Latent Learningscape Guided In-context Learning

Anonymous ACL submission

Abstract

The growing interest in leveraging large language models is driven by their exceptional imitation and reasoning capabilities. In-context learning (ICL), a streamlined method, has shown potential in boosting these models' performance without modifying their underlying parameters, especially when supplied with suitable demonstrations. However, existing methods mainly choose demonstrations by comparing surface-level semantic similarities (e.g., based on embedding) and fall short of identifying the most fitting ones. This paper introduces the concept of a "latent learningscape", a more nuanced representation that describes the characteristic of the demonstrations. Building on this concept, we develop a results-driven approach to characterize the latent learningscape features of demonstrations, which then inform the creation of more effective prompts. Through comprehensive testing across datasets in arithmetic, commonsense, and symbolic reasoning tasks, our approach outperforms leading models, showing an average increase in scores by 7.4 percentage points.

1 Introduction

The captivating imitation and reasoning abilities exhibited by large language models (Thoppilan et al., 2022; Chowdhery et al., 2022; Brown et al., 2020) have sparked growing interest in their applications. In-context learning (ICL), as a lightweight method that refrains from altering model parameters, is increasingly proved by experiments to be able to enable large language models to achieve impressive results when appropriate demonstrations are supplied. These demonstrations, when appropriately chosen, effectively stimulate the latent capabilities within large models, thereby enhancing overall performance. Kojima et al. (2023) revealed that Large language models (LLMs) are zero-shot reasoners. Wang et al. (2023a) compared and summarized these Zero-Shot methods. On the other hand, Wang et al. (2023c) introduced a method with different decoding strategy, significantly improved LLMs' performance. Zhang et al. (2022) introduced a demonstrations selecting methods based on text similarity and clustering to make LLMs better few-shots reasoners. However, to fully harness the impact of demonstrations, it is imperative to share same knowledge points, problem-solving approaches, or concepts with test question, rather than merely similar word or sentence embedding.

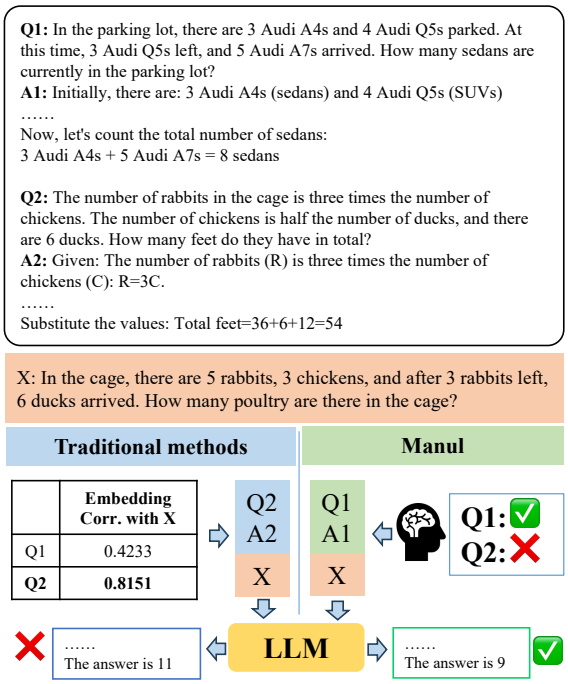

Figure 1: When facing problem X, traditional approach is to select demonstrations based on the similarity of sentence embedding, which is similar in surface-level. However, Q1 truly serves as a demonstration because Q1 and X share the same learningscape.

We first introduce the term **latent learningscape**, which refers to the knowledge points, problem-solving approaches, or concepts that aid in solving problems. As depicted in Fig. 1, unlike demonstrations selected solely based on the similarity of

surface-level semantics by embedding-based methods, Q1 share a same problem-solving approach with X, could truely serve as a effective demonstration. In other words, Q1 and X share a common latent learningscape. The problem-solving approach mentioned here is one of the latent learningscapes we refer to. When demonstrations possess or encompass the latent learningscape inherent in the test question, demonstrations can play a effective role.

