Knowledge Tagging on Math Questions via LLMs with Flexible Sequential Demonstration Retriever

Anonymous ACL submission

Abstract

Knowledge tagging for questions plays a crucial role in intelligent educational applications. Traditionally, these annotations are always conducted by pedagogical experts, as the task requires deep insights into connecting question-solving logic with corresponding knowledge concepts. With the recent emergence of advanced text encoding algorithms, such as pre-trained language models (PLMs), many researchers have developed automatic knowledge tagging systems based on deep semantic embeddings. In this paper, we explore automating the task using Large Language Models (LLMs), in response to the inability of prior encoding-based methods to deal with the hard cases which involve strong domain knowledge and complicated concept definitions. By showing the strong performance of zero- and few-shot results over math questions knowledge tagging tasks, we demonstrate LLMs' great potential in conquering the challenges faced by prior methods. Furthermore, by proposing a reinforcement learning-based demonstration retriever, we successfully exploit the great potential of different-sized LLMs in achieving better performance results while keeping the in-context demonstration usage efficiency high[1].

1 Introduction

Knowledge tagging aims to generate a precise knowledge index to educational materials. It has been recognized as an important factor of current intelligent education systems in providing high-quality educational content to educators and learners during the practice (Chen et al., 2014). For example, with well-annotated education materials, teachers can enjoy great conveniences in organizing coursing content through searching concept keywords index (Sun et al., 2018). Among the tagging objects, concept tagging over math questions has attracted greatly attention because of the recent successes of applying intelligent tutoring systems (ITS) in mathematical education (Burns and Capps, 2013). Traditionally, the questions' concept tags are annotated by the pedagogical experts. However, the rapid growth of the Internet has caused conventional manual methods to be insufficient to meet the demand for handling large volumes of online question data or updating existing concept tags in a timely fashion.

To solve the above issues, existing works (Sun et al., 2018; Zhang et al., 2021) have tried to automate the tagging process with different natural language processing (NLP) algorithms. For example, Du et al. (2021) use text embedding techniques to convert the knowledge definitions and question stems into dense vectors, and then train machine learning classifiers based on the embedding similarities. However, such practice focuses only on comparing the explicit text semantic information but dismisses the implicit relationship in question solutions and knowledge concepts. It can cause unsatisfactory performance when faced with complicated knowledge descriptions and questions. One recent study attempts to improve the tagging performance by leveraging pre-trained language models (PLMs) and fusing external information, such as solution text and conceptual ontology, with original question contents during the judging process (Huang et al., 2023a). Although this new trial demonstrates its effectiveness in solving the challenges faced by prior embedding-based methods, it introduces additional data requirements, e.g., complementary solution text to questions and conceptual ontology information between knowledge concepts, to the knowledge tagging model, which restrict the wide applications of the algorithm to questions with limited external resources.

In this work, we propose a novel knowledge tagging framework KnowTS. It can leverage the advanced mathematical and logical inference capa-

[1]Data and code is available in https://anonymous.4open.science/r/KnowTS-0563

bilities (Achiam et al., 2023) of LLMs to enable knowledge tagging with only knowledge definition text. In addition, owing to the strong in-context learning ability of LLMs, KnowTS has the potential to be swiftly applied with a few annotation samples, setting it apart from all previous training-based algorithms. This feature allows KnowTS to be rapidly adapted for annotating works encompassing nearly all knowledge concepts and questions. Furthermore, due to the huge performance gap among using different sets of demonstration samples (Wang et al., 2023), we propose a novel reinforcement learning (RL) based demonstration retriever focusing on dynamically providing flexible lengths of demonstration samples to every question knowledge matching queries. To validate the effectiveness of KnowTS, we experiment with an expert-annotated knowledge concept question dataset collected from a public K-12 education platform. Experimental results demonstrate that KnowTS can achieve the best in-context learning performance while using fewer demonstrations.

2 Related Work

2.1 Knowledge Concept Tagging

The major challenge of knowledge tagging tasks is how to construct a meaningful link in between the knowledge concepts and the problems, either through the description of the problem themselves or through solutions. The task formulation can primarily be categorized into two directions: retrieval and matrix decomposition. The former relies heavily on training a semantic representation. Sun et al. (2018) employs simple backbone models such as long short-term memory (LSTM) and some attention mechanisms to learn short-range dependency embeddings, where the questions are fed into LSTM layers and are ultimately connected to cross-entropy functions that indicate whether or not a tagging concept belongs to a given problem. Liu et al. (2019a) devised an exercise-enhanced recurrent neural network with Markov property and Attention mechanism to extract rich knowledge concepts information in the exercise's content. Similarly but with enriched data source such as text, multi-modal data (Yin et al., 2019) as well as latex formula combined data (Huang et al., 2021), semantic representations learned with LSTM have been improved to capture more implicit contexts. Huang et al. (2020) fills knowledge graph information into the embedding layers and achieves better mathematical semantic understanding. To take advantage of the robust transformers framework, Zemlyanskiy et al. (2021) pretrained a BERT model to learn jointly predicting words and entities as movie tags given the reviews of movies. Huang et al. (2023b) proposes an improved pretrained bidirectional encoder representation from transformers (BERT) for concept tagging with both questions and solutions.

2.2 In-context Learning Retriever

Few-shot in-context learning (ICL) is the ability of large language models (LLMs) to perform a new task when a few input-output examples or demonstrations for the new task are given alongside the actual task input (Brown et al., 2020). Importantly, the model parameters do not have to be fine-tuned towards the new task. However, the performance of ICL varies significantly based on the choice of demonstrations (Wang et al., 2023; Chen et al., 2024, 2023). In order to keep the stability of ICL performance and exploit the potential of LLMs, many ICL methods have been proposed in recent studies. Rubin et al. (2021) investigated using SBERT (Reimers and Gurevych, 2019) embeddings for demonstration retrieval and show that retrieving demonstrations based on SBERT embeddings often provides a boost in performance compared to zero-shot or random few-shot selection. Liu et al. (2021) found that fine-tuning pretrrained language models on task-related datasets offered further empirical gains. Besides using embedding to recall relevant demonstrations, recent works tried to build a proxy scoring model to score each candidate's demonstration. Then, a retriever is trained, which separates top-score examples from bottom-score examples (Rubin et al., 2021). At last, Scarlatos and Lan (2023) and Lu et al. (2022) directly use response correctness as the reward function and train policy network with RL-method to decide the best demonstration for different samples dynamically.

