Generalizable Chain-of-Thought Prompting in Mixed-task Scenarios with Large Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) have unveiled remarkable reasoning capabilities by exploiting chain-of-thought (CoT) prompting, which generates intermediate reasoning chains to serve as the rationale for deriving the answer. However, current CoT methods either simply employ general prompts such as *Let's think step by step*, or heavily rely on pre-defined task-specific demonstrations to attain preferable performances, thereby engendering an inescapable gap between performance and generalization. To bridge this gap, we propose **GeM-CoT**, a **Ge**neralizable CoT prompting mechanism in **M**ixed-task scenarios where the type of input questions is unknown. GeM-CoT first categorizes the question type and subsequently samples or constructs demonstrations from the corresponding data pool in an automatic pattern. With this technical design, GeM-CoT simultaneously enjoys superior generalization capabilities and remarkable performances on 10 public reasoning tasks and 23 BBH tasks.

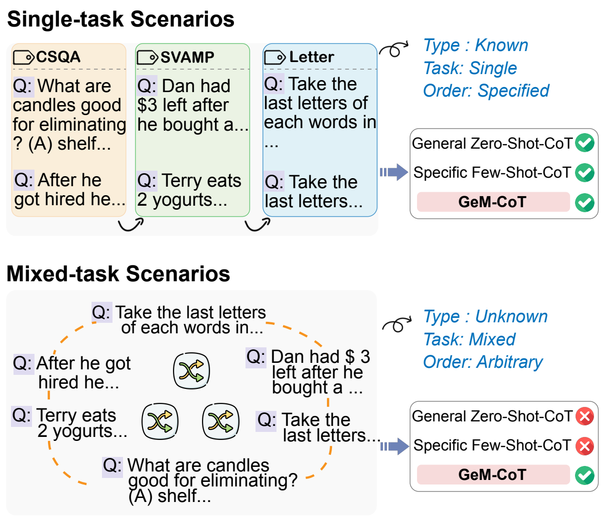

Single-task Scenarios
CSQA
Q: What are candles good for eliminating ? (A) shelf...
Q: After he got hired he...
SVAMP
Q: Dan had $3 left after he bought a...
Q: Terry eats 2 yogurts...
Letter
Q: Take the last letters of each words in ...
Q: Take the last letters...
Type : Known
Task: Single
Order: Specified
General Zero-Shot-CoT
Specific Few-Shot-CoT
GeM-CoT
Mixed-task Scenarios
Q: Take the last letters of each words in...
Q: After he got hired he...
Q: Dan had $ 3 left after he bought a ...
Q: Terry eats 2 yogurts...
Q: Take the last letters...
Q: What are candles good for eliminating? (A) shelf...
Type : Unknown
Task: Mixed
Order: Arbitrary
General Zero-Shot-CoT
Specific Few-Shot-CoT
GeM-CoT

Figure 1: Comparison of conventional *single-task scenarios* and our concerned setting: **mixed-task scenarios**. There are three major characteristics of mixed-task scenarios: (i) the type of any incoming question is unknown; (ii) the input data comes from a set of mixed tasks; (iii) the questions come in an arbitrary order.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Scao et al., 2022; Thoppilan et al., 2022; Chowdhery et al., 2022; Touvron et al., 2023; OpenAI, 2023) have exhibited commendable capabilities on complex reasoning by virtue of chain-of-thought (CoT) prompting (Wei et al., 2023). CoT prompting entails the generation of intermediate reasoning chains that serve as the rationale before deriving the answer.

Current CoT prompting methods predominantly fall into two categories, which we dub as *General Zero-Shot-CoT* and *Specific Few-Shot-CoT*, respectively. The former leverages general trigger prompts such as *Let's think step by step* and appends them directly to the input question, aiming to summon up the step-by-step reasoning potential from LLMs (Kojima et al., 2023; Yang et al., 2023). The latter provides task-specific input-output pairs as in-context demonstrations and puts them before the input question, for the purpose of instructing LLMs to carry out multi-step reasoning with elaborately selected demonstrations (Liu et al., 2022; Wei et al., 2023; Zhang et al., 2023).

Briefly, there are two major limitations in previous studies. On one hand, the *General Zero-Shot-CoT* pattern is endowed with favorable generalization ability as it does not need any task-related demonstrations, but it often pales in terms of performance when compared with the few-shot pattern. On the other hand, the *Specific Few-Shot-CoT* pattern heavily leans on task-specific demonstrations to attain superior performances, yet fails to bear on decent generalization ability. Although recent works have made progress by either mitigating manual labor (Zhang et al., 2023) or promoting the quality of demonstrations (Arora et al., 2023; Diao et al., 2023), all of them rest on the task-associated perspective thus far.

Nevertheless, in practical applications, LLMs tend to confront situations of mixed types of questions (Figure 1), where each question is not clearly pre-identified which task it belongs to. Under these circumstances, it is neither reasonable to improvise several task-related examples by hand nor possible to manually search for which task it refers to, not to mention that the question encountered in actual cases is not even from a pre-defined set of tasks. Besides, naive use of general trigger prompts may result in performance degradation as the lack of templated rationales often leads to spurious reasoning steps (Wan et al., 2023). As a result, there exists an inescapable gap between performance and generalization in our concerned realistic mixed-task scenarios.[1] To alleviate this gap, a potential strategy is to explore the trade-off area between generality and performance while ensuring certain practicality.

This work presents **GeM-CoT**: a **Ge**neralizable CoT prompting mechanism in **M**ixed-task scenarios where the type of input questions is unknown. GeM-CoT first routes the input question to different paths based on whether it can successfully match to a demo pool that is pre-constructed and continuously updated. On one hand, for a successful match, it fetches demonstrations of the matched type from the demo pool and performs a final inference to acquire the answer. On the other hand, when a match fails, it derives the answer through zero-shot reasoning and then stores in the data cache. Afterward, it updates the cache by conducting density-based clustering on the questions within and automatically constructing diverse demonstrations for data in a certain cluster that meets the requirements. The corresponding generated demonstrations are returned to the demo pool for subsequent inference.

We conduct experiments on 10 reasoning tasks covering arithmetic reasoning, commonsense reasoning, and symbolic reasoning. Besides, we further validate the stability and generalization of GeM-CoT on 23 BBH datasets. Experimental results show that GeM-CoT simultaneously enjoys superior generality and remarkable performances.

Our contributions are summarized as follows:

(i) To the best of our knowledge, our work pioneers a novel setting of mixed-task scenarios, which has significant practical application values.

(ii) We propose a generalizable CoT prompting mechanism in mixed-task scenarios, which not only bridges the gap between performance and generalization but also unearths their in-between mutual synergy by gaining performance improvements in sync with achieving generality.

(iii) Experimental results on a total of 33 datasets demonstrate the impressive performance and superior generality of our approach.

2 Related Work

In this section, we discuss two lines of research which are key to our work: CoT prompting and cross-task generalization.

2.1 Chain-of-thought Prompting

Recently, CoT prompting methods have pushed the multi-step reasoning abilities of LLMs to a remarkable aptitude by eliciting them to generate intermediate reasoning chains before deriving the final answer (Wei et al., 2023).

Currently, there are two flavors of research in CoT prompting: *General Zero-Shot-CoT* (Kojima et al., 2023) and *Specific Few-Shot-CoT* (Wei et al., 2023). The former merely appends a *general* prompt to the input question, wheras the latter leverages several task-*specific* input-output pairs as reasoning demonstrations and inserts them before the test question.