In leveraging ICL, we encounter the following challenges: (1) How to identify the latent learningscape and the latent learningscape features behind demonstrations. (2) How to construct prompts that encompass sufficient latent learningscape features for correctly solving reasoning problems.

In light of the challenges mentioned earlier, identifying the latent learningscape behind demonstrations without human intervention is difficult. However, it is possible to characterize the latent learningscape features of demonstrations by defining the differences between demonstrations. Therefore, we propose a result-driven prompt construction method, without the explicit identifying the latent learningscape behind demonstrations. Instead, we leverage the corrective nature of demonstrations to characterize the latent learningscape. Specifically, we first establish two distinct question pools: the A-pool, containing questions answered correctly by the LLM, and the C-pool, containing challenging questions answered incorrectly by the LLM. Next, we use questions from the A-pool as demonstrations to answer the questions in the C-pool, aiming to explore the representational capabilities of the demonstrations in the A-pool. We store the latent features of these demonstrations in vector form and use them to distinguish between demonstrations. Then, we construct prompts using linearly independent vector groups to encompass a comprehensive range of latent learningscape features. Finally, the answer to the current problem is obtained by a majority vote among the different groups of demonstrations defined by different linearly independent vectors.

To demonstrate the effectiveness of our method, we conduct extensive experiments on reasoning problem datasets, including arithmetic, commonsense, and symbolic reasoning datasets. The results indicate that our proposed method achieves the best or competitive performance compared to state-of-the-art methods, with an average score improvement of 7.4 percentage points. Additionally, we conduct ablation experiments to validate the effectiveness of different parts of our method and its performance across different LLMs.

In summary, our contributions include:

- We introduce a universally applicable and inherently more logical representation method for latent learningscape, named **latent learningscape feature**. This method overcomes the limitations of surface-level semantic features, circumventing the challenges of directly identifying latent learningscape. The latent learningscape feature represent differences between demonstrations and determine the inclusion relationship among them. This creates favorable conditions for constructing comprehensive prompts that cover a wide range of capabilities.

- We propose a framework to enhance LLM performance using latent learningscape feature. Our framework fully capitalizes on the excellent characteristics of latent learningscape features, constructing prompts that encompass a rich range of latent learningscape. This approach allows us to achieve superior results with lower energy consumption.

- We conducted extensive experiments to validate the effectiveness of our framework against various baseline methods and across different base large language models.

2 Related Work

2.1 Large Language Model

With the emergence of Transformer(Vaswani et al., 2023) and BERT (Devlin et al., 2019), significant progress has been made in many NLP tasks. Building upon these foundations, previous works (Brown et al., 2020; Chowdhery et al., 2022; Anil et al., 2023; Touvron et al., 2023) introduced large language models with a substantial number of parameters, showcasing powerful reasoning capabilities. As widely recognized, Kojima et al. (2023) discovered that adding simple sentences like "Let's think step by step" stimulated the emergent capability of LLM. Wang et al. (2023a) introduced a new statement, PS (based on the "plan-solve" problem-solving approach), significantly improving the model's performance across various datasets.