3 Method

Before diving into the details of the method, we first give a formal problem definition to the knowledge tagging task as follows: Given a pair of knowledge definition text k and a question's stem text q, the objective of a concept tagging model is to produce a binary judgment $y \in \{0, 1\}$, where 1 means k and q are matching, 0 otherwise. In the following subsections, we first present an overview of our proposed framework, KnowTS. Then, we introduce

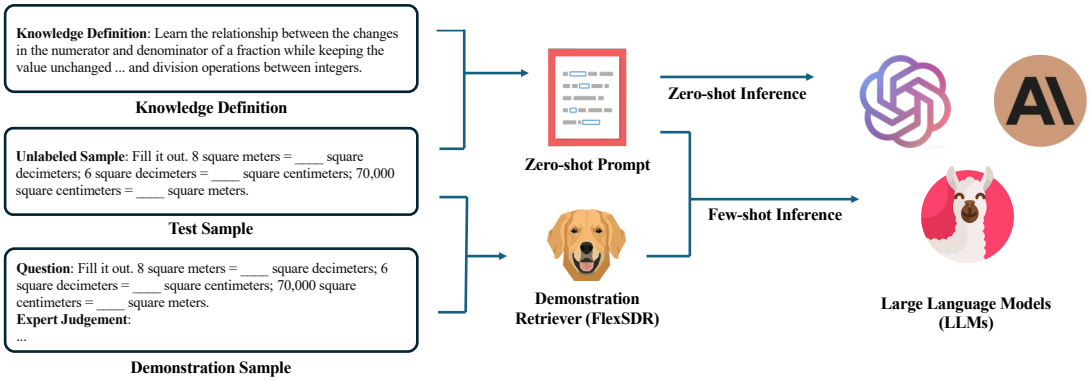


Figure 1: An overview of the workflow of the proposed KnowTS system.

details about the implementations of KnowTS with zero-shot and few-shot inference pipelines. Lastly, to further boost the performance of KnowTS with demonstration samples, we propose FlexSDR, an RL-based retriever algorithm, to achieve efficient and high-performance knowledge tagging.

3.1 An Overview

An overview of KnowTS is demonstrated in Fig. 1. It consists of three key components: (1) zero-shot inference, (2) few-shot inference, and (3) adaptive demonstration retriever. Each component in KnowTS plays different roles to accommodate different knowledge tagging scenarios. When there is no available annotated data, KnowTS will leverage a zero-shot pipeline to generate the judgment directly. And, when there are limited available demonstration examples, KnowTS will use the few-shot inference via its in-context learning capability. When the demonstration samples are needed to select for demonstration, KnowTS will utilize a demonstration retriever to adaptively select effective demonstrations for different (k, q) pairs.

3.2 Zero-shot Pipeline

One key difference between KnowTS and other prior machine learning models is its strong performance while facing limited or even no annotated data for each knowledge k. Such advantages contribute to the powerful zero-shot inference capability of LLMs, which is empowered by its huge size model parameter and the extensive pre-training on diverse and vast datasets. Prior studies by Wei et al. (2021) demonstrate that LLMs have strengths in comprehending instructions in natural language and applying learned knowledge to new problems with limited or even no training data specific to these tasks. In our problem, we leverage this capability by composing a zero-shot prompt as follows: we first describe the goal of the tagging task as *"You are a knowledge concept annotator. Your job is to judge whether the <Question> is concerning the <Knowledge>."* Then, for the convenience of the processing procedure, we add a response format instruction in the prompt: *"The judgment token: <Yes> or <No> should be provided at the end of the response."* At last, as the prior studies like Chain-of-Thought (COT) (Wei et al., 2022) have discovered, instructing LLMs to generate step-by-step problem-solving solutions will be helpful for the LLMs to draw the correct conclusions, we ask LLMs to not only provide their positive or negative predictions but also present the reason at first: *"You should first provide the reasons before giving your judgement."* The detailed zero-shot task instruction prompt engineering can be found in Appx. E.

3.3 Few-shot Pipeline

Although the zero-shot prompt provides a promising solution without using any annotated samples, the knowledge definition text k may sometimes not be specific enough for complicated judgments. For instance, there is a knowledge concept named *consecutive carry in multiplication*, which occurs when the product of two digits, along with any carry from the previous calculation, results in a number greater than 9, thus requiring another carry to be added to the next column in the calculation. It is hard for many LLMs to catch that key point during the judging process without any hints. To overcome this problem, KnowTS can leverage a few-shot inference pipeline when there are available demonstration samples associated with the given knowledge k. Contributing to LLMs' strong in-context learning (ICL) capability, KnowTS can imitate the judging logic of the given demonstrations and achieve significant performance gain even with limited samples provided. Detailed comparisons between zero-shot and few-shot responses are shown in Section 4.5.

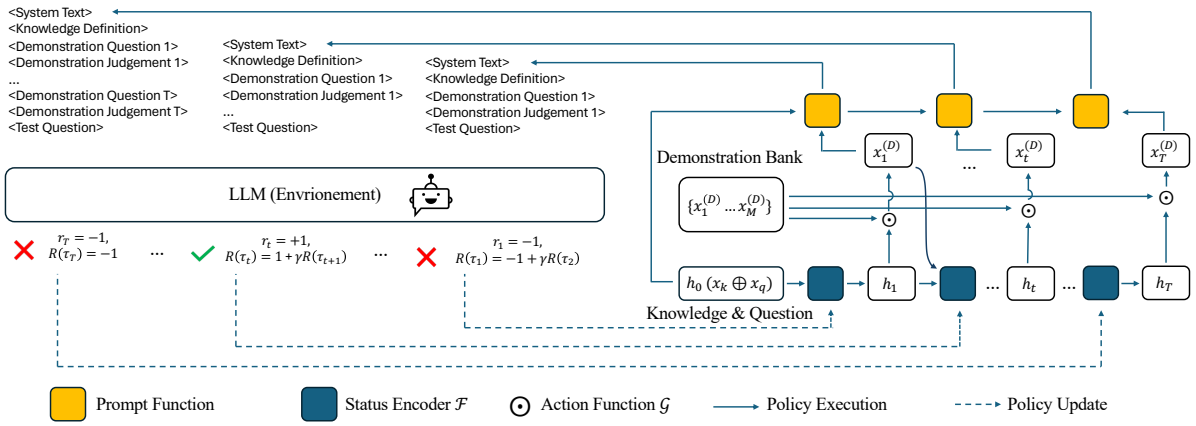


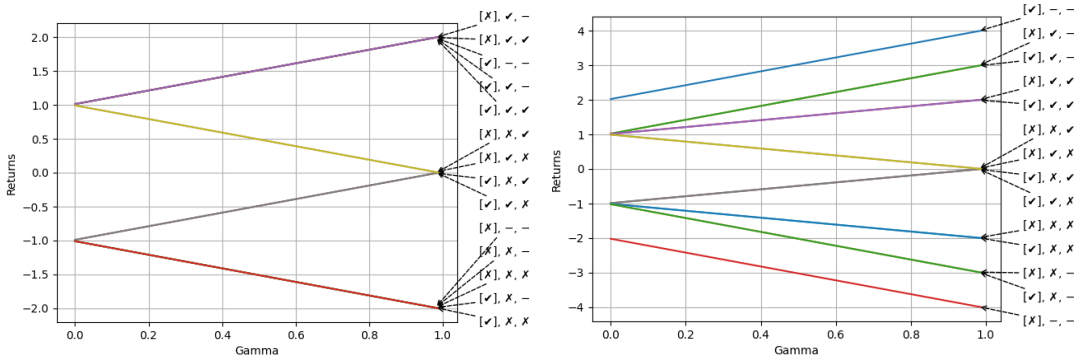
Figure 2: The framework of the proposed FlexSDR.