General Zero-Shot-CoT. LLMs have proven to be competent zero-shot reasoners by Kojima et al. (2023), which has greatly broadened the generalizability of CoT techniques and liberated the need to prepare task-specific examples in advance. While benefiting from its task-agnostic property, it often fails to excel at performance in comparison with its few-shot rivals (Wei et al., 2023; Zhang et al., 2023). In order to further boost the performance, recent works have laid emphasis on the optimization of triggering prompts (Zhou et al., 2022; Yang et al., 2023). In their work, LLMs are employed as optimizers, and new prompts are progressively generated based on the past optimization history. Despite the augmented performance, the optimization process for prompts reverts to a task-specific problem, and for unseen test questions in real-world circumstances, it may not be advisable to optimize prompts on the fly.

Specific Few-Shot-CoT. Owing to the well-crafted in-context demonstrations, Few-Shot-

[1] Detailed exploration will be provided in Section 3.2.

CoT achieves preferable performance, which consequently extends to a plethora of studies focusing on improvements upon it. According to the period of improvement, these studies are grouped into three categories: (i) pre-reasoning pattern; (ii) peri-reasoning pattern; and (iii) post-reasoning pattern.

For the pre-reasoning pattern, current research attends to either alleviating manual labor when selecting demonstrations (Zhang et al., 2023; Wan et al., 2023), or promoting demonstration quality (Creswell et al., 2023; Madaan and Yazdanbakhsh, 2022; Arora et al., 2023; Diao et al., 2023; Wang et al., 2023b). For the post-reasoning pattern, recent studies concentrate on fine-grained reasoning processes such as problem decomposition (Zhou et al., 2023; Press et al., 2022). For the post-reasoning pattern, related works principally enhanced the performance by verification (Weng et al., 2022; Lyu et al., 2023) or ensemble-like methods (Wang et al., 2023a; Li et al., 2023; Wang et al., 2022; Yoran et al., 2023).

However, the aforementioned works, which mainly hinge on task-associated demonstrations, fail to step outside the task-specific framework to pursue generalizability. In turn, there is an upper bound to the performance that a general Zero-Shot-CoT method can achieve, thus leading the current CoT prompting to a dilemma. Our work, in contrast, manages to find a way out of this dilemma by intuitively carrying out a routing mechanism, making our proposed GeM-CoT applicable in realistic mixed-task scenarios.

2.2 Cross-task Generalization

Cross-task generalization has been a long-standing research goal in natural language processing (NLP). The conventional pre-training and fine-tuning paradigm gains a foothold by pre-training on a large corpus of text to capture general knowledge and fine-tuning on specific tasks to acquire specific knowledge. Beyond this primitive paradigm, post pre-training and multi-task learning (Yu et al., 2022; Zhang and Zhao, 2021; Liu et al., 2019; Zhang et al., 2022) encourage further advancements in this research area. More recent works such as ExT5 (Aribandi et al., 2022), T0 (Sanh et al., 2022), and FLAN (Wei et al., 2022) strived to convert a variety of tasks into an identical text-to-text format, so that models can be trained on those tasks jointly. LoraHub (Huang et al., 2023) leveraged the composability of LoRA (Low-Rank Adaption of LLMs) modules to promote the task generalization ability of LLMs. Our work, however, manages to effectuate task generalization through timely and user-friendly ICL without any training.

3 Towards Generalizable CoT in Mixed-task Scenarios

In this section, we first define the concept of mixed-task scenarios and then present preliminary experiments to understand the challenge.

3.1 Concept of Mixed-task Scenarios

Existing studies (Wei et al., 2023) commonly assume that the type of questions fed to the model is known and conduct each set of evaluations on the questions from the same dataset, which is regarded as the single-task scenarios. However, a more realistic setting lies in **mixed-task scenarios** where the type of input questions is unknown and they come in an arbitrary manner. A comparison with the single-task scenarios is presented in Table 1.

Setting	Unknown Type	Mixed Source	Arbitrary Order
Single-task Scenarios	✗	✗	✗
Mixed-task Scenarios	✓	✓	✓

Table 1: Concept of **mixed-task scenarios**, which is more common in real-world situations.

Mixed-task scenarios have three main characteristics: (i) the type of any incoming question is unknown; (ii) the input data comes from a set of mixed tasks; (iii) the questions come in an arbitrary order. Such a setting is of pivotal importance because the specific task source of an incoming question is usually unavailable in many real-world applications.

3.2 Challenge of Mixed-task Scenarios

In the first place, we set up the mixed-task scenarios by adopting questions from ten reasoning tasks following Kojima et al. (2023) and Zhang et al. (2023). We shuffle all the questions and sample 100 examples to mimic their mixed and arbitrary pattern. We initially adopt two vanilla methods: Zero-Shot-CoT and Few-Shot-CoT,[2] the latter assuming a known dataset source for the input question, which cannot be applied to the mixed-task scenarios, but only serves a hypothetical upper bound for reference.

[2]We leverage ICL demonstrations from Wei et al. (2023) and refer them as *gold demos*.

Method	Mixed-task Scenarios	Accuracy
Few-Shot-CoT (w/ gold)	✗	78.0
zero-shot setting		
Zero-Shot-CoT	✓	66.0 (↓ 12.0)
few-shot setting		
w/ varied & single	✓	26.0 (↓ 52.0)
w/ varied & mixed	✓	20.0 (↓ 58.0)
w/ fixed & single	✓	27.0 (↓ 51.0)
w/ fixed & mixed	✓	19.0 (↓ 59.0)

Table 2: Results with initial attempts showing the challenge of mixed-task scenarios.

As seen in Table 2, the few-shot setting with gold demonstrations substantially outperforms the zero-shot setting (78.0% → 66.0%). Therefore, we focus on the few-shot setting and present four pilot attempts based on two perspectives: (i) *varied* / *fixed*: whether the ICL demonstrations vary for each input question; (ii) *single* / *mixed*: whether the ICL demonstrations originate from a single dataset. We observe catastrophic performance degradation with these naive approaches (e.g., 78.0% → 27.0%). Moreover, we find that the adoption of demonstrations from a single dataset source leads to better performance as the methods with *mixed* demonstrations exhibit subpar performances than those with *single* ones (20.0/19.0% → 26.0/27.0%). This investigation partially inspires us to design a plug-and-play routing module to assign LLMs with demonstrations of a shared type rather than mixed types for subsequent inference.

4 GeM-CoT

Based on the consideration above, we introduce GeM-CoT to tackle mixed-task scenarios. Figure 2 and Figure 3 illustrate its overall architecture and flow chart, respectively.

Concretely, GeM-CoT first routes the input question to different paths (*Type Matching*): (i) **path matched→**: For a successful match, it fetches demonstrations from the demo pool (*Demo Acquisition*) and performs a final inference (*Answer Derivation w/ demos*). (ii) **path unmatched→**: For a failed match, it derives the zero-shot answer with rationales (*Answer Derivation w/o demos*) and then updates the data cache through density-based clustering and automatically constructs demonstrations (*Data Cache Update*). We detail these modules as follows.

4.1 Type Matching

Given a demo pool DP containing n demonstrations $[dm^1, dm^2, \ldots, dm^n]$ and an input question q_{in}, the objective of *Type Matching* is to find the most similar demo question for q_{in} and decide whether this match is successful or not.

Similarity Calculation Note that each demonstration in DP is under the form: $dm^i = (q_d^i, r_d^i, a_d^i, t_d^i)$, where r_d^i, a_d^i, t_d^i refer to the rationale, answer and type of q_d^i. For a demo question $q_d^i \in dm^i$ and the input question q_{in}, we encode them independently using the same model Enc and employ the dot product of their representations as the similarity score:

$$sim(q_{in}, q_d^i) = \left\langle Enc(q_{in}), Enc(q_d^i) \right\rangle, \quad (1)$$

where $\langle , \rangle$ denotes the dot product operation.

