and stable knowledge taxonomies are available, but they are difficult to transfer to new subjects, revised curricula, or long-tail concepts. Large language models (LLMs) offer a training-free alternative. However, direct prompting with large label sets often leads to predictions outside the taxonomy. Existing discriminative evaluation methods reduce tagging to repeated binary decisions, which is costly for large taxonomies and does not support joint label ranking. In this paper, we propose **K-MASTER**, a training-free multi-agent framework for knowledge concept tagging. K-MASTER decomposes tagging into a generate–retrieve–evaluate loop: a generator proposes open-form concept descriptions, a retriever maps them to a standard knowledge taxonomy through subject routing and hybrid retrieval, and an evaluator filters candidates and provides feedback for iterative refinement. Experiments on six datasets across four subjects show that K-MASTER achieves a favorable balance among accuracy, efficiency, and adaptability. It outperforms a discriminative multi-agent baseline and remains more robust than a supervised model on low-frequency labels.

1 Introduction

The digital transformation of education has rapidly increased the volume of learning resources, test items, and learner-behavior data (Wang et al., 2024; Fischer et al., 2020). Educational platforms therefore require structured question banks to support intelligent indexing, personalized recommendation, and adaptive learning. Within this infrastructure, automatic knowledge concept tagging assigns each question one or more concepts from a predefined knowledge taxonomy, serving as a critical bridge between unstructured question content and downstream educational services.

Most existing supervised models rely on large quantities of high-quality annotations and are usually tied to a fixed subject and knowledge taxonomy (Sun et al., 2018; Huang et al., 2023b; Ding et al., 2025). When the target subject changes or the knowledge taxonomy is updated, these models typically require additional annotation and retraining. LLMs offer an attractive path toward training-free tagging because of their strong general reasoning capabilities (Li et al., 2024a, 2025), but applying them directly to a predefined knowledge taxonomy remains unreliable. A generative prompt that asks the model to choose from a long list of candidate labels may still produce concepts outside the taxonomy. A discriminative formulation that independently judges whether each question–concept pair is relevant avoids hallucinated labels, but its inference cost grows linearly with the size of the taxonomy and it cannot model the final concept set jointly. These limitations are especially severe in multi-subject settings, where the label space may contain hundreds of fine-grained concepts.

To address these issues, we propose **K-MASTER** (**K**nowledge **M**ulti-**A**gent **S**ystem for **T**agging, **E**valuation and **R**efinement), a training-free multi-agent framework for educational knowledge concept tagging. K-MASTER consists of three cooperative agents: the *generator* infers open-form concept descriptions from the question and answer; the *retriever* maps these descriptions into a standard knowledge taxonomy through subject routing, hybrid retrieval, and candidate filtering; and the *evaluator* checks relevance and coverage and feeds structured feedback into the next iteration. This design keeps final predictions inside the predefined knowledge taxonomy while avoiding exhaustive LLM judgments over all labels.

Our work makes three main contributions:

- We formulate training-free knowledge concept tagging as a closed-loop multi-agent process that combines concept generation, label retrieval, and quality-controlled evaluation.
- We design a routed hybrid retriever that selects subject-specific concept collections and dynamically combines embedding retrieval, BM25 matching, and graph-based expansion.
- Experiments on six datasets across four subjects show that K-MASTER achieves a favorable trade-off among accuracy, efficiency, and adaptability. It is substantially more efficient than a discriminative multi-agent baseline and more robust than a supervised model on low-frequency labels.

2 Related Work

Early work on knowledge concept tagging relied on manually designed rules or expert-built semantic patterns. Such methods can be effective in constrained domains, but their coverage depends on the completeness of the rule base and they transfer poorly to new subjects (Mohasseb et al., 2018). Traditional machine learning methods reduce manual annotation effort by training classifiers on engineered text features (Mandal and Naskar, 2021). However, their effectiveness is limited by the quality of feature engineering, and they have a limited ability to capture deep semantic correspondence between questions and knowledge concepts.

Deep learning methods improve automatic representation learning with recurrent networks, attention mechanisms, and pretrained language models. Sun et al. (2018) enhance question representations with position or keyword attention, Huang et al. (2023b) jointly model questions and answers, and Huang et al. (2023a) reformulate tagging as a sequence generation task. Viswanathan et al. (2024); Ding et al. (2025) further use label-aware attention to exploit concept structures. These methods generally outperform earlier approaches, but they remain supervised and are often coupled to a specific dataset and knowledge taxonomy.

Recent LLM-based methods reduce the dependence on labeled data. Several authors directly use LLMs for knowledge concept tagging by formulating the task as either discriminative matching or generative prediction. Li et al. (2024a) cast knowledge concept tagging as a discriminative task, using LLMs to determine whether a question and a candidate concept are matched. Ozyurt et al. (2024) apply LLMs to knowledge component tagging in knowledge tracing, where the model first generates solution steps and then assigns knowledge components based on the generated reasoning process. Moore et al. (2024) use systematic prompt engineering to support knowledge component tagging and generation for chemistry multiple-choice questions. Li et al. (2024b) further propose a reinforcement-learning-based example selection method to improve few-shot learning and strengthen the contextual understanding of LLMs. Li et al. (2025) introduce a multi-agent workflow that decomposes knowledge concept tagging into planning, solving, semantic judgment, and numerical judgment, and uses collaboration among multiple agents to determine whether a question matches a given knowledge concept. Despite their flexibility, these methods still face two practical limitations: direct generation can be weakly grounded in the standard knowledge taxonomy, while pairwise discriminative judging is expensive for large label spaces and lacks the ability to rank labels for multi-label annotation. K-MASTER addresses both issues through the collaborative design of the generator, retriever, and evaluator.

3 Method

3.1 Problem Definition

Given a question text $\mathbf{Q}$, its answer $\mathbf{S}$, and a predefined knowledge taxonomy containing k candidate concepts $\mathbf{L} = \{l_1, l_2, \ldots, l_k\}$, the goal is to predict a subset $\mathbf{Y} = \{y_1, y_2, \ldots, y_m\} \subseteq \mathbf{L}$ that best matches the educational content of the question. We seek a training-free workflow Φ such that $\Phi(\mathbf{Q}, \mathbf{S}, \mathbf{L}) \rightarrow \mathbf{Y}, \mathbf{Y} \subseteq \mathbf{L}$.

3.2 System Overview

To address these issues, we propose **K-MASTER**, which adopts a generate–retrieve–evaluate loop for educational knowledge concept tagging. As shown in Figure 1, at iteration t, the generator generates an open-form concept set $\mathbf{Y}_t^{gen}$, the retriever maps these concepts into the standard knowledge taxonomy to produce a candidate set $\mathbf{Y}_t^{ret}$, and the evaluator returns an accepted subset $\mathbf{Y}_t$, feedback $\mathbf{F}_t$, a decision $d_t \in \{\text{PASS}, \text{FAIL}\}$, and a round score s_t.