3.4 FlexSDR

Although incorporating available demonstrations has the potential to bring LLMs performance gain compared to the zero-shot setting, the effectiveness of each demonstration varies (Liu et al., 2021). Moreover, different input pairs (k, q) may prefer to different combinations of demonstrations. To fully exploit the potential of KnowTS, we propose a reinforcement learning (RL) based demonstration selection method, termed Flexible Sequential Demonstration Retriever (FlexSDR), aiming to help LLMs exploit their potential from the demonstration samples while keeping only the necessary demonstrations as input for each input query. Formally, we define the key components of Markov Decision Process (MDP) in our problem as follows: Given the t step's status $s_t = \{k, q, e_1, .., e_{t-1}\}$, where (k, q) are the input knowledge and question pair and $e_{i|1\leq i\leq t-1}$ are the demonstrations selected in prior $t-1$ steps, we hope to use the policy π to generate the subsequent action a_t, which selects one demonstration sample $e_t \in E_{\mathcal{D}}$, where $E_{\mathcal{D}}$ is the demonstration bank or choose the stop signal $e_{\mathcal{E}}$. This process's reward is finding the best demonstration sequence (action trajectory τ) which helps LLMs correctly judge the knowledge matching (k, q) while keeping the $|\tau|$ small. To be noticed although there are several RL-based methods, we point out that FlexSDR has its novelty in two perspectives: (1) we introduce the "early stop" option to each interactive step and use the stop bonus reward to guide the policy network π to learn when to stop during the reinforcing process. With such design, FlexSDR avoids retrieving redundant demonstrations because of the prefixed demonstration size parameter and reduces the risk of sub-optimal few-shot inference performance (Zhao et al., 2023); and (2) We incorporate each intermediate step as an individual training sample and help the policy network learn to conduct the best action decision based on each step's response correctness status. More details about the effectiveness of each design are discussed in Sec 4.6. In the following subsections, we introduce both the policy network and reward design of FlexSDR in details.

3.4.1 Policy Network

Firstly, we define the policy network as π_θ, and the policy execution process is to select an action a_t from the probability distribution calculated by $\pi_\theta(a|s_t)$. As the action space for our policy network is decided by the demonstration bank $E_{\mathcal{D}}$, we decompose π_θ into two components $\pi_\theta = \mathcal{G}(\mathcal{F}(s_t), [E_{\mathcal{D}} \| e_{\mathcal{E}}])$, where $\mathcal{F}$ is a status encoder function that converts the sequential based status variable s_t into a status vector h_t. $\mathcal{G}$ is an action function that calculates each action score for each available demonstration sample. $[\cdot\|\cdot]$ is the concatenation operation, and $e_{\mathcal{E}}$ is the early stop option. Following the prior work's setting (Scarlatos and Lan, 2023), we choose to use the long short-term memory (LSTM) model (Graves and Graves, 2012) as $\mathcal{F}$ and a bilinear transformation as $\mathcal{G}$. Overall, the policy execution process is shown as the right part of Fig. 2, where the input (k, q) pair is first encoded and used as the initial inputs for $\mathcal{F}$. Then, the tth-step hidden state output h_t is used by $\mathcal{G}$ to calculate the selection score for each available demonstration and the early stop option. After that, the action a_t will be selected based on the scores. If the demonstration is selected, it will be appended to the prompt and interact with LLMs to calculate the t-th step reward score. The process ends whenever the early stop option is hit or reaches the max-allowed length. Formally, the



(a) $R(\tau)$ with $\gamma \in (0, 1)$ (b) $R'(\tau)$ with $\omega = 1,\ \gamma \in (0, 1)$

Figure 3: Return functions w/o and w/ stop bonus reward where $T = 2$.

procedure can be defined as follows:

$$
\begin{aligned}
&x_e = \mathcal{E}(e),\ \ x_k = \mathcal{E}(k),\\
&x_q = \mathcal{E}(q), X_{E_\mathcal{D}} = \mathcal{E}(E_\mathcal{D}),\\
&h_t = \mathcal{F}(s_t) = \text{LSTM}(h_0; x_{e_1}, ..., x_{e_{t-1}}),\\
&h_0 = \tanh(W_0[x_k \| x_q] + b_0),\\
&\pi_\theta(a|s_t) = \text{Softmax}(\mathcal{G}(\cdot)),\\
&\mathcal{G}(h_t, [X_{E_\mathcal{D}} \| x_{x_\mathcal{E}}]) = h_t W_a [X_{E_\mathcal{D}} \| x_{x_\mathcal{E}}]^T
\end{aligned}
$$

where x_e, x_k, x_q are the encoding results of knowledge text, question text, and demonstration text. $\mathcal{E}$ is the pre-trained text encoding model. W_0 and b_0 are the parameters of knowledge and question information fusing layer, and W_a is the parameter of bilinear transformation.

3.4.2 Learning Rewards

To train the policy network π_θ, we use the proximal policy optimization (PPO) method (Sutton and Barto, 2018). To be specific, we define the step-wised reward function of FlexSDR as follows:

$$r_t = \text{EVAL}(\hat{y}_t, y),\ \ r_t \in \{-1, 1\} \qquad (1)$$

where EVAL is the evaluation function that compares the judgment response by LLMs $\hat{y} = \text{LLM}(k, q, e_1, ..., e_t)$ with the expertise judgment y. If the two judgments are the same, the reward for timestep t will be $+1$. Otherwise, the reward value will be -1. For early-stop actions $e_\mathcal{E}$, we calculate its correctness based on its most recent step. This reward design differs from previous RL-based retriever training algorithms (Scarlatos and Lan, 2023), which calculate the reward only at the final timestep T. At this point, the size of the retrieved demonstration reaches its maximum allowable limit. For FlexSDR, we calculate rewards r_t for all the timesteps and use the discounted trajectory return $R(s_t, a_t) = r_t + \gamma R(s_{t+1}, a_{t+1})$, where $\gamma \in (0, 1)$ is the discount factor, to calculate the action returns along the trajectory τ, presented in left part of Fig. 2. The final goal of our optimization is to maximize the expectation return of the trajectory τ generated by the iterative execution on the optimized policy network π_θ:

$$J_\theta = E_{\tau \sim p_{\pi_\theta}(\tau)}[\Sigma_t R(s_t, a_t)] \qquad (2)$$