Match Decision After obtaining n scores, we select the demonstration $dm_{sim} = (q_{sim}, r_{sim}, a_{sim}, t_{sim})$ that has the highest similarity score with q_{in}: $S = sim(q_{in}, q_{sim})$. Then we compare S with a constant threshold S_{thres} to make a matching decision D_{match}:

$$D_{match} = \begin{cases} 0, & \text{if } S \geq S_{thres} \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

For a successful match (i.e., $D_{match} = 0$), we follow the path: *Demo Acquisition* (§ 4.2) → *Answer Derivation w/ demos* (§ 4.3). For a failed match (i.e., $D_{match} = 1$), we choose the path: *Answer Derivation w/o demos* (§ 4.3) → *Data Cache Update* (§ 4.4).

4.2 Demo Acquisition

After successfully matching the input question q_{in} with a certain type t_{sim} in § 4.1, we are able to construct type-wise demonstrations for in-context learning: $DEM_q = \left[dm_q^1, dm_q^2, \ldots, dm_q^p\right]$, where p denotes the number of demonstrations under the type t_{sim} in DP.

4.3 Answer Derivation

w/ demos Now that we have p demonstrations of the formerly matched type t_{sim} acquired in § 4.2, we execute a final inference to obtain the answer to q_{in}. Specifically, each demonstration $dm_q^i \in DEM_q$ is formatted as: $\left[\text{Q: } q^i, \text{A: } r^i, a^i\right]$ where q^i, r^i, and a^i are from dm_q^i. Then we prepare the templated input prompt for inference by $P_{inf} = [\text{Q: } q_{in}, \text{A: }]$. After that, the formatted

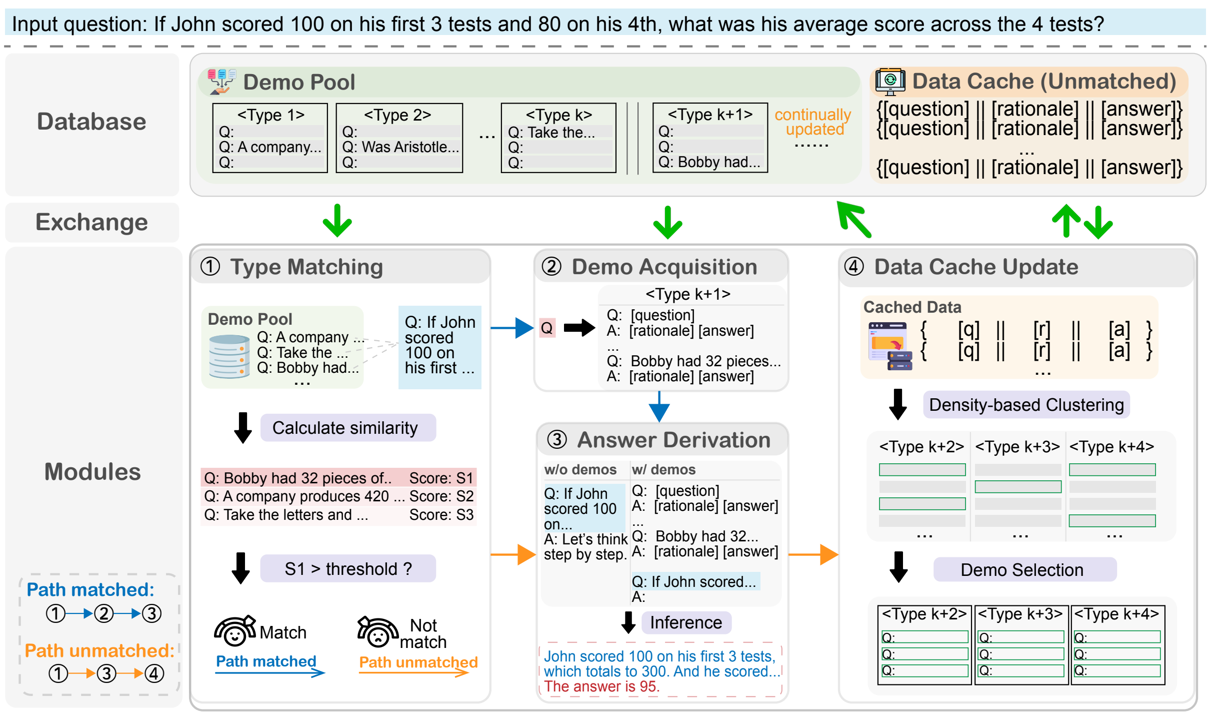


Figure 2: Overview of our proposed GeM-CoT mechanism. GeM-CoT first routes the input question to different paths (*Type Matching*): i) **path matched**→: For a successful match, it fetches demonstrations from the demo pool (*Demo Acquisition*) and performs a final inference (*Answer Derivation*). ii) **path unmatched**→: For a failed match, it derives the zero-shot answer with rationales (*Answer Derivation*) and then updates the data cache through density-based clustering and automatically constructing demonstrations (*Data Cache Update*).

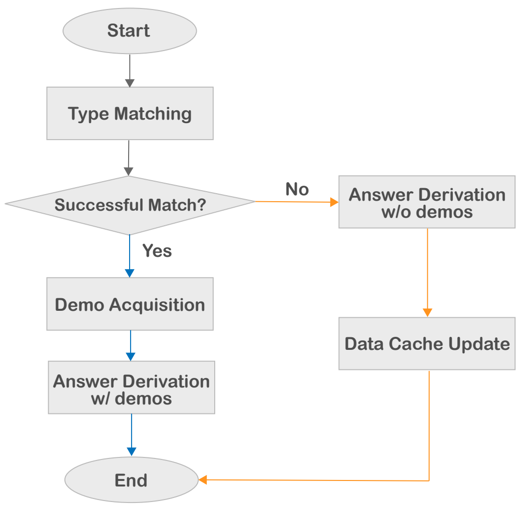


Figure 3: Flow chart of our GeM-CoT mechanism.

demonstrations are concatenated and inserted before the input prompt P_{inf}, which is eventually delivered to LLMs to derive the rationale r_{in} and answer a_{in} of input question q_{in}.

w/o demos In the case of a failed match, we directly invoke Zero-Shot-CoT (Kojima et al., 2023) to obtain the rationale r_{in} and answer a_{in} for the input question q_{in}. Afterward, the data (q_{in}, r_{in}, a_{in}) is returned to the data cache DC, which stores the data that undergoes a failed match with the demo pool DP in *Type Matching* module.

4.4 Data Cache Update

Given the data cache DC that encompasses m data $[cad^1, cad^2, \ldots, cad^m]$, the goal of *Data Cache Update* is to execute a density-based clustering upon the questions therein and select high-quality demonstrations for each cluster that meet certain requirements. The overall procedure of this module is presented in Algorithm 1.

Density-based Clustering Since the types of data in DC are unknown and mixed, we cannot know in advance the number of clusters into which these questions should be classified. To this end, we adopt the density-based clustering algorithm OPTICS (Ankerst et al., 1999).[3] Concretely, we first encode all the questions $\{q_c^i \in cad^i, i \in [1, \ldots, m]\}$ in DC with the model Enc and then

[3]This algorithm is capable of detecting meaningful clusters in data of varied density, and this feature fits our novel setting well, where the questions are mixed and unbalanced in type.

