Complementary Explanations for Effective In-Context Learning

Xi Ye[♢*] **Srinivasan Iyer**[♠] **Asli Celikyilmaz**[♠] **Ves Stoyanov**[♠]
Greg Durrett[♢] **Ramakanth Pasunuru**[♠]
[♢] The University of Texas at Austin [♠] Meta AI
[♢]{xiye,gdurrett}@cs.utexas.edu
[♠]{sviyer,ves,aslic,rpasunuru}@meta.com

Abstract

Large language models (LLMs) have exhibited remarkable capabilities in learning from explanations in prompts, but there has been limited understanding of exactly how these explanations function or why they are effective. This work aims to better understand the mechanisms by which explanations are used for in-context learning. We first study the impact of two different factors on the performance of prompts with explanations: the computation trace (the way the solution is decomposed) and the natural language used to express the prompt. By perturbing explanations on three controlled tasks, we show that both factors contribute to the effectiveness of explanations. We further study how to form maximally effective sets of explanations for solving a given test query. We find that LLMs can benefit from the *complementarity* of the explanation set: diverse reasoning skills shown by different exemplars can lead to better performance. Therefore, we propose a maximal marginal relevance-based exemplar selection approach for constructing exemplar sets that are both relevant as well as complementary, which successfully improves the in-context learning performance across three real-world tasks on multiple LLMs.

1 Introduction

Large language models (LLMs) have achieved promising progress in learning from only a few exemplars in prompts via in-context learning (ICL) (Brown et al., 2020; Chowdhery et al., 2022). To scale to complex tasks, recent work in the past year has shown that LLMs can benefit from explanations in prompts, particularly for tasks involving multi-step reasoning (Nye et al., 2021; Wei et al., 2022; Wang et al., 2022b; Madaan et al., 2022; Jung et al., 2022). However, while including explanations in prompts has been demonstrated to be useful, little has been shown regarding what particular features make them effective and how they function in ICL.

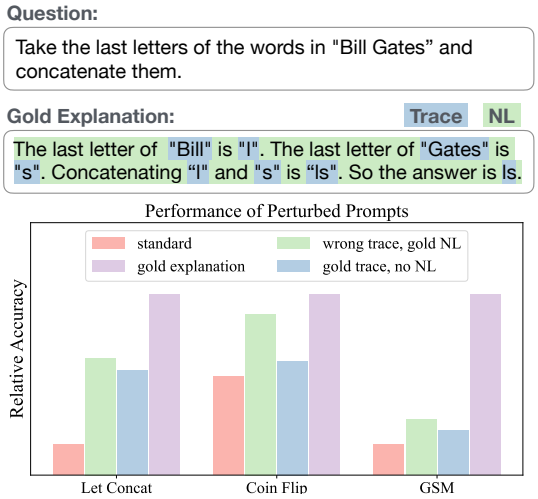

Figure 1: Prompting OPT (Zhang et al., 2022) with explanations where we perturb the computation traces or natural language. Perturbing either traces or natural language will lead to performance degradation.

Our work aims to better understand the mechanisms by which explanations are used for ICL. As shown in the example for a multi-step reasoning task in Figure 1, we view an explanation as a combination of a computation trace and natural language which glues together the states in the trace. We design a series of probing experiments that perturb the explanations (as shown in Figure 2) and test LLMs' performance to understand the sensitivity of LLMs on these two factors. The results suggest that both factors contribute to making effective explanations, as LLMs see substantial performance degradation when prompted with defective explanations. Nonetheless, incomplete explanations are still beneficial compared to no explanations at all (Figure 1). This suggests that LLMs "faithfully" follow the reasoning process specified by the explanations to some extent, as opposed to naively following the template patterns while disregarding critical information (Min et al., 2022c).

The observations from our probing experiments lead us to focus next on understanding what makes

[*] Work done during an internship at Meta AI.

an effective *set* of exemplars with explanations for solving a given test query. We primarily focus on two aspects, the exemplar-exemplar interplay (how exemplars work together) and the query-exemplar interplay. On the former, we find that the *complementarity* of the exemplar set is beneficial, as LLMs can fuse different reasoning processes exhibited by individual exemplars in-context. For these studies, we probe LLMs with a mixture of two types of exemplars; each type only specifies a part of the reasoning process (see Figure 3 for detailed instances). We also test how the relevance between the query and the exemplars impacts the performance. Choosing the nearest neighbors (NN) of the query to prompt LLMs has been shown to be effective in the standard prompting setting (Liu et al., 2021). Our experiments covering three similarity metrics show that this is also applicable in the setting that prompts LLMs with explanations.

Our analyses inspire us to rethink the exemplar selection process of using explanations for ICL. The prominent NN-based paradigm only considers relevance (Shin et al., 2021; Liu et al., 2021; Rubin et al., 2022), which could result in selecting mostly similar exemplars. We argue that complementarity should also be considered when constructing explanation-infused prompts. Therefore, we propose an exemplar selection strategy based on the maximum marginal relevance (MMR) (Carbonell and Goldstein, 1998) approach which selects exemplars that are both relevant as well as *diverse*. The underlying rationale is that a diverse set of exemplars is more likely to showcase complementary reasoning types that are required to illustrate the reasoning required in the query. We test our MMR-based strategy on three real-world datasets spanning multiple reasoning tasks. On a powerful LLM, `text-davinci-002`, our MMR-based strategy is able to improve the accuracy over the baseline of using random exemplars by 4.0%, 3.9%, and 8.5% on GSM, ECQA, and E-SNLI, respectively.