As the instant reward r_t can only be two values -1 or +1, and the maximum length of allowed demonstration for RL-based retriever training is limited, we enumerate all the possible cases for different types of correctness status across the fixed length trajectory ($T = 2$) in Fig 3a. By viewing the corresponding rewards for different trajectories, we can have the following observation: (1) due to the existing discount factor γ, the trajectory with earlier steps approaching the correct response will tend to have a higher reward: $([\times], \checkmark, \checkmark) > ([\times], \times, \checkmark)$, which encourages the policy network to find the most valid demonstration at each iteration; (2) when the policy makes an error attempt at the future steps, its return $R(\tau)$ will be decreased, e.g., $([\times], \checkmark, \checkmark) > ([\times], \checkmark, \times)$, this instructs the policy network to avoid appending the inappropriate demonstrations. In addition to the correct reward, we introduce another "stop bonus" reward r'_t to each time step:

$$r'_t = \begin{cases} 0, & a_t \neq e_\mathcal{E} \\ r_{t-1}, & a_t = e_\mathcal{E} \end{cases} \qquad (3)$$

The bonus added trajectory return can be written as $R'(s_t, a_t) = (r_t + \omega * r'_t) + \gamma R'(s_{t+1}, a_{t+1})$, where ω is a weight parameter balancing the influence of the stop bonus to the final returns. The stop bonus trajectory reward function is plotted in Fig 3b. From the plot, we can observe that $R'(\tau)$ with the earlier correct stop action

(T, S) will receive a higher return compared to the ones with keeping selecting the demonstrations: $([\checkmark], -, -) > ([\checkmark], \checkmark, -) > ([\checkmark], \checkmark, \checkmark)$. This reward design will encourage the policy network to stop early when the correct response is met. For the case when the stop action is given after an error attempt, the return will be penalized: $([\times], -, -) < ([\times], \times, -) < ([\times], \times, \times)$, since it stops exploring the other possibility for finding the correct response.

4 Experiment

In this section, we conduct experiments to validate the effectiveness of each component in KnowTS. Through the experiments, we aim to answer the following research questions:

- RQ1: Can KnowTS outperform prior methods while facing limited or even no annotated data?
- RQ2: Can FlexSDR further boost the few-shot pipeline while using fewer demonstrations compared to the other RL-based Retrievers?
- RQ3: Are components of FlexSDR effective?

4.1 Dataset Overview

To answer the research questions above, we collect a knowledge concept tagging dataset, MathKnowCT, from an online K-12 math education platform. The dataset consists of 24 knowledge concepts, spreading from the math concept learning goal of Grade 1 to Grade 6 students. For each knowledge concept, we collect 100 candidate questions from an unlabeled question database with the highest text embedding similarity and then ask at least two pedagogical experts to conduct the matching annotations. The ratio between matching and mismatching categories of the whole dataset is around 1:4. More details about the dataset statistics and knowledge concept definitions can be found from Tab. 5 in Appendix A. Before the experiment, we first split 5 positive samples and 5 negative samples for each knowledge concept as the training (demonstration) set. For each sample in the training (demonstration) set, we ask a pedagogical expert to complete the reasons for the judgment.

4.2 Baselines

To answer RQ1, we prepare baseline models as follows:

- **Embedding Similarity**: we first use two high-performed long text encoding models, sentence-BERT(S-BERT) (Reimers and Gurevych, 2019) and text-embedding-3-small [2], to encode both k and q into dense embedding vectors x_k and x_q and we calculate the cosine similarity between them. The judgment of each test sample is determined by the top-K selection or similarity threshold η comparisons. The value of hyper-parameter K and η is determined by performing a grid search on test data.
- **PLM Fine-tuning**: We add a binary classification layer to the top of <BOS> tokens outputs and fine-tune the parameter of the whole model with the binary entropy loss calculated on the samples in the training set. The PLMs we use in our experiment include BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), and RoBERTa (Liu et al., 2019b) and the learning rate during the fine-tuning process is tuned from 1e-3 to 1e-5.

To answer RQ2, we compare it with two prior SOTA RL-retrievers.

- **PromptPG** (Lu et al., 2022): We implement PromptPG from the public available source code [3]. During the retrieval process, we input the embedding of both input and demonstration into the retriever and optimize the policy network with the REINFORCE policy gradient algorithm (Williams, 1992) suggested by the paper.
- **RetICL** (Scarlatos and Lan, 2023): RetICL can be viewed as a special case of FlexSDR, where no early stop is added to the demonstration selection space, decay factor γ is set to 1 during the training, and the reward loss will only be calculated for the last step T's result, we implement the algorithm based on changing the setting parameter of FlexSDR.

4.3 LLMs and FlexSDR Settings

To validate the generosity of our proposed algorithm, we experiment with 3 representative LLMs frameworks including GPT (Brown et al., 2020), LLAMA3 (Touvron et al., 2023), Mistral and Mixtral (Jiang et al., 2024). Details about LLM's implementation can be found in Appendix B. During the implementation of FlexSDR, we employed training tricks, such as actor-critic optimization (Konda and Tsitsiklis, 1999), off-policy learning to improve convergence of the training process and data usage efficiency during the training. More details about FlexSDR settings are presented in Appendix C.

[2] https://platform.openai.com/docs/guides/embeddings/embedding-models

[3] https://github.com/lupantech/PromptPG

Table 1: Comparison between PLM Embedding Similarity, PLM Fine-tune, LLM 0-shot Inference. The best result under the comparable settings is marked with underline, and the best result among all settings is marked with **bold**.

Metric	Model Size	Human Expert	K / Q Similarity		Q / Q Similarity		PLM Fine-tune			LLM Zero-Shot		
			GPT-Embed	SBERT	GPT-Embed	SBERT	BERT	RoBERTa	T5	GPT	Llama-3	Mixtral
Accuracy	Base	91.75	67.43	79.90	78.52	63.58	58.45	35.51	77.18	75.30	58.44	68.60
	Large	-	-	-	-	-	76.64	79.08	79.55	**89.00**	68.14	74.73
Precision	Base	88.86	52.68	67.66	67.51	49.10	44.03	35.51	64.70	60.09	46.18	53.87
	Large	-	-	-	-	-	63.02	72.61	71.45	**78.38**	52.58	59.89
Recall	Base	88.16	75.27	82.39	75.40	87.63	62.77	**100.0**	78.63	89.25	92.37	89.74
	Large	-	-	-	-	-	82.80	65.94	70.62	95.03	98.79	85.89
F1	Base	88.51	61.98	74.30	71.24	62.93	51.75	52.41	70.99	71.82	61.58	67.32
	Large	-	-	-	-	-	71.57	69.12	71.03	**85.91**	68.63	70.57

Table 2: Comparisons between LLM 2-Shot and 4-Shot Inference. The best result under the comparable settings is marked with underline, and the best result among all settings is marked with **bold**.