2.2 In-context Learning

Liu et al. (2021) proposed using sentence embedding distance for selecting demonstrations in LLM's In-context learning. Su et al. (2022) utilized sentence embedding distance to select a representative small dataset from a large training set to retrieve demonstrations. Zhang et al. (2022) introduces a classical method for finding representative demonstrations, enhancing few-shot Commonsense Reasoning. Building on this, Gonen et al. (2022) introduced perplexity in selecting demonstration examples, and Levy et al. (2023) emphasized the importance of diversity. Nguyen and Wong (2023); Li and Qiu (2023) proposed outcome-oriented approaches, measuring outcome differences and defining InfoScore for model prediction improvement. Rubin et al. (2022) build a supervised retriever EPR using LM scores, an effective endeavor to bypass surface-level semantic features. Building upon this, Li et al. (2023a) further optimized the EPR for adaptability across diverse datasets. In Ye et al. (2023) trained a DPP retriever to align with LM output scores through contrastive learning and obtained the optimal demonstration set during inference. Wang et al. (2023b) proposed an approach that combined a small Language Model (LM) with a large LM, using the small LM to select demonstrations for In-context Learning by the large model. Zhou et al. (2023) proposed a multi-stage prompt approach, breaking down complex problems into simpler ones. Ye and Durrett (2022) instructed the LLM to cross-check the reasoning process based on the facts presented in the prompt, enabling a more keen identification of potential errors in the answers. Li et al. (2023b) not only constructed various prompts but also trained a Step-Aware Verifier to validate the correctness of the reasoning proces. Zelikman et al. (2022) treated incorrectly answered questions equally, obtaining rationales by adding hints and improved the LLM's performance. Gao et al. (2023); Chen et al. (2023) introduced a code compiler, guiding the model for better few-shot performance, demanding more from annotators. Its effectiveness for commonsense reasoning tasks is unclear.

3 Method

Our approach consists of three steps, namely: **(1) Basic Pools Construction**: Establish the appropriate question pool and the challenging question pool. **(2) Latent Learningscape Feature Characterization**: Characterize questions' latent learningscape features based on their ability to correct challenging questions and questions with demonstration potential in the demonstration pool. **(3) Comprehensive Demonstration Set Construction**: Formation of linearly independent groups from the latent learningscape features in the demonstration pool, creating a comprehensive set of prompts for testing and conducting majority voting.

3.1 Basic Pools Construction

Construction of Easy and Challenging Question Pools We commence by randomly selecting questions from the training set and employing LLM for answer generation, as shown in Algorithm 1. If the correct answer can be obtained, it indicates that the difficulty of the question aligns with the model's capabilities, and should be added to the A-Pool (appropriate question pool). If the correct answer cannot be obtained, it suggests that the question poses a challenge for the model and should be added to the C-Pool (challenging question pool). The stopping criteria include reaching a specified threshold (m) for the number of questions in the A-Pool and another threshold (n) for the number of questions in the C-Pool. Here, (m) and (n) are hyperparameters, where (n) determines the length of the latent learningscape feature vector, and (m) influences the final quantity of prompts generated. Typically, (m) is set to 30, and (n) is set to 10.

3.2 Latent Learningscape Feature Characterization

Each question in the A-Pool (appropriate question pool), along with its reasoning process and answer, serves as a one-shot prompt demonstration, as shown in Algorithm 2. These demonstration are concatenated with each question in the C-Pool (challenging question pool), challenging the large model to correct errors. If a question from the A-Pool fails to correct any question in the C-Pool, it is considered lacking potential as a demonstration and should be eliminated. Conversely, if a question from the A-Pool successfully corrects one or more Challenging questions, it is deemed suitable for the D-pool (demonstration Pool). The latent learningscape features of these questions are represented using binary vectors of length n, where each position is either *0* or *1*. If a particular challenging question from the C-Pool can be corrected by a appropriate question from the A-Pool, then the position in the binary vector corresponding to that