Algorithm 1: Data Cache Update

```
Input: demo pool DP, data cache DC, cached data
       $[cad^1, cad^2, \ldots, cad^m]$, threshold numbers
       $\{th_{ca}, th_{cls}\}$, density-based clustering
       function $\mathcal{OPTICS}$, demo selection function
       $\mathcal{SEL}$, function that returns cluster size $\mathcal{S}$,
Output: demo pool DP, data cache DC
if $n \geq th_{ca}$ then
    $[cls^1, cls^2, \ldots, cls^t] \leftarrow$
      $\mathcal{OPTICS}([cad^1, cad^2, \ldots, cad^m])$
    for $i$ in $1, ..., t$ do
        $num \leftarrow \mathcal{S}(cls^i)$
        if $num \geq th_{cls}$ then
            $demos \leftarrow \mathcal{SEL}(cls^i)$
            Add $demos$ to DP
            Remove $cls^i$ from DC
        end
    end
end
return DP, DC
```

perform OPTICS upon them to obtain t clusters:

$$\begin{gathered}\mathcal{C}_{emb} = Enc\left(\left[q_c^1, q_c^2, \ldots, q_c^m\right]\right), \\ \left[cls^1, cls^2, \ldots, cls^t\right] = \text{OPTICS}(\mathcal{C}_{emb}).\end{gathered} \tag{3}$$

Demo Selection After obtaining t clusters, we conduct a filtering and focus only on clusters whose size is no less than a threshold th_{cls}. For each filtered cluster cls^i, we leverage the encoder model Enc to obtain a vector representation for each candidate question in cls^i. After that, we perform k-means clustering over the acquired contextualized representations. We sort the questions in ascending order by distance from the cluster center. Next, we follow prior works (Zhang et al., 2023) to conduct simple operations on the question and rationale [4], which help obtain more effective demonstrations. Once the *question-rationale* pair is retained under the operation, we stop functioning on other questions in cls^i. As a result, we manage to collect a total of k representative and high-quality demonstrations for cls_i: $\left[\left(q^1, r^1, a^1\right), \left(q^2, r^2, a^2\right), \ldots, \left(q^k, r^k, a^k\right)\right]$, where r^j and a^j refer to the rationale and answer of q^j. In the end, we update the demo pool DP with the generated diverse demonstrations and remove the data of cls^i from the data cache DC.

5 Experiments

This section will describe our experimental setup and present the main results.

[4]More details are attached in Appendix A.1

5.1 Setup

Datasets. We evaluate our method on 10 reasoning datasets and a suite of 23 BIG-Bench Hard (BBH) tasks. The former is the basis of the original demo pool construction, whereas the latter can be regarded as questions of *unseen*[5] types for our mechanism. The 10 reasoning datasets include AQUA-RAT (Ling et al., 2017), MultiArith (Roy and Roth, 2015), AddSub (Hosseini et al., 2014), GSM8K (Cobbe et al., 2021), SingleEq (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021), Last Letter Concatenation (Wei et al., 2023), Coin Flip (Wei et al., 2023), StrategyQA (Geva et al., 2021), and CSQA (Talmor et al., 2019). For the BBH (Suzgun et al., 2022) tasks, we shuffle all the data and randomly sample 2000 questions to imitate the realistic mixed-task scenarios.[6]

Implementation. We utilize the popular and publicly available models GPT-3.5-Turbo and GPT-4 (OpenAI, 2023) from OpenAI API[7]. The temperature and *top_p* are both set to 1.0. The original demo pool DP is constructed based on the data from Wei et al. (2023). The threshold numbers S_{thres}, th_{ca} and th_{cls} are set to 0.35, 200 and 50 respectively. We employ Sentence-BERT (Reimers and Gurevych, 2019) as the encoder model Enc. We perform the density-based clustering and k-means clustering through the open-source scikit-learn[8] python package. We set the number of demonstrations k to 6 for simplicity when constructing demonstrations for a new type, since this number generally achieves decent performance on reasoning datasets (Wei et al., 2023).

Baselines. We compare GeM-CoT with 6 baselines, which can be divided into three groups: (i) ICL methods without CoT prompting (Kojima et al., 2023; Brown et al., 2020); (ii) task-specific CoT approaches (Wei et al., 2023; Zhang et al., 2023); (iii) CoT techniques with generalization (Kojima et al., 2023). Specifically, we devise a strong baseline named General-CoT for generalization comparison. It randomly collects one demonstration from each type of data in the demo pool DP and then leverages the gathered demonstrations as a generic inference prompt for

[5]Here *unseen* means there are no questions in the original demo pool that match the BBH tasks.

[6]Details about BBH tasks is presented in Appendix B.2.

[7]https://openai.com/blog/openai-api

[8]https://scikit-learn.org/stable/

Table 3: Accuracy (%) on ten reasoning datasets. The backbone model is GPT-3.5-Turbo. Results in **bold** and underline are the best and second-best performances, respectively.

Method	**Mixed-task Scenarios**	**AQuA**	**MultiArith**	**AddSub**	**GSM8K**	**SingleEq**	**SVAMP**	**Letter**	**Coin**	**Strategy**	**CSQA**	**Avg.**
**ICL methods without CoT*												
Zero-Shot	✓	29.1	67.2	88.9	36.9	86.5	67.9	4.8	44.0	**65.3**	74.3	56.5
Few-Shot	✗	33.1	87.5	91.1	48.9	92.7	79.1	7.2	64.4	62.3	**81.0**	64.7
**Task-specific CoT approaches*												
Few-Shot-CoT	✗	**54.3**	97.3	93.9	76.5	96.7	81.9	73.2	99.0	63.7	78.0	81.4
Auto-CoT	✗	49.6	**99.3**	**94.2**	**78.9**	96.3	84.6	81.2	**100.0**	64.6	72.2	82.1
**CoT techniques with generalization*												
Zero-Shot-CoT	✓	51.6	94.7	85.5	72.7	93.5	78.4	**85.8**	99.0	62.6	69.9	79.4
General-CoT	✓	46.9	98.7	92.4	77.2	97.4	83.8	75.2	**100.0**	63.4	72.2	80.7
GeM-CoT(**Ours**)	✓	51.9	99.0	93.7	77.5	**98.4**	**88.6**	77.2	**100.0**	63.5	72.8	**82.3**

Table 4: Accuracy (%) on four reasoning datasets. The backbone model is GPT-4.

Methods	**AQuA**	**GSM8K**	**SVAMP**	**Avg.**
Zero-shot-CoT	70.5	81.3	91.3	81.0
Few-shot-CoT	71.9	92.0	90.5	85.5
GeM-CoT(**Ours**)	**72.8**	**93.6**	**93.7**	**86.6**

all the input data.[9] More baseline details are presented in Appendix A.2.

5.2 Main Results

Performance on reasoning datasets. Table 3 presents the results on ten reasoning tasks. GeM-CoT generally towers above the baseline methods from different angles. On one hand, compared with two typical task-specific CoT approaches, GeM-CoT not only averagely surpasses them in performance but also enjoys the generalizable property, which means that the input question with an unknown type can be adapted to our method in an automatic and labor-free pattern. On the other hand, while the general CoT techniques both witness average performance degradation (i.e., 82.1%→79.4/80.7%), GeM-CoT stands out by continually boosting the performance (i.e., 82.1%→82.3%), thus shedding light on the mutual synergy between generalization and performance.

Performance on BBH datasets. As our proposed GeM-CoT is adept at tackling incoming questions of *unseen* types with its continuously updating databases, we set up a more realistic and complex streaming setting (Tang, 2023), where the original test set is not visible and the questions

[9]The generic inference prompt is constructed from the original demo pool DP without subsequent updates.