Metric	Model Size	2-Shot						4-Shot					
		Random			Heuristic			Random			Heuristic		
		GPT	Llama-3	Mixtral	GPT	Llama-3	Mixtral	GPT	Llama-3	Mixtral	GPT	Llama-3	Mixtral
Accuracy	Base	76.01	75.64	78.72	72.50	73.15	81.15	77.95	79.25	81.33	79.56	80.74	80.62
	Large	89.45	83.45	80.84	90.10	84.26	80.23	90.40	88.00	84.31	**91.11**	88.57	84.06
Precision	Base	60.33	60.59	65.11	57.22	58.47	68.86	62.65	64.91	68.44	64.98	67.52	67.94
	Large	79.86	69.16	67.67	81.86	70.98	67.64	82.56	76.03	73.18	**83.86**	77.91	73.35
Recall	Base	93.41	89.82	86.31	87.37	84.14	85.64	92.88	90.48	87.98	91.26	88.15	85.98
	Large	93.99	95.83	87.50	92.65	93.68	84.27	92.49	**96.37**	87.63	92.82	94.35	87.63
F1	Base	73.31	72.36	74.23	69.15	68.99	76.34	74.82	75.59	76.99	75.91	76.47	75.90
	Large	86.35	80.34	76.32	86.92	80.76	75.04	87.24	85.00	79.76	**88.11**	85.35	79.86

4.4 Zero-Shot and Naive Few-Shot Results

In this section, we answer RQ1 with comparisons between prior machine learning algorithms, including embedding similarity, PLM fine-tuning, and LLM-based methods, e.g., 0-shot, 2-shot, 4-shot inference. From Tab. 1, we can observe that the prior methods present acceptable performance. However, the performance of most of these methods got trapped at around 71% F1-score. For LLM-based methods, we can find even under the zero-shot setting, some of the large-sized ones, e.g., GPT-4-turbo, present extremely strong task-solving capability and achieving 85.9% F1 results, outperforming the non-LLM methods by a great margin. This observation proves our hypothesis that contributes to the broad prior knowledge (math concepts) learned during the pre-training phase and strong problem-solving skills taught in the instruction tunning stages, LLMs are good tools for knowledge tagging tasks with limited or even no annotation data. The result presented in Tab. 2 demonstrates the advantages of LLMs in-context learning capability. With the introduction of only 2 to 4 demonstration samples, most LLMs can achieve significantly better performance compared to the zero-shot cases, and LLMs with lower performance in zero-shot, e.g., Llama-3-70B, receive the performance boost by 10%. Such observation suggests the great potential of LLM-based algorithms in generating high-performed knowledge tagging results with sufficient demonstration samples.

4.5 Retriever Enhanced Few-shot Results

In this section, we answer RQ2 by presenting the comparisons between FlexSDR and other baselines in Tab. 3. From the table, we observe that FlexSDR constantly bring further boosts to the few-shot learning performance compared to the naive few-shot learning results in Tab. 2. Apart from that, as FlexSDR is designed with the early stop mechanism, the average demonstration length used in few-shot learning FlexSDR is always less than the max-shot size. From the table, we find FlexSDR uses 25% less demonstrations for its few-shot learning inference, which achieves our goal of providing fewer demonstrations but better performance. At last, by observing the positive relationship between the proportional performance gain and length increase between 2 and 4 max shots scenarios, we conclude that FlexSDR learns the correct time to end the retrieval process and adaptive incorporates the best demonstration for a good marginal performance gain.

Table 3: Comparisons between three RL-based retrievers on three LLMs. The best result under the comparable settings is marked with underline, and the best result among all settings is marked with **bold**. The number in (parentheses) for FlexSDR is the mean demonstration size the retriever decides.

Metric	Max-Shot Size	GPT Base (GPT-3.5-turbo)			Llama-3 Base (Llama-3-8B)			Mixtral Base (Mistral-7B)		
		PromptPG	RetICL	FlexSDR	PromptPG	RetICL	FlexSDR	PromptPG	RetICL	FlexSDR
Accuracy	2	77.32	80.36	81.21 (1.38)	78.75	81.02	81.50 (1.55)	79.03	80.83	82.92 (1.78)
	4	80.74	81.97	84.35 (2.10)	82.26	82.83	**84.54** (3.65)	82.16	84.16	84.35 (2.82)
Precision	2	62.79	66.99	69.12 (1.38)	65.00	68.15	69.64 (1.55)	66.13	69.69	70.83 (1.78)
	4	66.42	69.23	**72.92** (2.10)	68.81	69.78	72.46 (3.65)	69.75	74.26	74.05 (2.82)
Recall	2	91.05	89.74	86.58 (1.38)	88.95	88.95	86.32 (1.55)	85.79	82.89	89.47 (1.78)
	4	94.21	90.00	90.00 (2.10)	**92.89**	92.37	92.21 (3.65)	89.21	85.79	87.11 (2.82)
F1	2	74.33	76.72	76.87 (1.38)	75.11	77.17	77.09 (1.55)	74.68	75.72	79.07 (1.78)
	4	77.91	78.26	80.57 (2.10)	79.06	79.50	**81.11** (3.65)	78.29	79.61	80.05 (2.82)

4.6 Ablation Studies

To answer RQ3, we ablate the intermediate reward design from FlexSDR and name the new model as FlexRetICR. From the performance comparison between the three models, shown as in Fig. 5 in Appx. D, we observe that FlexRetICR outperforms RetICR in 4 out of 6 cases. It indicates that introducing early stop rewards not only helps to use less demonstrations but also could be beneficial to the final performance. Finally, by comparing FlexSDR with the other two RL-Retrivers, we find that it achieves the best performance in 5 out of 6 scenarios, which proves the effectiveness of the step-wise reward design. More details about our ablation study implementation can be found in Appx. D.

4.7 Case Studies

In addition to RQs above, we further conduct experiments to study behaviors of FlexSDR and KnowTS. First, we present FlexSDR's behavior while facing to knowledge concepts with different zero-shot accuracy. The negative relationship between zero-shot performance and number of demonstrations retrieved in Fig. 6 in Appx. F indicates that FlexSDR learns to retrieve fewer demonstrations for knowledge points that already perform well with no demonstration samples. Second, we categorize the error cases of KnowTS into four major types in Tab. 9 in Appx. G. By analyzing each error type, we conclude that errors, such as restriction dismiss, task distraction, are related with the instruction following issue of LLMs, which indicates the knowledge tagging specific instruction-tuning could be a potential way to further improve the performance of KnowTS. Besides, concept misinterpretation is caused by hallucination shortages of LLMs. To mitigate that, injecting authentic knowledge definition with retrieval augmented generation (RAG) (Gao et al., 2023) will be a good solution. At last, the wrong fact errors reflect LLMs' weakness in deal with quantity constraints, and the recent emerging tool-use LLMs (Zhuang et al., 2024) will be a promising improvement direction.

4.8 General Application of FlexSDR

To explore the general effectiveness of FlexSDR in other few-shot learning scenarios, we applied it with dataset GSM8K (Cobbe et al., 2021). Our experiment compares the effectiveness of different retrieving algorithms in helping LLMs generate correct solutions to questions, and we test all the models using the standard test split of GSM8K. The metric in this experiment is solution accuracy. From Tab. 8 in Appx. H, we can find FlexSDR always outperforms the other retrievers while using fewer demonstrations. Based on this fact, we demonstrate the effectiveness of FlexSDR in generous few-shot learning scenarios.