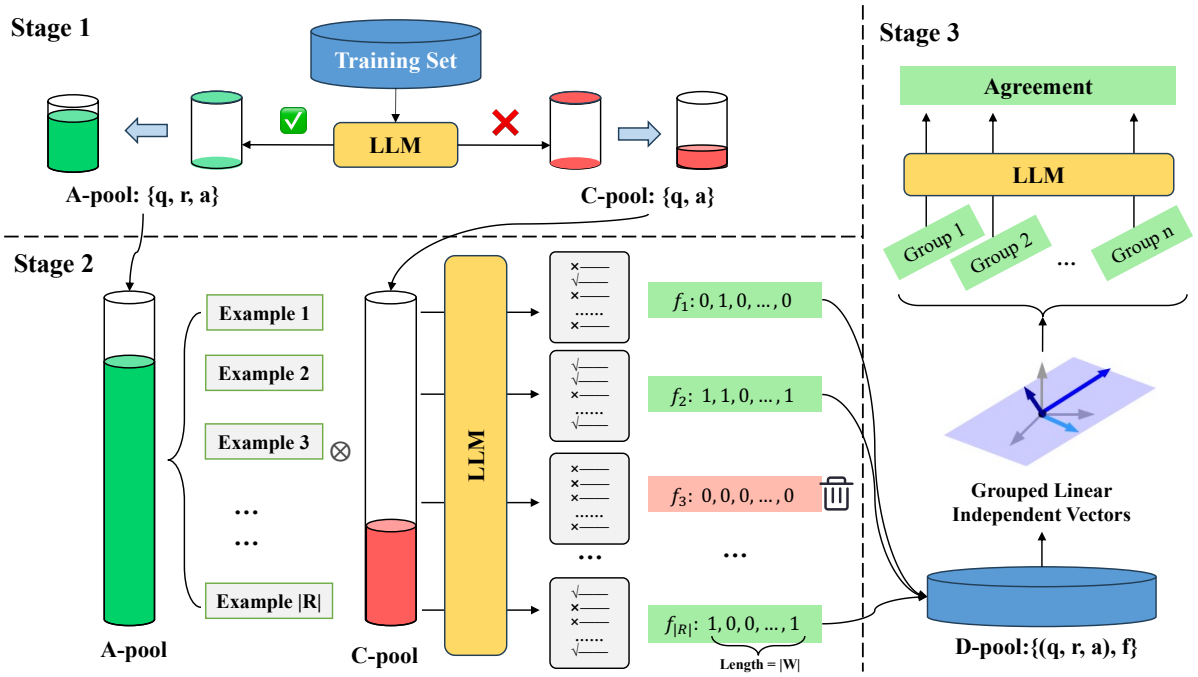

Figure 2: Stage 1: Basic Pools Construction.Random some questions from the training set and let the LLM perform reasoning. Those obtaining the correct answers are added to the A-Pool (appropriate question pool), while those that cannot are added to the C-Pool (challenging question pool). Stage 2: latent learningscape feature Characterization. Characterize the latent learningscape feature by utilizing the ability to correct questions in the C-Pool. Stage 3: Construction of Prompts. Constructing prompts based on linearly independent groups and ranking them according to the number of *1*s. $\otimes$ means pairing each example on its left with every challenging question on its right in a pairwise manner.

challenging question in the C-Pool is set to *1* in the latent learningscape feature vector of the appropriate question. If the question cannot be corrected, the corresponding position is set to *0*. Additionally, if a challenging question was ever corrected, it should be added to the A-Pool for further consideration, with the condition that they can correct any challenging question other than itself. Importantly, if a question in the C-Pool was never corrected, it suggests that the corresponding knowledge points, problem-solving approaches, concepts, and themes have not been covered in the A-Pool. Therefore, through labeled LLM question-answering to get reasoning process, it should be added to the D-pool with a latent learningscape feature vector containing a *1* at its corresponding position and *0* elsewhere, forming a vector of length n.

3.3 Construction of Prompts

Each entry in the D-pool (demonstration Pool), comprising a question, reasoning process, and correct answer, is referred to as a demonstration example. In order to make the most of these selectively filtered demonstration examples, two metrics are considered when choosing demonstration examples from the D-pool to construct a few-shot prompt: the number of times a demonstration example has been used (t) and its contribution (c). The usage count (t) indicates how many prompts have already utilized the current demonstration example, while the contribution (c) signifies the increase in rank of the vector group formed by its latent learningscape feature vector after using the current demonstration example. Priority is given to the usage count as the primary consideration, with lower usage counts taking precedence. In cases of equal usage counts, the contribution (c) is considered, with higher values receiving higher priority. With this algorithm, we ensure the thorough utilization of each demonstration example from the D-pool while obtaining several full-rank, linearly independent vector groups. These vector groups are then used to determine the set of demonstration examples constituting the few-shot prompt. At this point, we have obtained several sets of demonstration examples that meet our criteria. However, we may not actually need to use all of them, so it is necessary to rank them based on their quality. We rank them according

to the number of *1*s in the vector groups corresponding to each demonstration example set. A higher count of *1*s indicates better quality, and thus a higher ranking. We select the top-*k* demonstration example sets for concatenation with the test and determine the final answer through majority voting. Typically, k is set to 10.