The framework maintains a shared history $\mathbf{H}_t = (\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_t)$, where each memory item

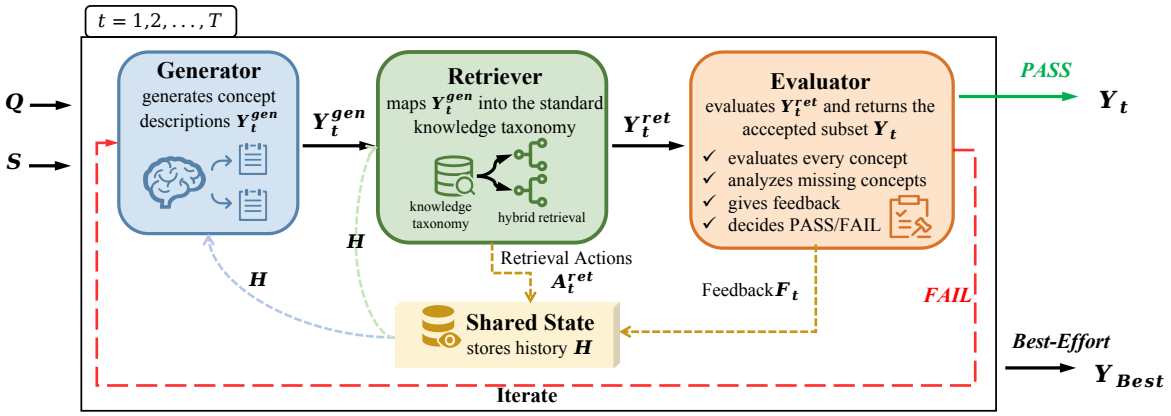

Figure 1: Overall architecture of K-MASTER. The framework maintains a shared state across the generator, retriever, and evaluator, enabling feedback-driven refinement across iterations.

records generated concepts, retrieved examples, retrieval actions, retrieved candidates, accepted concepts, feedback, score, and round decision. This shared state gives later iterations access to concrete failure evidence, such as an incorrect subject route, unmapped generated concepts, irrelevant retrieved labels, or missing core concepts.

The evaluator computes a round score

$$s_t = \begin{cases} \dfrac{\sum_{i \in \mathbf{Y}_t} q_i}{|\mathbf{Y}_t^{ret}|}, & |\mathbf{Y}_t^{ret}| > 0, \\ 0, & |\mathbf{Y}_t^{ret}| = 0, \end{cases} \tag{1}$$

where $q_i \in [0, 10]$ is the evaluator's quality score for accepted candidate i. The score rewards high-quality accepted labels while penalizing noisy candidate pools through the denominator. If any round passes, its accepted set becomes the final output. Otherwise, after the maximum number of iterations T, K-MASTER enters a best-effort fallback and returns the highest-scoring historical round:

$$t^* = \arg \max_{t \in \{1,\ldots,T\}} s_t, \qquad \mathbf{Y} = \mathbf{Y}_{t^*}. \tag{2}$$

Algorithm 1 summarizes the complete workflow. For compactness, we abbreviate the generator, retriever, and evaluator as Gen, Ret, and Eval, respectively, and use superscripts g and r for generated and retrieved quantities.

3.2.1 Generator

The generator generates an initial set of concept descriptions $\mathbf{Y}_t^{gen}$ from the question $\mathbf{Q}$, the reference answer $\mathbf{S}$, and the shared history $\mathbf{H}_{t-1}$. It leverages the LLM's semantic understanding to propose a small set of open-form concepts that describe the main educational content of the question.

Algorithm 1 K-MASTER tagging workflow.

Require: question $\mathbf{Q}$; answer $\mathbf{S}$; taxonomy $\mathbf{L}$; max rounds T.
Ensure: predicted concept set $\mathbf{Y}$.

```
H_0 ← ∅
for t = 1 to T do
    (E_t, Y_t^g) ← Gen(Q, S, H_{t-1})
    (A_t^r, Y_t^r) ← Ret(Q, Y_t^g, L, H_{t-1})
    (Y_t, F_t, s_t, d_t) ← Eval(Y_t^r, Q, S, E_t)
    M_t ← (E_t, Y_t^g, A_t^r, Y_t^r, Y_t, F_t, s_t, d_t)
    H_t ← H_{t-1} ∪ {M_t}
    if d_t = PASS then
        return Y_t
    end if
end for
t* ← arg max_{t∈{1,...,T}} s_t
return Y_{t*}
```

These concepts are subsequently grounded by the retriever, which prevents the final prediction from containing out-of-taxonomy labels.

The generator is augmented with retrieved few-shot examples. Specifically, we retrieve several training questions semantically similar to the current question and include their gold concept labels as examples $\mathbf{E}_t$. These examples provide an explicit reference for the desired granularity and naming style of concept labels. They are also passed to the evaluator, so that generation and evaluation are conditioned on the same annotation standard.

In later iterations, the generator additionally reads the evaluator feedback $\mathbf{F}_{t-1}$ and the retriever action records $\mathbf{A}_{t-1}^{ret}$ from the shared history. The feedback contains unmappable generated concepts, rejected candidates, missing concepts, and

a natural-language critique. The prompt instructs the generator to avoid previously rejected or unmappable concepts, to cover the identified missing concepts, and to adjust its overall generation strategy according to the critique.

3.2.2 Retriever

The retriever maps the generator's open-form concepts $\mathbf{Y}_t^{gen}$ to the predefined knowledge taxonomy $\mathbf{L}$ and returns a compact candidate set $\mathbf{Y}_t^{ret}$ for evaluation. It also records its intermediate decisions as an action trace $\mathbf{A}_t^{ret}$, so that later iterations can revise routing and retrieval strategies based on concrete failures. The retrieval process has three stages: collection routing, strategy planning and execution, and preliminary candidate filtering.

Subject routing. K-MASTER maintains multiple subject- or curriculum-specific concept collections. Before retrieval, the retriever selects a target collection $\mathbf{C}_t$ based on the question $\mathbf{Q}$ and collection descriptions. The router follows a subject-first and curriculum-stage-second rule: it first identifies the broad subject and then resolves finer stages when necessary, such as elementary, middle-school, or high-school mathematics. In later iterations, failed retrieval concepts, rejected candidates, missing concepts, and the evaluator's critique can trigger route correction.