5 Conclusion

In this paper, we present, KnowTS, a LLMs based knowledge-tagging system, which differs from prior machine learning models in its strong performance while facing limited or even annotated data for knowledge-tagging tasks. Besides that, we further propose a novel RL-based demonstration retriever, FlexSDR, focusing on dynamically providing flexible lengths of demonstration samples to every question knowledge-matching query. To validate the effectiveness of KnowTS, we experiment with an expertly annotated knowledge concept question dataset, MathKnowCT. The experiment results demonstrate the effectiveness of FlexSDR, which enables KnowTS to achieve the best few-shot learning performance while using fewer demonstrations. At last, through the ablation study and case analyzing results, we demonstrate the effectiveness of each component in FlexSDR.

Limitation

In this work, we explore the usage of LLMs for knowledge tagging task. Although we have successfully demonstrated that KnowTS is a powerful system for knowledge tagging task while comparing to the previous SOTA algorithms, especially facing to complicated tagging scenarios, we have to admitted that the computation resource consumption of KnowTS is relatively much larger than the traditional methods. During practice, we choose to integrate traditional method with the KnowTS and always use KnowTS for processing the challenging matching cases. Apart from that, in our error analysis section, we observe KnowTS sometimes make simple errors during the judgement while dealing with complicated domain concepts or strict quantity constraint in math. To fully exploit the potential of LLMs on knowledge tagging, more complicated techniques like as RAG and Tool-use LLMs are also deserved to be explored in the future studies.

References

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

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

Hugh L Burns and Charles G Capps. 2013. Intelligent tutoring systems: an introduction. *Foundations of intelligent tutoring systems*, 1.

Jiuhai Chen, Lichang Chen, Chen Zhu, and Tianyi Zhou. 2023. How many demonstrations do you need for in-context learning? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11149–11159.

Jun-Ming Chen, Meng-Chang Chen, and Yeali S Sun. 2014. A tag based learning approach to knowledge acquisition for constructing prior knowledge and enhancing student reading comprehension. *Computers & Education*, 70:256–268.

Pei Chen, Shuai Zhang, and Boran Han. 2024. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1720–1738.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Wei Du, Haiyan Zhu, and Teeraporn Saeheaw. 2021. Application of the lda model to semantic annotation of web-based english educational resources. *Journal of web engineering*, 20(4):1113–1136.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45.

T. Huang, M. Liang, H. Yang, Z. Li, T. Yu, and S. Hu. 2021. Context-aware knowledge tracing integrated with the exercise representation and association in mathematics. In *Proceedings of the International Educational Data Mining Society*, volume 1, pages 360–366.

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

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.

Zhenya Huang, Qi Liu, Weibo Gao, Jinze Wu, Yu Yin, Hao Wang, and Enhong Chen. 2020. Neural mathematical solver with enhanced formula structure. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1729–1732, New York, NY, USA. Association for Computing Machinery.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. 2019a. Ekt: Exercise-aware knowledge tracing for student performance prediction. *Preprint*, arXiv:1906.05658.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Alexander Scarlatos and Andrew Lan. 2023. Reticl: Sequential retrieval of in-context examples with reinforcement learning. *arXiv preprint arXiv:2305.14502*.

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.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Yu Yin, Qi Liu, Zhenya Huang, Enhong Chen, Wei Tong, Shijin Wang, and Yu Su. 2019. Quesnet: A unified representation for heterogeneous test questions. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19. ACM.

Yury Zemlyanskiy, Sudeep Gandhe, Ruining He, Bhargav Kanagal, Anirudh Ravula, Juraj Gottweis, Fei Sha, and Ilya Eckstein. 2021. Docent: Learning self-supervised entity representations from large document collections. *Preprint*, arXiv:2102.13247.

Xiao Zhang, Meng Liu, Jianhua Yin, Zhaochun Ren, and Liqiang Nie. 2021. Question tagging via graph-guided ranking. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–23.

Fei Zhao, Taotian Pang, Zhen Wu, Zheng Ma, Shujian Huang, and Xinyu Dai. 2023. Dynamic demonstrations controller for in-context learning.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2024. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36.

A MathKnowCT Dataset

The detailed statistics about MathKnowCT is shown in Tabel 5. Overall, there are 2,349 samples covering 24 knowledge concepts of math study for student from Grade 1 to Grade 6. Some exampling knowledge definitions and questions are presented in Table 4. During the data collecting phase, we first ask two pedagogical experts to annotate each knowledge-question sample pair, and if a conflict happens, we ask the third expert to give the final judgment. The Cohen's kappa value of the first round of annotation is 0.9321, and about 3.1% of samples are submitted to the third judger for annotation. In Tab. 4, we present the example knowledge concepts. Please find more details from the link.

B LLMs Implementation

To be noticed in this paper, we choose to experiment only with each LLM's instruct-tuned (chat-tuned) version as we observe that the instruct-tuned LLMs can better follow the given annotating instructions and generate the correct format

Table 4: Example knowledge definitions

Knowledge ID	Knowledge Definition
x01010201	Learn the definitions of following types of numbers, including integers, odd numbers, even numbers, fractions, decimals, positive numbers, negative numbers, and natural numbers. Common related question types include the following: (1) Select a number of a specified type from a given set of numbers; (2) Determine whether a number is within the defined range; (3) Determine whether a proposition about the classification of numbers is true.
x02040502	Learn the composition of two-digit numbers less than or equal to 100 (how many tens and how many ones). Common related question types include the following: (1) Convert a two-digit number into a combination of tens and ones; (2) Fill in the corresponding two-digit number based on the combination of tens and ones.
x02061003	Learn to use 3 or 4 digits to form a three-digit or four-digit number, and judge the size relationship between the digits. Related question types are limited to the following: (1) Use 3 digits to form a three-digit number smaller than a certain number. Find the total number of such three-digit numbers, the largest number, and the smallest number. Each digit can only be used once in the combination process. (2) Knowing that the sum of the digits in each digit of a four-digit number is a certain number, find the largest number and the smallest number of this four-digit number.
x04030501	Learn to calculate the reciprocal of a number. Common related question types include the following: (1) Calculate the reciprocal of one or more given numbers; (2) Given an equation where the product of a number and a blank is 1, find the value of the number that can be filled in the blank.
x48040202	Learn how to estimate the total purchase price of three items in a shopping scenario. Common related question types include the following: (1) Given the prices of three items (each item can be a three-digit or two-digit price), but at least one of the items has a three-digit price, calculate the approximate total purchase price of the three items; (2) Calculate both the approximate and exact total purchase price of the three items;

Table 5: Detailed sample statistics for different knowledge concepts in MathKnowCT.