Algorithm 1 Basic Pools Construction

Input: Training data set S_{train}= {q_i,a_i}
Output: Appropriate Question Pool A, Challenging Question Pool C

$m \leftarrow A$ threshold, $n \leftarrow C$ threshold
while $|A| < m$ or $|C| < n$ **do**;
  Select (q_i, a_i) from S_{train}
  $r_i \leftarrow$ Model response to q_i;
  if r_i matches a_i **then**
    Add (q_i, r_i, a_i) to A;
  else;
    Add (q_i, a_i) to C;
  end if;
end while;

Algorithm 2 Latent Learningscape Feature Characterization

Input: Appropriate Question Pool A, Challenging Question Pool C
Output: Demonstration Pool D

for each (q_i, r_i, a_i) in A **do**
  $f_i = [0] \times |C|$
  for each (q_j, a_j) in C **do**
    $p_{ij} \leftarrow concat(q_i, r_i, a_i, q_j)$
    $r_{ij} \leftarrow$Model response to p_{ij}
    if r_{ij} matches a_j **then**
      f_i[j] = 1
    end if
    if $CountOne(f_i) > 0$ **then**
      Add $((q_i, r_i, a_i), f_i)$ to D
    end if
  end for
end for

4 Main Experiments

We conducted experiments using the widely adopted large language model, ChatGPT(gpt-3.5-turbo-0613), across three distinct categories encompassing a total of nine datasets to validate the feasibility and efficacy of our proposed method. The first categories include five mathematical datasets: GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), MathQA (Amini et al., 2019) (first 400 questions), SVAMP (Patel et al., 2021), and Type 1 of NumGLUE (Mishra et al., 2022). The second category involves two commonsense reasoning datasets, namely CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). The third category comprises two symbolic reasoning datasets, specifically Last Letter (Wei et al., 2023) and Coin Flip (Wei et al., 2023).

4.1 Baseline

We employed three zero-shot CoT methods, namely, a method that directly answers without the addition of a trigger sentence,to show the **Basic** abilities of the model (The current model already has a certain level of CoT capability). A method proposed by (Wang et al., 2023a), and another proposed by (Kojima et al., 2023) and updated by (Zhang et al., 2022). They each appended the following statement to the end of the question:

PS+: *A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.*

One by one: *A: let's think not just step by step, but also one by one.*

Additionally, we utilized a few-shot CoT method **Auto-CoT** based on text similarity, proposed by (Zhang et al., 2022), and a method employing a **self-consistency** decoding strategy as a baseline, proposed by (Wang et al., 2023c).

The selection of three zero-shot CoT methods stemmed from an intriguing observation during experimentation. The performance variations among different zero-shot CoT methods across various datasets seemed to exceed our initial expectations. This observation prompted some contemplation, leading us to retain these three baselines. A detailed discussion on these baselines will follow in the subsequent sections of the experimental results analysis.

4.2 Main Results

The results of the main experiments are presented in Table 1. From the results, our method exhibits significant advantages over various baselines in mathematical analogy reasoning tasks and symbolic reasoning tasks. Furthermore, within these datasets, despite the increased complexity com-

Methods (based on GPT-3.5)	Arithmetic					Commonsense		Symbolic	
	NumGLUE	GSM8K	AQuA	MathQA	SVAMP	CSQA	Strategy	Letter	Coin
Basic	76.5	69.37	53.9	53.6	82.0	59.7	75.5	22.0	55.0
One by one	69.1	76.8	55.5	53.8	81.0	63.5	69.9	65.6	91.2
PS+	72.8	72.9	53.5	57.1	78.0	66.3	**77.2**	45.2	59.8
Auto-CoT	82.0	78.3	57.4	49.8	80.0	65.8	74.3	79.0	95.0
Self-Consistency	82.0	85.5	64.5	61.1	84.0	**77.1**	70.3	83.8	58.2
Ours	**83.3**	**88.0**	**69.6**	**63.3**	**88.0**	75.6	73.7	**92.0**	**100.0**