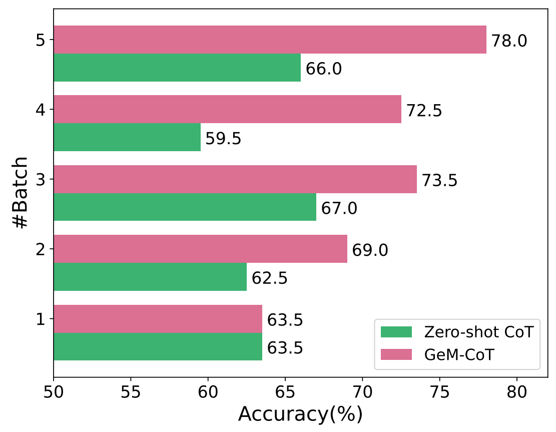


Figure 4: Process of five subsequent streaming batch data with batch size of 400 on BBH datasets.

appear in the form of batch data. As illustrated in Figure 4, the superiority of GeM-CoT gets prominent from batch 2, suggesting that as the data amount increases, our approach enjoys broader adaptability and higher generality by learning more representative and fine-grained features.

6 Analysis

6.1 Methods of Selecting Demonstrations

Since our work is situated in realistic mixed-task scenarios, accessing high-quality demonstrations in a labor-saving pattern is of crucial importance. Accordingly, we select two representative labor-free methods for comparison: (i) Similarity-based, which retrieves the top-k similar questions based on cosine similarity; (ii) Randomness-based, which randomly samples k examples for each input question. Results in Table 5 show our proposed GeM-CoT (diversity-based) performs the best, verifying the importance of diversity in demonstrations.

Table 5: Influence of demonstration selection methods. Our proposed GeM-CoT method is based on diversity-based demonstration selection.

Method	AQuA	AddSub	Strategy	Coin
GeM-CoT	**51.9**	**93.7**	63.5	**100.0**
w/ similarity	49.6	90.1	**64.1**	99.2
w/ randomness	52.0	92.2	61.2	99.0

Table 6: Effect of *Type Matching* module. Applicability stands for whether the method is applicable to our proposed mixed-task scenarios.

Method	Applicability	Cost-free	AddSub	Strategy
GeM-CoT	✓	✓	**93.7**	63.5
w/ classifier	✓	✗	93.4	64.5
w/ correct type	✗	✓	90.1	**65.0**

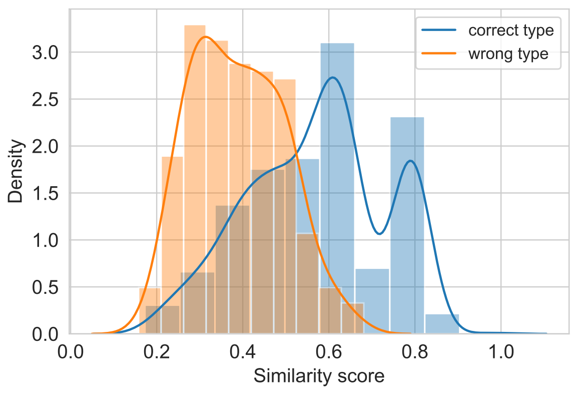


Figure 5: Distribution of similarity scores in *Type Matching* module. We separately present the distribution of correctly and incorrectly matched scores.

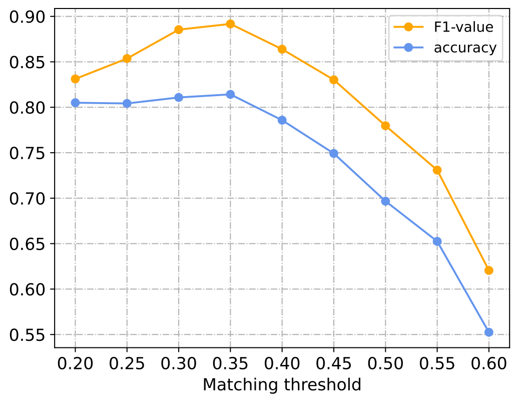


Figure 6: F1 value and accuracy of *type matching* with respect to varying matching thresholds.

6.2 Effect of *Type Matching* module

In order to further explore the effect of *Type Matching* which plays a key role in generalization, we discard this module and adopt two alternatives: (i) an LLM-based classifier that groups the questions based on its *category* and *form* using few-shot examples in the prompt;[10] (ii) an idealized strategy in which we assume that the model is given the gold type, noting that this case does not apply to our proposed mixed-task scenarios, and serves only as a reference for comparison. Results are presented in Table 6. Compared with the LLM-based classifier, GeM-CoT not only achieves comparable performance but also relieves the need for any API cost. In addition, GeM-CoT bears stronger generalization capabilities because the matching is based on semantic similarity, eliminating the effort of defining and updating the question *type* in the prompt.

6.3 Choice of Matching Threshold

We provide further analysis to validate the rationality of the chosen threshold for the *Type Matching* module. We focus on a total of 1200 questions from ten reasoning datasets (Wei et al., 2023), from which the original demo pool is constructed so that we can easily determine if the match types are correct or not. Figure 5 presents the distribution of correctly and incorrectly matched scores, which are concentrated in the $[0.2, 0.6]$ range. We select the scores within this range as the threshold and calculate the corresponding F1 value and accuracy. As shown in Figure 6, choosing 0.35 yields the best results in general across our tasks.

7 Conclusion

In this work, we initially put forward a novel setting with significant application values, namely mixed-task scenarios where the questions come in a mixed and arbitrary way with their types unknown. Upon this challenging setting, we propose GeM-CoT, a generalizable CoT prompting mechanism that first performs type matching and then automatically samples or constructs corresponding ICL demonstrations, with continuously updated databases. Evaluation results on a total of 33 datasets demonstrate the impressive performance and superior generality of our proposed method. While most existing works focus on either promoting performance or pursuing generality, we open up a pioneering perspective to bridge the two aspects in a simple and practical manner.

[10]We construct the few-shot examples from the ten reasoning datasets following (Wei et al., 2023). More information about how to define the *category* and *form* is presented in Appendix C.

Limitations

There are two limitations. First, our proposed approach focuses on the application of CoT methods to a novel and practical scenario while ignoring the improvement of the reasoning process to a certain extent. As discussed in Related Work, existing reasoning improvement approaches can be further applied to strengthen GeM-CoT. Second, there might be more efficient ways of selecting high-quality ICL demonstrations in our proposed mixed-task scenarios, which is left to be further explored in future works.

References

Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Prakash Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. 2022. Ext5: Towards extreme multi-task scaling for transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. 2023. Ask me anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311.

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*, abs/2110.14168.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *ArXiv preprint*, abs/2302.12246.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *ArXiv preprint*, abs/2212.10071.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *ArXiv preprint*, abs/2307.13269.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting* *of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *ArXiv preprint*, abs/2301.13379.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *ArXiv preprint*, abs/2209.07686.

OpenAI. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *ArXiv preprint*, abs/2210.03350.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *ArXiv preprint*, abs/2210.09261.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuxin Tang. 2023. Chain-of-thought prompting under streaming batch: A case study. *arXiv*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *ArXiv preprint*, abs/2201.08239.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan Arik, and Tomas Pfister. 2023. Better zero-shot reasoning with self-adaptive prompting. In *Findings of the Association for Computational Linguistics:*

ACL 2023, pages 3493–3514, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *ArXiv preprint*, abs/2207.00747.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023b. Meta-reasoning: Semantics-symbol deconstruction for large language models. *arXiv preprint arXiv:2306.17820*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification. *ArXiv preprint*, abs/2212.09561.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *ArXiv preprint*, abs/2309.03409.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. *ArXiv preprint*, abs/2304.13007.