Hybrid retrieval planning. After routing, the retriever plans retrieval strategies for each generated concept $y_{t,i}^{gen}$. Each action is represented as $a_{t,i}^{ret} = (y_{t,i}^{gen}, \mathbf{C}_t, \mathbf{P}_{t,i}, \mathbf{Y}_{t,i}^{map})$, where $\mathbf{P}_{t,i}$ is the selected strategy set and $\mathbf{Y}_{t,i}^{map}$ is the mapped label set. The planner chooses from three tools:

- **BM25 keyword retrieval** is used for proper nouns, formula names, symbolic expressions, and other concepts with strong lexical cues.
- **Embedding retrieval** is used for descriptive concepts that require semantic matching.
- **Graph 1-hop expansion** uses the concept co-occurrence graph to retrieve neighboring concepts, which provides coverage-oriented expansion when direct retrieval is insufficient.

In later iterations, the planner inspects previous zero-hit cases, rejected results, missing-concept feedback, and the evaluator's critique to adjust the strategy set. The mapped results from all selected tools are merged into $\mathbf{Y}_{t,i}^{map}$, and the full trace is $\mathbf{A}_t^{ret} = \{a_{t,i}^{ret}\}_{i=1}^{|\mathbf{Y}_t^{gen}|}$.

Candidate filtering. All mapped labels are merged into a candidate pool, which may contain duplicates and noisy labels. The retriever therefore applies a preliminary LLM-based filtering step to obtain $\mathbf{Y}_t^{ret}$. This step removes obvious noise, improves global ranking in the multi-label setting, and keeps the evaluator input compact.

3.2.3 Evaluator

The evaluator is the final quality-control module. Given the retrieved candidate set $\mathbf{Y}_t^{ret}$, the question $\mathbf{Q}$, the answer $\mathbf{S}$, and the few-shot examples $\mathbf{E}_t$, it evaluates both candidate-level accuracy and set-level coverage. The few-shot examples provide the same annotation standard used by the generator, allowing the evaluator to judge relevance, granularity, and naming consistency.

For accuracy scoring, the evaluator reviews each candidate in $\mathbf{Y}_t^{ret}$. Relevant candidates are retained in $\mathbf{Y}_t \subseteq \mathbf{Y}_t^{ret}$ and assigned quality scores $q_i \in [0, 10]$, while irrelevant candidates are recorded as bad cases $\mathbf{B}_t$ with rejection reasons. It then performs gap analysis, and uncovered core concepts are collected as missing concepts $\mathbf{G}_t$.

Finally, the evaluator issues a decision $d_t \in \{\text{PASS}, \text{FAIL}\}$ and produces a natural-language critique c_t. PASS means that the accepted concepts cover the main knowledge concepts without serious errors, whereas FAIL indicates critical errors or important omissions. The feedback is $\mathbf{F}_t = (\mathbf{B}_t, \mathbf{G}_t, c_t)$, and the outputs $(\mathbf{Y}_t, \mathbf{F}_t, s_t, d_t)$ are written into the shared history to guide subsequent generation, retrieval, and routing refinement.

The prompt templates for all modules are provided in Appendix A.

4 Experiments

The experiments aim to address the following research questions:

- **RQ1:** How does K-MASTER perform with different backbone LLM configurations?
- **RQ2:** How does K-MASTER compare with supervised and multi-agent baselines?
- **RQ3:** What are the effects of the key components in K-MASTER?
- **RQ4:** How effective is K-MASTER's routing mechanism in a unified multi-subject knowledge space?

Table 1: Statistics of the cleaned datasets. Math-PM, Math-MD, and Math-HS refer to elementary, middle-school, and high-school mathematics, respectively.

Dataset	Domain	#Questions	#Concepts
CCNU	Math-PM	9,091	98
GeoQA	Math-MD	4,950	49
DA-20K	Math-HS	22,498	427
Biology	Biology	14,260	400
Geography	Geography	7,187	116
History	History	4,970	234

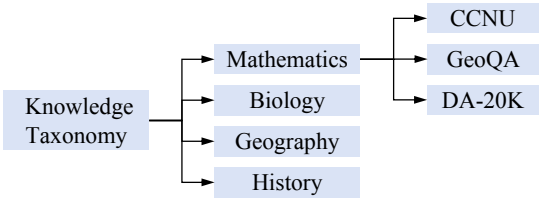

Figure 2: Unified multi-subject concept tree.

4.1 Datasets

We evaluate K-MASTER on six open-source Chinese datasets covering four subjects: Biology, Geography, History[1], and Mathematics. The mathematics datasets cover three curriculum levels: CCNU for elementary mathematics (Huang et al., 2023b), GeoQA for middle-school geometry (Chen et al., 2021), and DA-20K for high-school mathematics (Ding et al., 2025). The original DA-20K dataset does not include answers, so we use GPT-4 (Achiam et al., 2023) to generate answer texts. Table 1 summarizes the cleaned datasets. Detailed data distributions are provided in Appendix B.

To test cross-subject transfer, we merge all label spaces into a unified multi-subject concept tree containing 1,324 leaf concepts, as shown in Figure 2. This setting requires the system to resolve both coarse subject boundaries and fine-grained curriculum-stage distinctions within mathematics.

4.2 Baselines

We compare K-MASTER with two baselines. **LHABS** (Ding et al., 2025) is a supervised deep model trained jointly on the unified multi-subject label space. **MAS** (Li et al., 2025) is a discriminative multi-agent method that judges question–concept pairs independently. Because the MAS code is not publicly available, we reimplement it according to the original paper and convert its binary decisions into relevance scores for Top-K evaluation.

[1] https://github.com/jiangnanboy/knowledge-automatic-tagging

Table 2: Backbone LLM configurations. CoT indicates whether chain-of-thought reasoning is enabled.

Config.	Model ID	Arch.	CoT
8B-Instruct	Qwen3-8B	Dense	No
8B-Think	Qwen3-8B	Dense	Yes
30B-A3B-Instruct	Qwen3-30B-A3B-Instruct-2507	MoE	No
30B-A3B-Think	Qwen3-30B-A3B-Thinking-2507	MoE	Yes
32B-Instruct	Qwen3-32B	Dense	No
32B-Think	Qwen3-32B	Dense	Yes

4.3 Backbone Models

We evaluate six Qwen3 backbone configurations (Yang et al., 2025) that vary in scale, architecture, and chain-of-thought (CoT) setting (Wei et al., 2022). As shown in Table 2, the 8B and 32B models are dense models, while 30B-A3B is a mixture-of-experts model with approximately 3B activated parameters. Instruct mode generates the final response directly, whereas Think mode enables CoT before generating the final answer.