Knowledge ID	Total Size	Positive	Negative
x02030701	100	25	75
x02021101	100	40	60
x06020104	100	40	60
x02061003	100	16	84
x48040202	100	29	71
x11041602	100	24	76
x04030501	100	48	52
x04030601	100	23	77
x07010103	100	50	50
x06030101	100	44	56
x57130902	100	35	65
x20041003	62	50	12
x07020402	87	29	58
x07020502	100	50	50
x20050401	100	50	50
x09020509	100	50	50
x07020314	100	30	70
x01010201	100	50	50
x11040205	100	26	74
x11040203	100	22	78
x11040202	100	25	75
x02040502	100	44	56
x47060201	100	17	83
x20070401	100	47	53

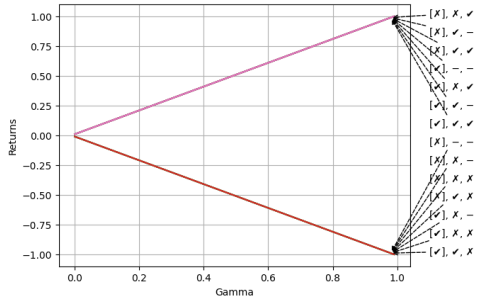
Table 6: Details about LLM implementation in this paper and source file links.

LLM Name	Model ID
GPT-Large	gpt-4-turbo-2024-04-09
GPT-Base	gpt-3.5-turbo-0125
Llama3-Large	Llama-3-70B-Instruct
Llama3-Base	Llama-3-8B-Instruct
Mixtral-Large	Mixtral-8x7B-Instruct-v0.1
Mixtral-Base	Mistral-7B-Instruct-v0.2
Qwen1.5-Large	Qwen1.5-72B-Chat

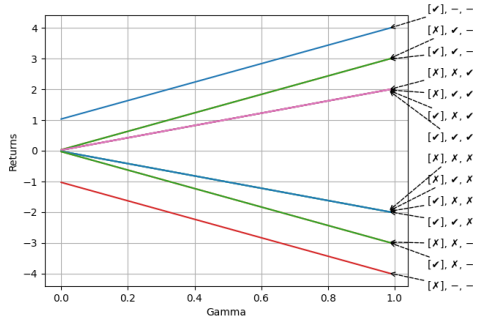
responses. For each framework, we experiment with two-sized versions (small and large) and the prompt text is adjusted based on the preference of each LLM. We run our experiment with 8 * Nvidia A100 80G GPUs.

C FlexSDR Implementation

To implement FlexSDR, we choose to use a 2-layer LSTM with 64 hidden neurons for each layer, and the text encoder $\mathcal{E}$ is text-embedding-3-small, the early-stop bonus weight $\omega = 1$. The discount factor $\gamma = 0.99$. To improve the convergence of the whole training process, we employ the actor-critic optimization framework (Konda and Tsitsiklis, 1999) during training. We train the value function estimator using mean squared error (MSE) based on each step's hidden state $\mathcal{V}(h_t)$, the weight for the loss of value function is set as 0.5. Besides, to further improve the data usage efficiency, we also incorporated off-policy learning epochs during the training, and the off-policy epochs we set in our experiment is 80. Finally, to encourage exploration during the reinforcement steps, we add the negative entropy of the policy to each time step's loss, and the weight is set as 0.01. During the inference time, we use the greedy decoding method at each timestep t, and once the early stop option is hit, the demonstration retrieval procedure stops.

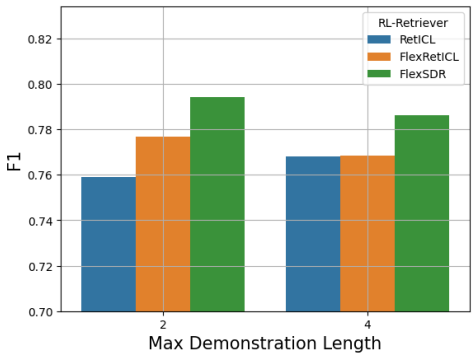


(a) $R(\tau)$ with $\gamma \in (0, 1)$

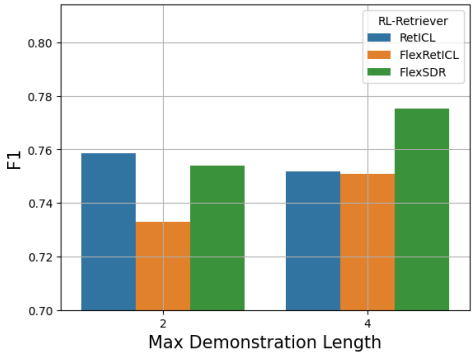


(b) $R''(\tau)$ with $\omega = \frac{1}{2}$, $\gamma \in (0, 1)$

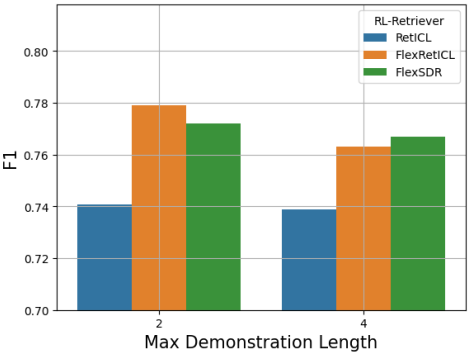
Figure 4: Return functions w/o and w/ stop bonus reward for FlexRetICR where $T = 2$.



(a) GPT-3.5-turbo



(b) Llama3-8B



(c) Mistral-7B

Figure 5: Performance of RetICL, FlexRetICL and FlexSDR with different LLMs.

D Ablation Study

We ablate the intermediate reward design from FlexSDR and name the new model as FlexRetICR since it is similar to RetICL but can perform the early stop action. We train FlexRetICR with both rewards r_t and r'_t and set weight parameter $\omega = \frac{1}{T}$ since we do not want the accumulated stop bonus reward to become larger than the correctness reward. The return function $R''(\tau)$ for this model is shown as Fig. 4. For the fair comparison between FlexRetICR and RetICR, we set $\gamma = 1$ for FlexRetICR.

The performance comparison between the three models is shown as Figure 5. From the figure, we

can observe that FlexRetICR outperforms RetICR in 4 out of 6 cases, which indicates that introducing early stop rewards not only helps to use less demonstrations but also could be beneficial to the final performance. Finally, by comparing FlexSDR with the other two RL-Retrivers, we find that it achieves the best performance in 5 out of 6 scenarios, which proves the effectiveness of the step-wise reward design.

E Instruction Prompt Engineering

During the initial explorations, we tried tuning the prompt for the task and decided to use the prompt presented in the paper based on their empirical performance. The prompts below are the three ones that we explored:

- **Type I (Naive Judgment)**: You are a knowledge concept annotator. Your job is to judge whether the <Question> is concerning the <Knowledge>. The judgment token: '<Yes>' or '<No>' should be provided at the start of the response.

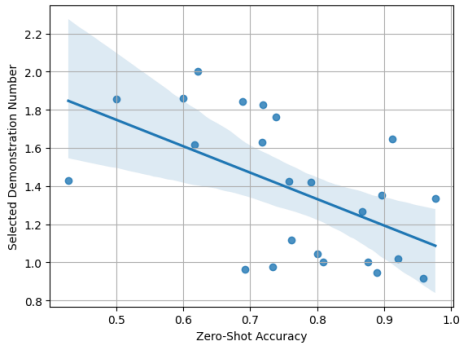
- **Type II (Judgment + Reason)**: You are a knowledge concept annotator. Your job is to judge whether the <Question> is concerning the <Knowledge>. The judgment token: '<Yes>' or '<No>' should be provided at the start of the response. You should also provide the judging reasons for your judgment.