Table 1: Performance comparison of different methods based on gpt-3.5-turbo across various tasks. Evaluation metrics include NumGLUE (type 1) (Mishra et al., 2022), GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), MathQA (first 400) (Amini et al., 2019), SVAMP (Patel et al., 2021), ComensenseQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021), Last Letter (Wei et al., 2023), and Coin Flip (Wei et al., 2023). Results are reported as percentages. The highest scores for each task are highlighted in bold, and the second-highest scores are underlined. **Some incorrect questions in the **Coin Flip** dataset have been corrected.*

pared to baselines other than self-consistency, our method demonstrates a more pronounced superiority, substantially enhancing performance (average improvement of 11% across 7 datasets). In contrast to the self-consistency approach, our method achieves superior performance while operating at significantly lower energy consumption (approximately 0.25-0.5 times of its energy consumption). However, in two general datasets, our method does not replicate the performance observed in the other two datasets. Notably, the baseline with the highest energy consumption also appears less robust, presenting an intriguing phenomenon that will be further analyzed in the subsequent sections.

4.3 Analysis

Just like the diverse nature of the world, each dataset possesses its unique characteristics. Upon observing unexpected performances in the StrategyQA, and Coin Flip datasets across various methods, we conducted a thorough analysis of the datasets. In the StrategyQA dataset, the PS+ method outperforms both our method and the high-energy consumption Self-consistency method. Upon analyzing the data, we attribute this to the dataset's alignment with a problem-solving approach known as "plan-solve." For example, consider the question *"Hydrogen's atomic number squared exceeds the number of Spice Girls?"* in this dataset. The PS+ method tackles such problems by first macro-analyzing the entire question, formulating a solving plan based on the complete question, and then executing it. On the other hand, the "step-by-step" method finds an entry point, determines the next action direction based on the previous step. In comparison, the former aligns more closely with the problem-solving approach humans employ when facing such questions, while the latter demonstrates robust performance in the Coin Flip dataset. In this dataset, questions are structured similarly to *"A coin is heads up. Lucky does not flip the coin. Mireya flips the coin. Jj flips the coin. Kc flips the coin. Is the coin still heads up? Note that 'flip' here means 'reverse.'"* Clearly, the human problem-solving approach for such questions is step-by-step, strictly relying on the results of the previous step to determine the parameters for the next step. This explains why the "step-by-step" method achieves a score of 91.2 in this dataset. Moreover, both PS+ and "step-by-step" methods exhibit unexpected differences in two symbolic reasoning tasks, as analyzed above. These tasks are better suited for the "step-by-step" method, and the difficulties faced by the PS+ method in tasks it is not adept at can be comprehended based on this analysis.

We continue the discussion on the performance of the Auto-CoT method, and it is observable that this embedding similarity-based approach performs well on the majority of datasets, particularly excelling in the Coin Flip dataset. This is attributed to the inherent characteristics of the Coin Flip dataset, where clear distinctions between data categories exist, allowing for effective differentiation based on surface-level semantics. Consequently, the method can identify representative data for each category based on surface-level semantics, making it particularly suitable for such datasets.

In other words, datasets characterized by high uniformity in surface-level semantics are well-

suited for this method. However, it's important to note that this is essentially a case of overfitting. Datasets with such characteristics are not prevalent, nor is this a common occurrence in real-life scenarios. In contrast, our method is capable of transcending the limitations imposed by surface-level semantics, resulting in superior performance across a broader spectrum of datasets.