4.4 Metrics and Implementation Details

We evaluate predictions using P@K, R@K, and F1@K for $K \in \{1, 3, 5\}$, MRR, and average per-sample inference time. Each dataset is split into training and test sets at a 75:25 ratio. The training split is used to train LHABS and to construct a few-shot example bank and a concept co-occurrence graph for K-MASTER. For experiments averaged over all six datasets, we randomly sample 800 questions from each dataset's test split. For comparison with existing methods, we evaluate on the full test splits of CCNU and GeoQA. We define low-frequency concepts as the tail 10% of cumulative label occurrences in the training split and evaluate on test samples containing these concepts. Appendix C gives the low-frequency concepts subset statistics. All experiments are conducted on four NVIDIA A100 GPUs. We use SGLang (Zheng et al., 2024) for LLM inference, LangGraph for agent orchestration[2], ChromaDB[3] and Qwen3-Embedding-0.6B (Zhang et al., 2025) for vector retrieval, BM25 for lexical retrieval, and NetworkX[4] for graph construction. We set the number of retrieved few-shot examples to $K_{\text{fs}} = 3$ and the maximum iteration count to $T = 3$. Appendix D provides the corresponding parameter sensitivity analysis.

[2] https://github.com/langchain-ai/langgraph
[3] https://github.com/chroma-core/chroma
[4] https://github.com/networkx/networkx

Table 3: Average results of K-MASTER with different backbone LLM configurations over six datasets.

Model	P@1	P@3	P@5	R@1	R@3	R@5	F1@1	F1@3	F1@5	MRR	Time
8B-Instruct	0.613	0.354	0.247	0.357	0.538	0.592	0.431	0.405	0.332	0.691	**2.25**
8B-Think	0.646	0.385	0.272	0.377	0.586	0.650	0.456	0.443	0.367	0.734	5.96
30B-A3B-Instruct	**0.665**	**0.392**	**0.278**	**0.388**	**0.596**	**0.662**	**0.470**	**0.450**	**0.374**	**0.750**	2.43
30B-A3B-Think	0.635	0.383	0.270	0.375	0.588	0.651	0.452	0.442	0.365	0.731	11.97
32B-Instruct	0.629	0.367	0.261	0.377	0.568	0.632	0.451	0.423	0.352	0.715	6.23
32B-Think	0.634	0.379	0.273	0.377	0.586	0.659	0.454	0.439	0.369	0.728	11.32

Table 4: Comparison with a prior multi-agent baseline on CCNU and GeoQA.

Dataset	Method	P@1	P@3	P@5	R@1	R@3	R@5	F1@1	F1@3	F1@5	MRR	Time
CCNU	MAS	0.095	0.081	0.077	0.079	0.205	0.321	0.087	0.117	0.124	0.183	85.26
	K-MASTER	**0.413**	**0.210**	**0.138**	**0.364**	**0.526**	**0.569**	**0.387**	**0.300**	**0.223**	**0.491**	**2.77**
GeoQA	MAS	0.423	0.339	0.290	0.232	0.536	0.726	0.299	0.416	0.414	0.604	44.82
	K-MASTER	**0.851**	**0.561**	**0.376**	**0.489**	**0.842**	**0.908**	**0.621**	**0.673**	**0.532**	**0.909**	**2.77**

5 Results and Analysis

5.1 Backbone Comparison

Table 3 shows the average performance over all six datasets. The results lead to three main observations. First, larger dense models improve direct instruction mode, but the gain is not proportional to latency: 32B-Instruct outperforms 8B-Instruct, yet it is substantially slower. Second, the 30B-A3B configuration is both more effective and faster than the similarly sized 32B model, suggesting that MoE architecture is well suited to the K-MASTER workflow. Third, chain-of-thought reasoning is not uniformly beneficial. It improves 8B-Think and 32B-Think relative to their Instruct counterparts, but degrades 30B-A3B-Think while substantially increasing latency. Overall, the best trade-off is achieved by **30B-A3B-Instruct**, which obtains the strongest scores on all quality metrics considered while remaining nearly as fast as the smallest model. We therefore use 30B-A3B-Instruct (Qwen3-30B-A3B-Instruct-2507) as the default backbone in subsequent experiments.

5.2 Comparison with Baselines

Table 4 compares K-MASTER with the prior multi-agent baseline MAS. Since MAS scales linearly with the number of labels, we evaluate it only on CCNU and GeoQA, which contain relatively small concept sets of 98 and 49 concepts, respectively. K-MASTER clearly outperforms MAS on both datasets while substantially reducing inference time. MAS performs poorly because it treats multi-label tagging as repeated binary classification and therefore considers each question–concept pair in isolation. K-MASTER, in contrast, proposes and filters concept sets holistically while constraining outputs through retrieval into the standard knowledge taxonomy. The runtime comparison is also revealing: K-MASTER takes 2.77 seconds per sample on both CCNU and GeoQA, whereas MAS increases from 44.82 seconds on GeoQA to 85.26 seconds on CCNU as the taxonomy grows from 49 to 98 concepts.

Table 5 further compares K-MASTER with the supervised LHABS model on the full test set and the low-frequency subset. On the full test sets, LHABS remains stronger than K-MASTER, which is expected because LHABS is trained with supervision on the unified label space. The advantage of K-MASTER becomes clearer in the low-frequency setting defined in Section 4.4. LHABS drops substantially from all samples to low-frequency samples. In contrast, K-MASTER is more stable. It even surpasses LHABS on the Top-3 metrics and MRR in the low-frequency subset. These results suggest that K-MASTER is less dependent on the label-frequency distribution of the training split. It can better adapt to concepts with limited annotated examples, demonstrating its effectiveness in data-scarce scenarios.

5.3 Ablation Study

We examine five variants: removing iterative refinement, removing few-shot examples, replacing dynamic planning with fixed BM25 retrieval, replacing it with fixed embedding retrieval, and removing graph expansion. Table 6 shows the average results.

Few-shot support is the most important component: removing it lowers F1@3 from 0.450 to 0.391

Table 5: Comparison between LHABS and K-MASTER on all samples versus low-frequency samples.

Dataset	Split	Method	P@1	P@3	P@5	R@1	R@3	R@5	F1@1	F1@3	F1@5	MRR
CCNU	All	LHABS	**0.658**	**0.311**	**0.200**	**0.586**	**0.783**	**0.833**	**0.620**	**0.444**	**0.323**	**0.750**
		K-MASTER	0.413	0.210	0.138	0.364	0.526	0.569	0.387	0.300	0.223	0.491
	Low-Freq.	LHABS	0.262	0.226	**0.164**	0.199	0.522	**0.632**	0.225	0.314	**0.260**	0.460
		K-MASTER	**0.372**	**0.227**	0.154	**0.301**	**0.527**	0.590	**0.333**	**0.317**	0.245	**0.484**
GeoQA	All	LHABS	**0.906**	**0.579**	**0.389**	**0.515**	**0.866**	**0.936**	**0.656**	**0.693**	**0.549**	**0.944**
		K-MASTER	0.851	0.561	0.376	0.489	0.842	0.908	0.621	0.673	0.532	0.909
	Low-Freq.	LHABS	**0.844**	0.613	0.455	0.348	0.694	0.836	0.492	0.651	0.589	0.901
		K-MASTER	0.843	**0.627**	**0.469**	**0.362**	**0.717**	**0.856**	**0.507**	**0.669**	**0.606**	**0.902**

Table 6: Average ablation results over six datasets.