Wenhao Yu, Chenguang Zhu, Yuwei Fang, Donghan Yu, Shuohang Wang, Yichong Xu, Michael Zeng, and Meng Jiang. 2022. Dict-BERT: Enhancing language model pre-training with dictionary. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1907–1918, Dublin, Ireland. Association for Computational Linguistics.

Zhuosheng Zhang, Shuohang Wang, Yichong Xu, Yuwei Fang, Wenhao Yu, Yang Liu, Hai Zhao, Chenguang Zhu, and Michael Zeng. 2022. Task compass: Scaling multi-task pre-training with task prefix. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5671–5685, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.

Zhuosheng Zhang and Hai Zhao. 2021. Structural pre-training for dialogue comprehension. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5134–5145, Online. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Experimental Details

A.1 Implementation details

Filtering operations in *Demo Selection*. We follow the works from (Wei et al., 2023; Zhang et al., 2023) to filter the *question-rationale* pair as follows: the question needs to be no more than 60 tokens and the rationale should not exceed 5 reasoning steps. The objective of this filtering strategy is to seek simple heuristics by sampling simpler questions and rationales.

A.2 Baseline Methods

We introduce the baseline methods in detail.

- **ICL methods without CoT**: Zero-Shot (Kojima et al., 2023) adds the prompt "A: The answer is" to an input question and leverage it as the input delivered to LLMs. Few-Shot (Brown et al., 2020) employs several additional templated demonstrations as: [Q: q, A: The answer is a] before the input question, where q and a are manually crafted questions and answers.
- **Task-specific CoT approaches.**: Few-Shot-CoT (Wei et al., 2023) follows similar patterns as Few-Shot but differs in that rationales are inserted before deriving the answer. Auto-CoT (Zhang et al., 2023) divides questions of a given dataset into a few clusters, samples a representative question from each cluster, and constructs its reasoning chain using Zero-Shot-CoT with simple heuristics.

- **CoT techniques with generalization**: Zero-Shot-CoT (Kojima et al., 2023) simply inserts the prompt *Let's think step by step* after a question to conduct inference, which rids the necessity of handcrafted task-wise demonstrations. We also compare our method with a strong baseline General-CoT, in which the in-context demonstrations for inference come from distinct question groups.

B Dataset Information

B.1 Reasoning Datasets

Our method is evaluated on 10 reasoning benchmark datasets that cover three categories including arithmetic, commonsense and symbolic tasks and involve three forms encompassing short-answer, multiple-choice, and yes-or-no questions. The corresponding categories and forms of these datasets are shown in Table 7.

- **Arithmetic Reasoning**: we choose the following six datasets: (i) MultiArith (Roy and Roth, 2015), (ii) GSM8K (Cobbe et al., 2021), (iii) AddSub (Hosseini et al., 2014), (iv) AQUA-RAT (Ling et al., 2017), (v) SingleEq (Koncel-Kedziorski et al., 2015), and (vi) SVAMP (Patel et al., 2021). MultiArith, AddSub, and SingleEq come from the Math World Problem Repository (Koncel-Kedziorski et al., 2016), while the other three are from more contemporary benchmarks. Among them, all the arithmetic datasets belong to short-answer form except for AQUA-RAT which is in multiple-choice format.
- **Commonsense Reasoning**: we take the following two datasets into account: (i) CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). CSQA poses difficult questions with rich semantic relations by making use of ConceptNet (Talmor et al., 2019). StrategyQA requires models to derive answers using implicit reasoning steps (Geva et al., 2021). CSQA is in multiple-choice form whereas StrategyQA belongs to the yes-or-no format.
- **Symbolic Reasoning**: we employ the typical datasets Last Letter Concatenation and Coin Flip from Wei et al. (2023), which are in short-answer and yes-or-no form respectively. Last Letter Concatenation asks the model to concatenate the last letters of each word. Coin Filp requires the model to answer whether a coin heads up after a series of actions of either flipping or not flipping the coin.

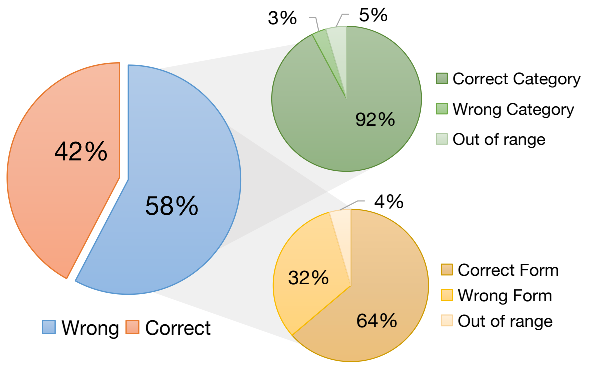


Figure 7: Ratio of wrong cases in task classification.

B.2 BBH Datasets

We further evaluate our method on a suite of 23 BBH tasks, the questions of which can be regarded as *unseen* types for our proposed mechanism. The detailed information about these BBH datasets are listed in Table 8.

C LLM-based classifier in *Type Matching*

We detail the implementations and provide extended analysis on the alternative in *Type Matching* module: the LLM-based classifier. The proposed classifier employs few-shot examples in the prompt to group the questions based on its *category* and *form*. To implement the LLM-based classifier, we need to ensure the appropriate way of defining the *type* of questions.

C.1 Defining the *Type* of Questions.

As stated in Section 3.2, we have collected questions from ten reasoning tasks to set up the mixed-task scenarios. Those questions cover three categories including arithmetic, commonsense, and symbolic reasoning, and three forms encompassing short-answer, multiple-choice, and yes-or-no questions. Initially, we make a simple attempt to test how well LLMs can identify various tasks (i.e., regarding the question type as task name). We randomly sample one question from each of the ten tasks. For each question, we retain the task name from which it originates so that we obtain ten question-task pairs, which we employ as ICL demonstrations for task classification. As can be seen from Figure 7, the classification accuracy is only 42%, which indicate that LLMs are not qualified for distinguishing task names. Meanwhile, we discover that up to 92% and 64% of wrong examples belong to the same category and form as the correct task respectively. We speculate

Table 7: Information of 10 reasoning datasets (Ari.: arithmetic; Com.: commonsense and Sym.: symbolic; SAQ: short-answer question; MCQ: multiple-choice question; Y/N: yes-or-no question).

Task	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	CSQA	Strategy	Letter	Coin
Category	Ari.	Ari.	Ari.	Ari.	Ari.	Ari.	Com.	Com.	Sym.	Sym.
Form	SAQ	SAQ	SAQ	MCQ	SAQ	SAQ	MCQ	Y/N	SAQ	Y/N
Size	600	1319	395	254	508	1000	1221	2290	500	500

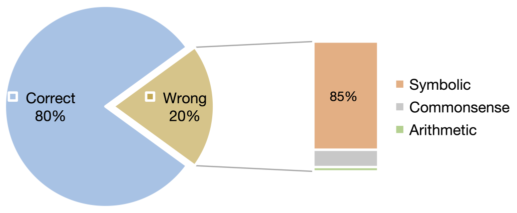


Figure 8: Ratio of wrong cases in category classification, 85% of wrong cases are from symbolic category.

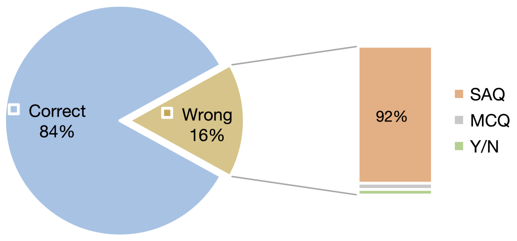


Figure 9: Ratio of wrong cases in form classification, 92% of wrong cases are from SAQ form.

that the underlying reason can be two-fold: on one hand, task names themselves are too abstract for LLMs to well perceive their differences through in-context learning alone. On the other hand, there exist potential similarities and correlations among tasks themselves (Zhang et al., 2022). Based on this, we try three schemes for defining the type of questions based on: (i) category; (ii) form; (iii) category and form.