- **Type III (Reason + Judgment)**: You are a knowledge concept annotator. Your job is to judge whether the <Question> is concerning the <Knowledge>. You should first provide the judging reasons before giving your judgment. The judgment token: '<Yes>' or '<No>' should be provided at the end of the response.

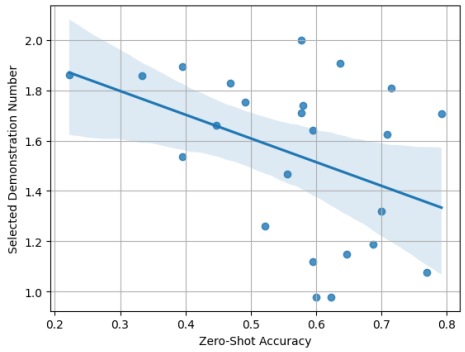
In short, Type I prompt asks LLMs to provide judgment without providing a reason, and Type II prompt requests LLMs to provide both judgment and explanation, but the explanation is given after the judgment. Type III prompt instructs LLMs to give their judging reason before arriving at the final conclusion. The performance of the three types is shown in Tab. 7, and experiments are conducted using zero-shot settings. Based on the results below, we chose Type III prompts as our default prompt for the following experiments in our paper.

Table 7: Comparison between different instruction prompts. The best performance of each metric with different models is marked with **bold**, and the second best one is marked with underline.

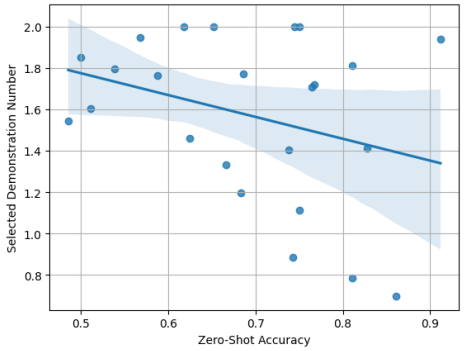
Model	Prompt	Accuracy	Precision	Recall	F1
GPT-Base	Type I	70.21	55.24	**91.58**	68.91
	Type II	63.28	49.50	91.05	64.13
	Type III	**76.28**	**61.73**	90.00	**73.23**
Mixtral-Base	Type I	64.61	53.04	16.05	24.65
	Type II	63.47	49.45	58.95	53.78
	Type III	**68.60**	**53.86**	**90.00**	**67.39**
Llama-3-Base	Type I	63.95	50.00	0.26	0.52
	Type II	58.63	46.48	**97.37**	62.93
	Type III	**67.17**	**52.71**	87.11	**65.67**



(a) GPT-3.5-turbo



(b) Llama3-8B



(c) Mistral-7B

Figure 6: Zero-shot accuracy of different knowledge concepts with corresponding demonstration numbers on different LLMs. Each point in the figure represents a knowledge concept.

F Case Study

From Figure 6, we observe that there is a significant negative relationship between the knowledge-level accuracy at zero-shot performance and the number of demonstrations suggested by FlexSDR. This fact indicates that FlexSDR learns to retrieve fewer demonstrations for knowledge points that already perform well with no demonstration samples. Such a phenomenon provides evidence that FlexSDR learns how to provide an adaptive number of demonstrations to different knowledge concepts.

G Error Analysis

To further enhance the depth of our studies, we sample error samples and categorize the common errors into four categories: restriction dismiss, concept misinterpretation, wrong fact, and task distraction. Detailed explanations and representative error examples for each category are shown in Tab. 9.

H Perform FlexSDR on GSM8K

To demonstrate the generous capability of FlexSDR in dealing with the demonstration selection task, we conduct experiments over the GSM8K dataset. Our experiment on GSM8K compares the effectiveness of different retrieving algorithms in helping LLMs generate correct solutions to questions GSM8K, and we test all the models using the standard test split of GSM8K. The metric in this experiment is solution accuracy. From Tab. 8, we can see that FlexSDR consistently outperforms the other retrievers by using fewer demonstrations. Based on this fact, we demonstrate the general effectiveness of FlexSDR in generous few-shot learning scenarios.

Table 8: Comparison between different retriever over GSM8K dataset. The best performance of each metric with different models is marked with **bold**, and the second best one is marked with underline.

Model	Retriver	Demonstration Size	Accuracy
GPT-Base	Random	4	72.86
	Heuristic	4	73.39
	PromptPG	4	73.62
	RetICL	4	74.60
	FlexSDR	1.85	**76.95**
GPT-Large	Random	4	92.04
	Heuristic	4	92.19
	PromptPG	4	92.72
	RetICL	4	92.80
	FlexSDR	1.84	**93.10**

Table 9: Error category examples, blue text marks the key parts of knowledge definition and red text indicates the error reasons in responses returned by LLMs.

Error Type	Knowledge Definition	Question Stem	LLM Response
Restriction Dismiss	Learn the composition of two-digit numbers less than or equal to 20 (how many tens and how many ones). Common related . . .	Fill in the corresponding two-digit number based on 4 tens and 7 ones.	...specifically "4 tens and 7 ones." This directly aligns with type 2 of the common related question types as outlined in the knowledge definition: "Fill in the corresponding two-digit number based on the combination of tens and ones." Therefore, this question is relevant to the provided knowledge.
Concept Misintepretation	Learn to compare the size relationship between one-digit decimals. Related question types include the following:. . .	Arrange 2.4, 0.9, 1.7, 1.5, and 0.4 from small to large: ____.	...However, the numbers presented in the <Question>include decimals greater than one (specifically, 2.4), which do not fall under the category of "one-digit decimals" as defined in the <Knowledge>. Therefore, the content of the <Question>does not align with the criteria specified in the <Knowledge>.
Wrong Facts	...Common related question types include the following: (1) Calculate addition or subtraction of three tens; (2) After calculating an addition or subtraction of three single-digit numbers, ...	First, find the sum of 23, 35, and 42. Next, find the sum of the tens digits of the three numbers and the initial sum. What is the final sum?	The <Question>involves calculating the sum of three numbers (23, 35, and 42) ...This question aligns with part (2) of the <Knowledge>, where it involves calculating the sum of three single-digit numbers (here, the tens digits of 23, 35, and 42) and . . .
Task Distraction	Learn the relationship between the changes in the numerator and denominator of a fraction while keeping the value unchanged. Common related question types include the following: ...	If the numerator of $\frac{3}{11}$ increases by 6, the denominator should be increased by 6 to keep the size of the fraction unchanged. ____. (True or false)	To determine if the statement is true or false, we need to evaluate the situation described in the question about the fraction $\frac{3}{11}$...Thus, the claim that "the denominator should be increased by 6 to keep the size of the fraction unchanged" is false. <No>