Metric	w/o Iter	w/o FS	w/o Plan (BM25)	w/o Plan (Emb)	w/o Graph	K-MASTER
P@1	0.616	0.538	0.650	0.662	0.660	**0.665**
P@3	0.362	0.340	0.379	0.387	0.391	**0.392**
P@5	0.257	0.246	0.265	0.273	0.276	**0.278**
R@1	0.361	0.303	0.379	0.388	**0.388**	**0.388**
R@3	0.548	0.515	0.577	0.589	**0.595**	**0.595**
R@5	0.609	0.587	0.636	0.654	0.661	**0.662**
F1@1	0.436	0.371	0.459	**0.469**	**0.469**	**0.469**
F1@3	0.415	0.391	0.436	0.445	**0.450**	**0.450**
F1@5	0.345	0.332	0.358	0.368	0.372	**0.374**
MRR	0.695	0.650	0.733	0.744	0.748	**0.750**
Avg. Iter.	1.00	1.63	1.58	1.57	1.59	1.58

and MRR from 0.750 to 0.650. This indicates that retrieved examples provide crucial guidance on label style and granularity for both generation and evaluation. Iterative refinement is the second most important factor, confirming that feedback-driven correction substantially improves results. Dynamic retrieval planning also matters: both fixed-strategy variants underperform the full system, and fixed embedding retrieval is consistently stronger than fixed BM25 retrieval.

Figure 3 shows that embedding retrieval is the dominant strategy across all datasets, while BM25 is used more often on mathematics datasets than on non-mathematics datasets. This reflects the fact that many mathematical concepts contain explicit lexical cues, such as theorem names, formula names, and symbolic expressions. Graph 1-hop expansion is used less frequently overall, but its higher usage on Biology and History suggests that the planner applies it selectively when the co-occurrence graph provides useful expansion signals.

Table 7 further shows that the benefit of graph expansion depends on graph density. It improves most metrics on Biology and History, where the co-occurrence graphs are relatively dense, but brings only marginal changes on CCNU, whose co-

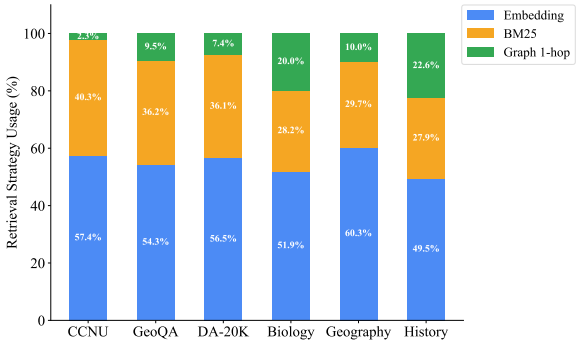

Figure 3: Distribution of retrieval strategies.

occurrence graph is very sparse. This explains why w/o Graph is close to the full model on average: graph expansion serves as a selective coverage-oriented complement rather than a dominant retrieval mechanism.

5.4 Routing Analysis

We also analyze multi-subject routing under a unified taxonomy. We compare standard LLM-based routing with manual routing, where the correct collection is specified in advance. Table 8 shows that LLM routing is close to manual routing on non-mathematics subjects, but the gap is larger within mathematics, indicating that fine-grained curriculum-stage routing is more challenging.

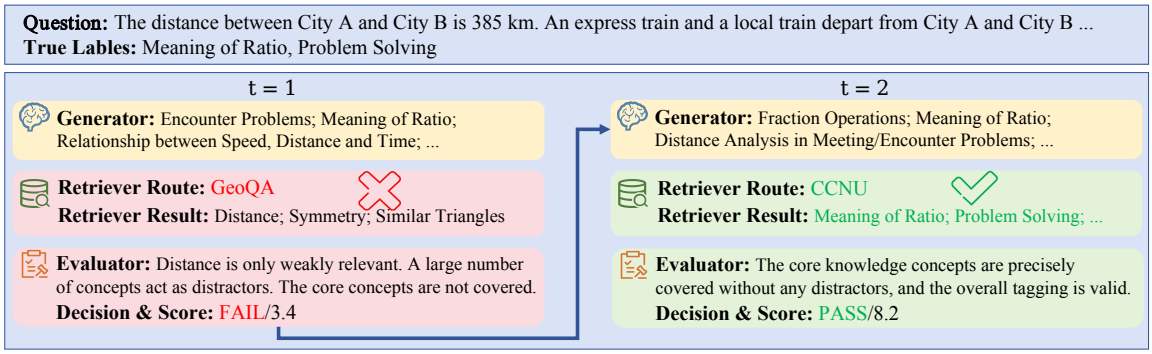

Figure 4: Case study illustrating how evaluator feedback corrects an early routing error.

Table 7: Effect of graph expansion on selected datasets.

Dataset	Model	F1@1	F1@3	F1@5	MRR
Biology	w/o Graph	0.300	0.393	0.376	0.822
	K-MASTER	**0.313**	**0.403**	**0.389**	**0.836**
History	w/o Graph	0.319	**0.414**	0.386	0.744
	K-MASTER	**0.326**	0.408	**0.398**	**0.751**
CCNU	w/o Graph	0.386	0.295	0.220	0.486
	K-MASTER	**0.387**	**0.300**	**0.223**	**0.491**

Table 8: Routing performance by subject type.

Metric	K	Math		Other Subjects	
		LLM	Manual	LLM	Manual
P@K	1	0.600	**0.655**	**0.730**	0.727
	3	0.371	**0.397**	0.413	**0.418**
	5	0.253	**0.273**	0.303	**0.307**
R@K	1	0.401	**0.447**	0.375	**0.382**
	3	0.656	**0.715**	0.534	**0.547**
	5	0.721	**0.793**	0.603	**0.614**
F1@K	1	0.477	**0.527**	0.462	**0.466**
	3	0.470	**0.505**	0.431	**0.438**
	5	0.372	**0.403**	0.376	**0.382**
MRR	–	0.686	**0.748**	0.814	**0.817**

Table 9: Routing distribution over selected concept collections (%).