The self-consistency method demonstrates universality and often achieves commendable results. However, in comparison, our approach attains superior performance with lower energy consumption. This is attributed to our identification of genuinely representative examples, and by adhering to diversity rules, we assemble them into high-quality demonstration example combinations. In contrast to other baselines, this enables us to achieve better results through the amalgamation of diverse and representative examples, all while operating at a lower energy cost.

For our method, it can be observed that it did not achieve particularly favorable results on two commonsense reasoning datasets. This is because, compared to other datasets, commonsense datasets have more open-ended content and emphasize the model's memory of basic knowledge. In this scenario, the impact of In-context learning abilities on the results is no longer the primary factor. In other words, our method can identify high-quality demonstration example sets, thereby better stimulating the model's In-context learning abilities. Mathematical reasoning datasets and symbolic reasoning datasets heavily rely on the model's ability in this regard, explaining the excellent performance of our method on these datasets. Commonsense reasoning, with its open-ended and memory-focused nature, requires attention to the suitability of problem-solving approaches, rather than model's In-context learning ability (such as "Plan-Solve" for the StrategyQA dataset).

5 Ablation Experiment

5.1 Settings & Datasets

To further investigate the effectiveness of our method, we conducted ablation experiments from the following perspectives:

- We employed different models Gemini pro (Anil et al., 2023) and Llama-70B (Touvron et al., 2023), to validate the robustness of our approach. This aimed to observe whether our method could yield the expected results across diverse models.

Methods (based on Gemini)	**Arithmetic**		**Commonsense**	**Symbolic**
	NumGLUE	SVAMP	CSQA	Letter
PS+	67.9	75.0	36.3	54.5
Auto-CoT	75.6	79.0	58.0	31.2
Self-Consistency	83.3	92.0	57.4	**73.6**
Ours	**84.6**	**94.0**	**77.6**	67.0

Table 2: Ours methods on Gemini pro.

Methods (based on Llama)	**Arithmetic**		**Commonsense**	**Symbolic**
	NumGLUE	SVAMP	CSQA	Letter
PS+	55.5	60.0	33.7	33.0
Auto-CoT	65.4	65.0	57.3	34.0
Self-Consistency	66.7	81.0	65.3	34.5
Ours	**74.1**	**87.0**	**66.7**	**46.5**

Table 3: Ours methods on Llama-70B.

- We replaced the construction of linearly independent groups in Stage 3 with a method that randomly selects 10 sets of 3 examples from demonstration pool to verify the effectiveness of Stage 3. Additionally, we used a method of randomly selecting 10 sets of 3 examples from the training set for comparison to validate the overall effectiveness of our method.

- In Stage 3, we sorted the vector groups based on the number of *1*s in each vector to examine the influence of the quantity of vector groups participating in the final vote. We conducted experiments with different values of k in the top-k selection, including k=3, 5, 10 and 20.

- Regarding datasets, to comprehensively evaluate the method's performance while considering computational costs, we selected four datasets from all three categories, encompassing two arithmetic reasoning datasets, one commonsense reasoning dataset, and one symbolic reasoning dataset for experimentation.

5.2 Analysis

Table 2 and 3 demonstrate that our method consistently achieves outstanding performance when used with different models. However, the results indicate that when the performance of the auto-CoT method declines, our method also experiences a decrease in effectiveness. We posit that the auto-CoT method relies on the model's In-context learning ability, and its performance is inhibited when this capability is insufficient. While our method is a superior approach to finding demonstration examples, it is evident that it also relies heavily on the model's In-context learning ability. Relatively speaking, the self-consistency method exhibits less sensitivity to

the capabilities of In-context learning, as its design is not particularly dependent on this learning ability. Being inherently a decoding strategy, this method is less influenced by the impact of In-context learning abilities on its performance.

It is evident in table 4 that the performance of the randomly selected 10 3-shot demonstration example sets from the D-pool (demonstration pool) remains significantly higher than that of the randomly selected 10 3-shot demonstration example sets from the training set. However, the optimal performance is still achieved by our complete method. This indicates that the D-pool we have selected indeed captures potential demonstrations, resulting in more stable and superior performance even when randomly chosen from it compared to randomly selecting from the training set. Similarly, our method of linearly independent vector groups further solidifies this advantage, making the performance more stable and outstanding.