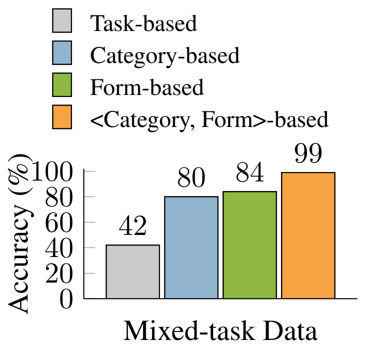


Figure 10: Classification accuracy (%) with different partitioning schemes.

C.2 Determining the *Type* of Questions.

Since the majority of cases that misidentify task names fall into the same category or form, we compare the classification accuracy with the following three variants of partitioning schemes: (i) Category-based scheme which separates mixed questions into diverse categories; (ii) Form-based scheme which segments data into different answer forms; (iii) <Category, Form>-based scheme which concurrently takes the two aspects into account. As is shown in Figure 8 and 9, we particular group tends to dominate the wrong cases. For instance, 85% of wrong cases in category classification belong to the symbolic group. We discover that this is because the sampled symbolic group demonstrations do not cover symbolic yes-or-no question, thus hindering LLMs from accurately identifying this missing type. As such, partitioning mixed questions based on both its category and form is a sensible strategy. The results in Figure 10 show that this strategy reaches high accuracy.

Through further experiments, we conclude that defining the type of questions based on its **category and form** is a sensible strategy, which adequately considers the two major natures of question data and achieves high classification accuracy as well.

C.3 Constructed Demonstrations for the LLM-based classifier

Table 9 shows the constructed demonstrations for the LLM-based classifier.

D Comparisons of GeM-CoT and existing CoT methods

Table 10 demonstrate the comparisons of our proposed GeM-CoT and existing CoT methods in an intuitive and multi-facet way.

E Interpretability: Case Study and Error Analysis

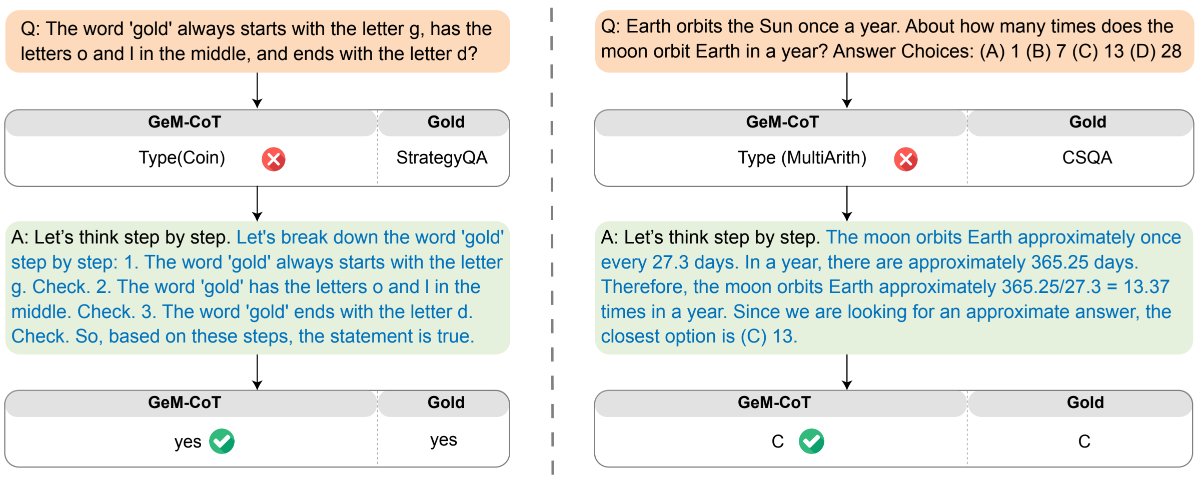
E.1 Wrong Type and Correct Answer

Figure 11 illustrates two examples from StrategyQA and CSQA, in which the type that GeM-

CoT identifies differs from the gold type but the final answer from our proposed method is correct. We observe that the proposed *type matching* phase manages to capture the type where the unseen input question is applicable in a more accurate and reasonable way. For instance, the question from StrategyQA (left in Figure 11) asks whether *the word 'gold' always starts with the letter g, has the letters o and l in the middle, and ends with the letter d*. Although this question belongs to a commonsense question, to answer it would require a process of splitting the word, which has more in common with a symbolic question. Similarly, answering the question from CSQA (right in Figure 11) necessitates a calculation process, and thus the identified *arithmetic* type leads to more specific and targeted arithmetic reasoning.

E.2 Wrong Type and Wrong Answer

We select two examples from StrategyQA, where GeM-CoT fails but the strategy that provides the model with the gold type succeeds. As is shown in Figure 12, we find that some wrongly identified types may result in disastrous reasoning. We analyze that this may be because incorrect ICL demonstrations will disrupt the direction of model inference.


Q: The word 'gold' always starts with the letter g, has the letters o and l in the middle, and ends with the letter d?

GeM-CoT	Gold
Type(Coin) ✗	StrategyQA

A: Let's think step by step. Let's break down the word 'gold' step by step: 1. The word 'gold' always starts with the letter g. Check. 2. The word 'gold' has the letters o and l in the middle. Check. 3. The word 'gold' ends with the letter d. Check. So, based on these steps, the statement is true.

GeM-CoT	Gold
yes ✓	yes

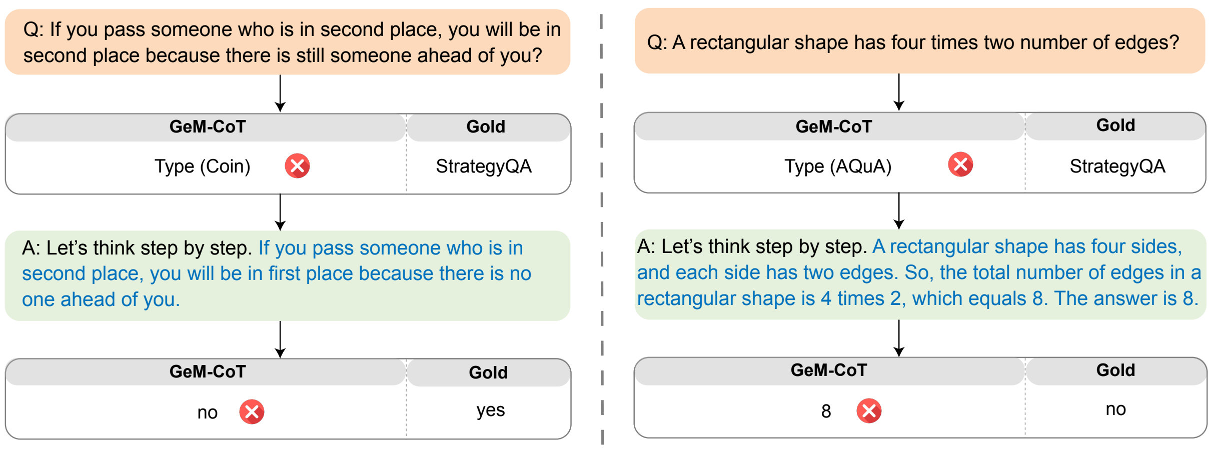
Q: Earth orbits the Sun once a year. About how many times does the moon orbit Earth in a year? Answer Choices: (A) 1 (B) 7 (C) 13 (D) 28

GeM-CoT	Gold
Type (MultiArith) ✗	CSQA

A: Let's think step by step. The moon orbits Earth approximately once every 27.3 days. In a year, there are approximately 365.25 days. Therefore, the moon orbits Earth approximately 365.25/27.3 = 13.37 times in a year. Since we are looking for an approximate answer, the closest option is (C) 13.