Dataset	CCNU	GeoQA	DA-20K	Bio.	Geo.	Hist.
CCNU	**76.1**	18.3	1.8	–	3.8	0.1
GeoQA	0.8	**94.5**	4.3	–	0.4	–
DA-20K	2.5	9.4	**87.4**	0.1	0.3	0.3
Biology	0.1	0.1	–	**99.2**	0.7	–
Geography	0.1	0.1	0.1	0.5	**94.9**	4.4
History	0.2	0.2	0.3	0.6	0.5	**98.2**

Table 9 further breaks down the routing distribution. Cross-subject confusion is rare: Biology and History are routed correctly in more than 98% of cases, and Geography reaches 94.9%. The main failure mode is within-math stage confusion. CCNU is routed to the elementary math collection only 76.1% of the time, with 18.3% misrouted to middle-school geometry; DA-20K also has a non-trivial 9.4% confusion rate with middle-school geometry. This suggests that future improvements should focus on finer-grained stage signals, such as curriculum level, question difficulty, or long-term routing memory.

Figure 4 illustrates the role of iterative feedback in such cases. In a representative CCNU example, the first round routes an elementary mathematics question to the middle-school geometry collection GeoQA and retrieves irrelevant geometry concepts. The evaluator rejects these candidates and identifies the missing arithmetic concepts; the next round corrects the route to the elementary mathematics collection and recovers the relevant labels. This case highlights the importance of iterative feedback in correcting early routing errors and improving overall robustness.

6 Conclusion

In this paper, we presented K-MASTER, a training-free multi-agent framework for educational knowledge concept tagging. The framework addresses two core obstacles in practical tagging: constraining LLM outputs to a predefined knowledge taxonomy and maintaining efficiency in large multi-label spaces. Its generate–retrieve–evaluate architecture, routed hybrid retrieval, and iterative feedback loop together provide a robust workflow for cross-subject deployment.

Experiments on six datasets across four subjects show that K-MASTER achieves a strong balance between accuracy, efficiency, and adaptability. It clearly outperforms a prior discriminative multi-agent baseline in both effectiveness and latency, and it is substantially more stable than a supervised model on low-frequency labels.

Limitations

K-MASTER is designed for training-free knowledge concept tagging. As a result, it may not outperform supervised models when large amounts of high-quality labeled data are available and the knowledge taxonomy is stable. Its performance also depends on the reasoning ability of the backbone LLM and on the quality of the retrieved few-shot examples, which provide important guidance on label granularity and annotation style. Another limitation lies in fine-grained routing. The current routing module performs well at coarse-grained subject routing, but it remains vulnerable when different curricula within the same subject share overlapping terminology. This is most evident in the confusion among elementary, middle-school, and high-school mathematics collections. In addition, K-MASTER only uses feedback within the iterative process of a single instance; it does not accumulate experience across samples or continuously improve its routing and tagging behavior over time. Finally, our evaluation mainly focuses on the scores of tagging metrics, without further validating the effectiveness of the model in downstream educational tasks, such as knowledge tracing, question recommendation, or adaptive learning.

Acknowledgments

Acknowledgments are omitted for anonymous review.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. 2021. GeoQA: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics*, pages 513–523.

Ziqi Ding, Xiaolu Wang, Yuzhuo Wu, Guitao Cao, and Liangyu Chen. 2025. Tagging knowledge concepts for math problems based on multi-label text classification. *Expert Systems with Applications*, 267:126232.

Christian Fischer, Zachary A Pardos, Ryan Shaun Baker, Joseph Jay Williams, Padhraic Smyth, Renzhe Yu, Stefan Slater, Rachel Baker, and Mark Warschauer. 2020. Mining big data in education: Affordances and challenges. *Review of Research in Education*, 44(1):130–160.

Tao Huang, Shengze Hu, Keke Lin, Huali Yang, Hao Zhang, Houbing Song, and Zhihan Lv. 2023a. Sequence generation model integrating domain ontology for mathematical question tagging. *ACM Transactions on Asian and Low-Resource Language Information Processing*.

Tao Huang, Shengze Hu, Huali Yang, Jing Geng, Sannyuya Liu, Hao Zhang, and Zongkai Yang. 2023b. Pqsct: Pseudo-siamese bert for concept tagging with both questions and solutions. *IEEE Transactions on Learning Technologies*, 16(5):831–846.

Hang Li, Tianlong Xu, Ethan Chang, and Qingsong Wen. 2025. Knowledge tagging with large language model based multi-agent system. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, 28, pages 28775–28782.

Hang Li, Tianlong Xu, Jiliang Tang, and Qingsong Wen. 2024a. Automate knowledge concept tagging on math questions with llms. *arXiv preprint arXiv:2403.17281*.

Hang Li, Tianlong Xu, Jiliang Tang, and Qingsong Wen. 2024b. Knowledge tagging system on math questions via llms with flexible demonstration retriever. *arXiv preprint arXiv:2406.13885*.

Sourav Mandal and Sudip Kumar Naskar. 2021. Classifying and solving arithmetic math word problems—an intelligent math solver. *IEEE Transactions on Learning Technologies*, 14(1):28–41.

Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. 2018. Question categorization and classification using grammar based approach. *Information Processing and Management*, 54(6):1228–1243.

Steven Moore, Robin Schmucker, Tom Mitchell, and John Stamper. 2024. Automated generation and tagging of knowledge components from multiple-choice questions. In *Proceedings of the 11th ACM Conference on Learning @ Scale*, pages 122–133.

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

Bo Sun, Yunzong Zhu, Yongkang Xiao, Rong Xiao, and Yungang Wei. 2018. Automatic question tagging with deep neural networks. *IEEE Transactions on Learning Technologies*, 12(1):29–43.

Venktesh Viswanathan, Mukesh K. Mohania, and Vikram Goyal. 2024. Tagrec++: Hierarchical label aware attention network for question categorization. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3529–3540.

Chengliang Wang, Xiaojiao Chen, Teng Yu, Yidan Liu, and Yuhui Jing. 2024. Education reform and change driven by digital technology: A bibliometric study from a global perspective. *Humanities and Social Sciences Communications*, 11(1):1–17.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-Thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.

A Prompt Templates

This appendix provides the prompt templates used by K-MASTER. Each block contains the base prompt; when a module uses iterative feedback, the base prompt is followed by an optional refinement segment. In the first iteration, only the base prompt is used. In later iterations, the refinement segment is appended and instantiated with feedback from the shared history.

A.1 Generator Prompt

Base prompt. You are a domain expert for educational concept tagging. Your task is to identify the core knowledge concepts from a question and its reference answer.