In Table 5, it can be observed that although the performance did not reach the optimal level when $k = 10$ in top-k, there is a significant improvement compared to $k = 5$ (average of 3.38%). In comparison to $k = 20$, the performance loss of this method is relatively slight (average of -1%). We believe that even though our method is an approach that is closer to the essence of things relative to surface semantics, it is not necessarily the only final correct answer. Specifically, the latent learningscape feature vectors of certain questions may be very similar or even identical, but this does not imply complete consistency in the knowledge points, problem-solving approaches, concepts, themes behind those questions. Therefore, we need a majority voting process to balance this potential bias. Experimental results indicate that too few votes still pose a risk of exposing the aforementioned problem, but the risk significantly decreases when the number of votes is 10. With 20 votes, the improvement trend starts to decline. Considering comprehensive performance and energy consumption, we choose to involve the top 10 in the voting process.

6 Conclusion

In this paper, we introduced a latent learningscape that is more closely aligned with the essence of In-context learning abilities, representing knowledge points, problem-solving approaches, concepts, and themes that truly enable effective In-context learning. Building upon this foundation, we proposed a result-driven three-stage method. Firstly, we constructed latent learningscape features for demonstration example with the potential to serve as demonstrations. Next, we generated several demonstration example sets covering a comprehensive latent learningscape by constructing linearly independent vector groups, which were used as demonstrations in the few-shot prompt. Experimental results demonstrate that our method indeed enhances the model's In-context learning abilities, achieving significant performance improvements without a substantial increase in energy consumption. Moreover, compared to the highest energy-consuming methods, our approach not only outperforms them but also achieves better results. Ablation experiments confirm the effectiveness of our method across different models, indicating that we have genuinely identified examples with demonstration potential and, by defining their latent learningscape features, combined them into more valuable compositions.

Methods (based on GPT-3.5)	**Arithmetic**		**Commonsense**	**Symbolic**
	NumGLUE	SVAMP	CSQA	Letter
Ours	**87.1**	**94.0**	**76.3**	**93.0**
w/o stage 3	84.6	92.0	**76.3**	92.0
w/o stage 2 & 3	82.7	88.0	75.5	81.6

Table 4: Comparison of ours method with our method without stage 3 Construction of Prompts and our method without stage 2 latent learningscape feature Characterization & 3 Construction of Prompts. **w/o stage 3**: random select demonstrations from demonstration pool; **w/o stage 2 & 3**: random select demonstrations from training set;

top-*k*	**Arithmetic**		**Commonsense**	**Symbolic**
	NumGLUE	SVAMP	CSQA	Letter
k=3	**87.1**	89.0	69.3	82.7
k=5	78.2	92.0	73.0	89.2
k=10	**87.1**	94.0	76.3	93.0
k=20	**87.1**	**96.0**	**77.6**	**93.7**

Table 5: Ours methods with different top-*k* in stage 3.

7 Limitation

There are two limitations to this work. First, the performance of our method is inherently tied to the capabilities of LLMs. Due to the reliance of our method on the ICL capabilities of the model, its performance on small or middle language models may not be optimal. Further research is needed to explore the application methods of our approach

on small or middle language models. Second, although our method aims to achieve superior results with lower energy consumption, the energy efficiency aspect is not explicitly quantified in comparison to other methods. A more comprehensive energy analysis is needed to provide a holistic understanding of the trade-offs involved.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

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.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

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.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023a. Unified demonstration retriever for in-context learning.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making large language models better reasoners with step-aware verifier.

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. 2021. What makes good in-context examples for gpt-3?

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523, Dublin, Ireland. Association for Computational Linguistics.

Tai Nguyen and Eric Wong. 2023. In-context example selection with influences.

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.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Selective annotation makes language models better few-shot learners.

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.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023b. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models.