GeM-CoT	Gold
C ✓	C

Figure 11: Examples from StrategyQA (left) and CSQA (right), in which the type that GeM-CoT identifies is different from the gold type but the final answer from GeM-CoT is correct.


Q: If you pass someone who is in second place, you will be in second place because there is still someone ahead of you?

GeM-CoT	Gold
Type (Coin) ✗	StrategyQA

A: Let's think step by step. If you pass someone who is in second place, you will be in first place because there is no one ahead of you.

GeM-CoT	Gold
no ✗	yes

Q: A rectangular shape has four times two number of edges?

GeM-CoT	Gold
Type (AQuA) ✗	StrategyQA

A: Let's think step by step. A rectangular shape has four sides, and each side has two edges. So, the total number of edges in a rectangular shape is 4 times 2, which equals 8. The answer is 8.

GeM-CoT	Gold
8 ✗	no

Figure 12: Examples from StrategyQA, in which wrongly identified type leads to wrong answer.

Table 8: Information of 23 BBH datasets. Categories and descriptions about the datasets are from Suzgun et al. (2022). (Algo.+Ari.: Algorithmic and Multi-Step Arithmetic Reasoning; NLU: Natural Language Understanding; Knowledge: Use of World Knowledge).

Task	Category	Description
Boolean Expressions	Algo.+ Ari.	Evaluate the truth value of a random Boolean expression consisting of Boolean constants (`True`, `False`) and basic Boolean operators (`and`, `or` and `not`).
Causal Judgement	Knowledge	Given a short story (involving moral, intentional, or counterfactual analysis), determine how a typical person would answer a causal question about the story.
Date Understanding	Knowledge	Given a small set of sentences about a particular date, answer the provided question (e.g., "The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date yesterday in `MM/DD/YYYY`?").
Disambiguation QA	NLU	Given a sentence with an "ambigious" pronoun, either determine whether the sentence is inherently ambiguous (i.e., the thing that the pronoun refers to cannot be inferred by given information) or, if the pronoun can be implicitly deduced, state the antecedent of the pronoun (i.e., the noun to which the pronoun refers).
Dyck Languages	Algo.+ Ari.	Predict the sequence of the closing parentheses of a Dyck-4 word without its last few closing parentheses.
Formal Fallacies	Algo.+ Ari.	Given a context involving a set of statements (generated by one of the argument schemes), determine whether an argument—presented informally—can be logically deduced from the provided context
Geometric Shapes	Algo.+ Ari.	Given a full SVG path element containing multiple commands, determine the geometric shape that would be generated if one were to execute the full path element.
Hyperbaton	NLU	Given two English-language sentences, determine the one with the correct adjective order.
Logical Deduction	Algo.+ Ari.	Deduce the order of a sequence of objects based on the clues and information about their spacial relationships and placements.
Movie Recommendation	Knowledge	Given a list of movies a user might have watched and liked, recommend a new, relevant movie to the user out of the four potential choices user might have.
Multi-Step Arithmetic	Algo.+ Ari.	Solve multi-step equations involving basic arithmetic operations (addition, subtraction, multiplication, and division).
Navigate	Algo.+ Ari.	Given a series of navigation steps to an agent, determine whether the agent would end up back at its initial starting point.
Object Counting	Algo.+ Ari.	Given a collection of possessions that a person has along with their quantities (e.g., three pianos, two strawberries, one table, and two watermelons), determine the number of a certain object/item class (e.g., fruits).
Penguins in a Table	Knowledge	Given a unique table of penguins (and sometimes some new information), answer a question about the attributes of the penguins.
Reasoning about Colored Objects	Algo.+ Ari.	Given a context, answer a simple question about the color of an object on a surface.
Ruin Names	Knowledge	Given an artist, band, or movie name, identify a one-character edit to the name that changes the meaning of the input and makes it humorous.
Salient Translation Error Detection	NLU	Given a source sentence written in German and its translation in English, determine the type of translation error that the translated sentence contains.
Snarks	NLU	Given two nearly-identical sentences, determine which one is sarcastic.
Sports Understanding	Knowledge	Determine whether a factitious sentence related to sports is plausible.
Temporal Sequences	Algo.+ Ari.	Given a series of events and activities a person has completed in the course of a day, determine what time, during the day, they might have been free to perform another activity.
Tracking Shuffled Objects	Algo.+ Ari.	Given the initial positions of a set of objects and a series of transformations (namely, pairwise swaps) applied to them, determine the final positions of the objects.
Web of Lies	Algo.+ Ari.	Evaluate the truth value of a random Boolean function expressed as a natural-language word problem.
Word Sorting	Algo.+ Ari.	Given a list of words, sort them lexicographically.

Table 9: Constructed demonstrations for type classification.

Q: Bobby had 32 pieces of candy. He ate some pieces of candy. If he has 20 pieces of candy left How many pieces of candy did Bobby eat?

Type: <arithmetic, short-answer>

Q: The man took paperwork to other people to consult over it, where was he heading? Answer Choices: (A) desk (B) meeting (C) office (D) table (E) work

Type: <commonsense, multiple-choice>

Q: A coin is heads up. Kristie does not flip the coin. Johnnie flips the coin. Marisa flips the coin. Derick does not flip the coin. Is the coin still heads up? Note that "flip" here means "reverse".

Type: <symbolic, yes-no>

Q: Take the last letters of each words in "Cruz Wilber Marilu Malik" and concatenate them.

Type: <symbolic, short-answer>

Q: A company produces 420 units of a particular computer component every month, at a production cost to the company of $110 per component, and sells all of the components by the end of each month. What is the minimum selling price per component that will guarantee that the yearly profit (revenue from sales minus production costs) will be at least $626,400 ? Answer Choices: (A) 226 (B) 230 (C) 240 (D) 260 (E) 280

Type: <arithmetic, multiple-choice>

Q: Was Aristotle a member of the House of Lords?

Type: <commonsense, yes-no>

Table 10: Typical CoT techniques (ICL: in-context learning; FT: fine-tuning; KD: knowledge distillation). Segment 1: fine-tuning techniques; Segment 2: in-context learning techniques. To the best of our knowledge, our work is the first to apply CoT prompting to mixed-task scenarios with enjoyable generality and superior performance without additional manual labor. In our work, we focus on in-context learning techniques, eliminating the burden of fine-tuning LLMs.

Model	**Training**	**Mixed-task Scenarios**	**w/o Manual Labor**	**w/ Input-related Info.**
Fine-tune-CoT (Ho et al., 2022)	KD	✗	✓	✗
LoRAHub (Huang et al., 2023)	FT	✓	✓	✗
Zero-Shot-CoT (Kojima et al., 2023)	ICL	✓	✓	✗
Few-Shot-CoT (Wei et al., 2023)	ICL	✗	✗	✓
Self-Consistency-CoT (Wang et al., 2023a)	ICL	✗	✗	✓
Least-to-Most Prompting (Zhou et al., 2023)	ICL	✗	✗	✓
Auto-CoT (Zhang et al., 2023)	ICL	✗	✓	✓
Active Prompt (Diao et al., 2023)	ICL	✗	✗	✓
OPRO (Yang et al., 2023)	ICL	✗	✓	✗
GeM-CoT (our work)	ICL	✓	✓	✓