Task requirements. 1. Carefully analyze the question and answer, and generate `<kc_num>` most relevant and central knowledge concepts. 2. Each concept should be concise, domain-specific, and highly relevant to the question. 3. Briefly explain the reason for selecting these concepts. 4. You will be given several similar examples. Learn their label style and granularity, but do not copy them mechanically; the current question must remain the basis for prediction.

Input.
Question: `<question>`.
Reference answer: `<answer>`.
Similar examples:
Example index: `<idx>`.
Similarity score: `<score>`.
Example question: `<example_question>`.
Example answer: `<example_answer>`.
Example gold labels: `<example_labels>`.
Prefer standardized, retrievable, and domain-general concept names.

Output format. Return a valid JSON object with the following fields:

```
{
"reason": "...",
"knowledge_concepts": ["...", "..."]
}
```

`reason`: a brief explanation of the concept selection.
`knowledge_concepts`: a list of `<kc_num>` generated concepts.

Refinement segment for later iterations. Revision task for round `<last_turn.turn_id + 1>`: the previous generation has the following problems and must be corrected. 1. Unmappable terms: `<failed_retrieval_concepts>`. These terms do not exist in the knowledge collection; replace them with more standard domain terms or check whether they are hallucinated. 2. Rejected concepts: `<bad_cases>`. These concepts were judged incorrect or distracting; remove or revise them, and do not generate them again. 3. Missing core concepts: `<missing_concepts>`. The evaluator identified these omissions; include the relevant concepts in this round. 4. Expert critique: `<critique>`. Revise the generation strategy according to this feedback. Based on all feedback above, regenerate `<kc_num>` accurate knowledge concepts.

A.2 Retriever Routing Prompt

Base prompt. You are a knowledge-collection router. Assign the question to the most appropriate vector database collection.

Routing criteria. 1. Subject first: identify the broad subject, such as Biology, History, Geography, or Mathematics. 2. Curriculum stage second: for Mathematics, further determine whether the question belongs to elementary mathematics, middle-school geometry, or high-school mathematics according to difficulty and question-type cues. 3. Select exactly one most appropriate collection for retrieval.

Input.
Collection descriptions: `<collection_infos>`.
Question to route: `<question>`.

Output format. Return only a JSON list containing the selected collection name: `["kc_math_pm"]`. Do not output any additional text.

Refinement segment for later iterations. Revision task for round `<last_turn.turn_id + 1>`: the previous retrieval may have selected an inappropriate collection. 1. Unmappable terms: `<failed_retrieval_concepts>`. These may indicate that the previous collection did not contain the required concepts. 2. Rejected concepts: `<bad_cases>`. Judge from their `k_id` values whether the wrong collection was selected. 3. Missing core concepts: `<missing_concepts>`. Infer which collection is most likely to contain these concepts. 4. Expert critique: `<critique>`.

Use this feedback to revise the routing decision. Based on all feedback above, select the most appropriate collection again.

A.3 Retriever Strategy Planning Prompt

Base prompt. You are an advanced retrieval strategy planner. For a generated concept, choose the most appropriate retrieval tools according to the concept features and historical reflection.

Available tools. `BM25`: suitable for proper nouns, formula names, person names, and specific terms that require exact lexical matching. `Embedding`: suitable for descriptive concepts, semantic understanding, and fuzzy queries. `Graph_1hop`: suitable for supplementary expansion when concept coverage is insufficient.

Decision logic. If the query is a proper noun, such as a theorem name, prefer `BM25`. If the query is descriptive, such as an analysis of a physical process, prefer `Embedding`. If the best strategy is uncertain or broader coverage is needed, select multiple tools, such as `["BM25", "Embedding"]`. If the history shows zero hits from `BM25`, try `Embedding`. If both `BM25` and `Embedding` have been used but coverage remains insufficient, consider `Graph_1hop`.

Input.
Question: `<question>`.
Query concept: `<kc>`.

Output format. Return a JSON list containing the selected tool names, for example:
`["Embedding"]`
`["Graph_1hop", "Embedding"]`

Refinement segment for later iterations. Revision task for round `<last_turn.turn_id + 1>`: adjust the retrieval strategy according to previous failures. 1. Unmappable terms: `<failed_retrieval_concepts>`. These may indicate that the previous retrieval strategy caused zero-hit cases. 2. Missing core concepts: `<missing_concepts>`. These may indicate insufficient retrieval coverage. 3. Expert critique: `<critique>`. Use this feedback to improve the tool selection. 4. Previous action for concept `<kc>`: strategy `<last_action.strategy>`, number of hits `<len(related_case)>`, and rejected retrieved results `<related_bad_cases>`. Based on all feedback above, select the retrieval tools for the current round.

A.4 Retriever Candidate Filtering Prompt

Base prompt. You are a knowledge-concept filtering expert. From the candidate concepts, select the `<choose_k>` concepts that are most relevant to the question.

Requirements. 1. Select only from the candidate list; do not add new concepts or modify any `k_id`. 2. Return exactly `<choose_k>` concepts, unless the candidate list contains fewer concepts, in which case return all candidates. 3. Return only a JSON list. Each element must contain `k_id` and `knowledge`. 4. Do not output any additional explanation.

Input.
Question: `<question>`.
Generated concepts: `<generated_concepts>`.
Candidate concepts list:
`<candidate_concepts_list>`.

Output format. Return a JSON list in the following form:
`[{"k_id": "...", "knowledge": "..."}]`

A.5 Evaluator Prompt

Base prompt. You are an expert evaluator for educational concept tagging. Evaluate whether each candidate concept matches the question, select the `<num_candidates>` most appropriate concepts, perform gap analysis, provide an overall critique, and decide whether the system should regenerate the result.

Evaluation steps. 1. Candidate-level judgment: mark each concept as `relevant` if it is correct, relevant, and useful for solving the question; otherwise mark it as `irrelevant`. If a concept is marked as `irrelevant`, provide a `reject_reason`, such as formula error, concept confusion, duplication, or inappropriate granularity. Assign each concept a `quality_score` from 0 to 10, where 10 indicates a highly appropriate concept and 0 indicates a completely irrelevant concept. 2. Gap analysis: check whether any core concepts are missing. If so, list the missing concepts in `missing_concepts`. 3. Overall decision: return `PASS` if the relevant concepts cover the main knowledge requirements; return `FAIL` if serious distractors remain or key concepts are missing.

Input.
Candidate list: `<candidate_list>`.
Question: `<question>`.
Reference answer: `<answer>`.
Similar examples:
Example index: `<idx>`.
Similarity score: `<score>`.
Example question: `<example_question>`.
Example answer: `<example_answer>`.
Example gold labels: `<example_labels>`.

Output format. Return a valid JSON object with the following fields:

```
{
"evaluations": [
{
"k_id": "...",
"knowledge": "...",
"status": "relevant/irrelevant",
"reject_reason": "...",
"quality_score": 0-10
},...],
"missing_concepts": ["..."],
"critique": "...",
"decision": "PASS/FAIL"
}
```

`evaluations`: candidate-level judgments and scores.
`missing_concepts`: uncovered core concepts.
`critique`: a natural-language summary and revision suggestions.
`decision`: either `PASS` or `FAIL`.

Important rules. The `k_id` values in the evaluation results must be identical to those in the candidate list. Do not modify existing `k_id` values or create new ones. Select concepts only from the candidate list, and always return valid JSON.

B Dataset Distributions

This section presents the distribution statistics of the six datasets used in our experiments. Figure 5 shows the question and answer length distribution, the number of concepts per sample, and the concept-frequency distribution.

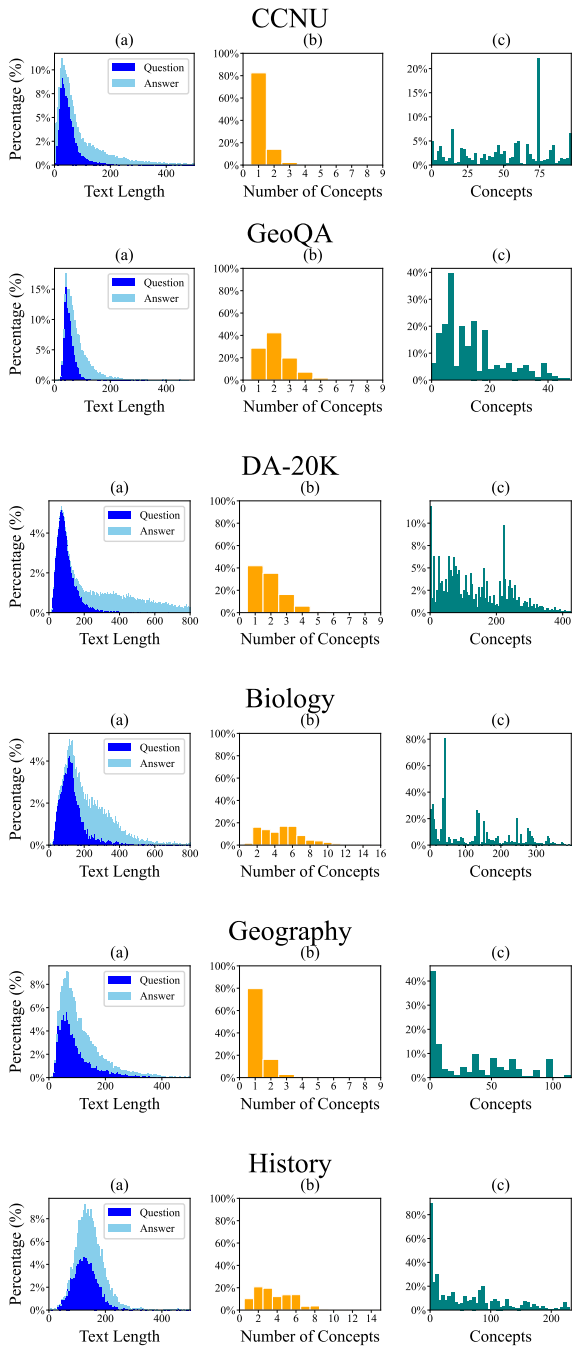

Figure 5: Distribution statistics of the six datasets.

C Low-Frequency Subset Statistics

Table 10 summarizes the low-frequency subsets used in Table 5. Low-frequency concepts are selected from the tail 10% of cumulative label occurrences in the training split.

Table 10: Low-frequency subset statistics for CCNU and GeoQA.

Statistic	CCNU	GeoQA
Concepts	98	49
Full test samples	2,273	1,238
Low-freq. concepts	45	23
Low-freq. test samples	285	216

D Parameter Sensitivity Analysis

This section analyzes K-MASTER's sensitivity to two key hyperparameters: the number of retrieved few-shot examples K_{fs} and the maximum number of refinement iterations T.

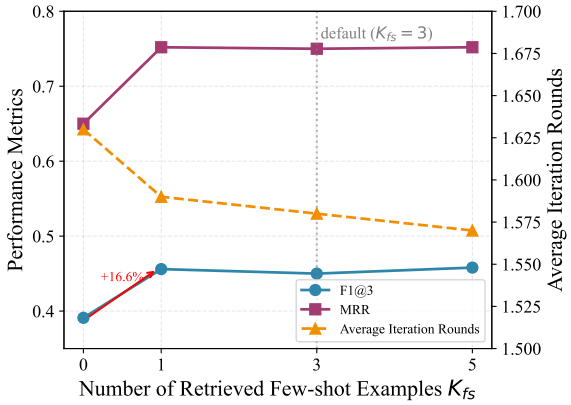

Figure 6: Sensitivity analysis of the number of retrieved few-shot examples K_{fs}.

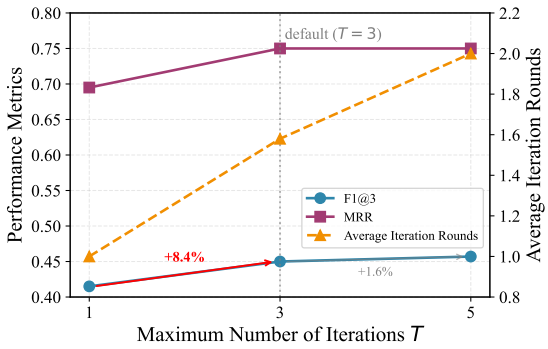

Figure 7: Sensitivity analysis of the maximum number of iterations T.

Figure 6 shows that adding retrieved few-shot examples substantially improves performance over the zero-shot setting, indicating that concrete examples are useful for aligning the model with the desired concept granularity and naming style. The

improvement quickly saturates after a small number of examples: using more examples brings only small changes in F1@3 and MRR, while also increasing prompt length and the risk of introducing less relevant demonstrations. Three examples can cover more annotation styles while remaining in the stable performance region and without substantially increasing the prompt length. We therefore set $K_{\text{fs}} = 3$, which provides enough annotation guidance without making the prompt unnecessarily long.

Figure 7 shows that iterative refinement is also important. Increasing T from 1 to 3 clearly improves F1@3 and MRR, because the system can use evaluator feedback to revise missing or mismatched concepts. However, increasing T from 3 to 5 yields only a small additional gain while the average number of executed iterations continues to rise. This suggests diminishing returns after three rounds. We therefore set $T = 3$ as a practical balance between accuracy and inference cost.