In summary, our main findings are: (1) We show that both the computation trace and natural language contribute to making effective explanations for ICL. (2) We show that LLMs can benefit from exemplar sets that exhibit both complementarity and relevance to a given test query. (3) We propose an MMR-based exemplar selection strategy considering both complementarity and relevance and demonstrate that it is more effective than solely choosing the nearest neighbors.[1]

2 Background

In-Context Learning Our study is focused on the usage of explanations in in-context learning (ICL). Let q be the test query to solve. The standard ICL prompts a language model, M, with a set of exemplar input-output pairs, $\{(q_1, a_1)...(q_m, a_m)\}$, and predict an answer $\hat{a}$ for the query:

$$\hat{a} = \arg\max_a p_M(a \mid q, \{(q_1, a_1)...(q_m, a_m)\}).$$

In addition to just input-output pairs, we can also include explanations (in the style of Scratchpad (Nye et al., 2021) or chain-of-thought (Wei et al., 2022)) in prompts, which leads the LLM to generate explanations for its predictions as well:

$$\hat{a} = \arg\max_a \sum_e p_M(a, e \mid q, C),$$

where $C = \{(q_1, e_1, a_1)...(q_m, e_m, a_m)\}$ is the set of input-explanation-output triplets in prompts. Ideally, inference in this requires marginalizing out the explanation e, which is impractical, especially with LLMs. Following Wei et al. (2022); Ye and Durrett (2022), we employ greedy decoding to make an approximate prediction during inference.

The end task performance of ICL is sensitive to the selected exemplars (Liu et al., 2021). While much prior work uses a fixed set of manually selected exemplars (Wei et al., 2022; Wang et al., 2022b), there is also work devoted to studying how to select more effective exemplars from a pool of exemplars. Given a test query q, the task is to select a set of m exemplars from a pool of n exemplars $D = \{(q_1, e_1, a_1)...(q_n, e_n, a_n)\}$ to construct a prompt for solving q. We note that this yields varying exemplar sets for different queries.

Datasets & Large Language Models Our analysis is based on model performance on various reasoning datasets. For probing experiments, we mainly use symbolic reasoning datasets, including 1) LETTER CONCATENATION (LET CAT) (Wei et al., 2022) which requires extracting the last letters of two words and then concatenating them, 2) COIN FLIPS (Wei et al., 2022) which reasons about the states of a coin after two steps of operations (flipping or not flipping), and 3) GRADE SCHOOL

[1]Code is available at https://github.com/xiye17/ComplementaryExpl.

MATH (GSM) (Cobbe et al., 2021) which focuses on grade school-level arithmetic reasoning problems expressed in natural language.

To investigate the effectiveness of different exemplar selection strategies, we use two more textual reasoning datasets, namely ECQA (Aggarwal et al., 2021) and E-SNLI (Camburu et al., 2018), in addition to GSM. The task of E-SNLI is to decide whether a premise entails a hypothesis. ECQA asks multiple-choice commonsense questions. These three tasks include human-annotated explanations and cover diverse reasoning abilities.

Our experiments cover an array of LLMs, including `OPT-175B` (Zhang et al., 2022), `GPT-3 (davinci)` (Brown et al., 2020), `InstructGPT (text-davinci-001)` (Ouyang et al., 2022), and `text-davinci-002`. In addition, we also use GPT-3 Codex models (Chen et al., 2021) that are finetuned on a large scale of code snippets in our exemplar selection experiments, namely `code-davinci-001` and `code-davinci-002`. Though codex models primarily target code-related applications, we find that they are strong in textual reasoning tasks as well.

3 Do LLMs Follow Explanations?

We first investigate what makes explanations effective for LLMs to learn from. We view an explanation as a computation trace (T) that is transformed by a natural language function (L) which maps the trace to a complete natural language explanation. A computation trace T is the chain of intermediate steps (instantiated as tokens in explanations), $s_1, \ldots, s_n$, that are used to derive the final answer. For instance, the trace for LETCAT is the two last letters of the two words and the concatenated two-letter tokens; the traces for GSM are the intermediate equations. These computation traces are wrapped by natural language function L to form the final explanation $L(s_1, \ldots, s_n)$, which presumably makes the generation of these traces more "natural" with respect to language modeling.

Setup We choose the three symbolic reasoning datasets mentioned in Section 2 for our probing experiments for two reasons. First, LLMs see substantial benefits from including explanations in prompts for these tasks. Second, we can easily manipulate the traces in their explanations. The gold explanations are directly taken from or adapted from Wei et al. (2022). More details on the gold explanations used for these tasks can be found in Appendix A.

	Question: Take the last letters of the words in "Bill Gates" and concatenate them.
LETCAT	**Gold:** The last letter of Bill is l. The last letter of Gates is s. Concatenating l and s is ls. So the answer is ls. **Mask1:** The last letter of Bill is _. The last letter of Gates is _. Concatenating l and s is ls. So the answer is ls. **Mask2:** The last letter of Bill is l. The last letter of Gates is s. Concatenating _ and _ is _. So the answer is ls. **Incorrect:** The last letter of Bill is y. The last letter of Gates is e. Concatenating y and e is ye. So the answer is ye. **No NL:** Bill, l. Gates, s. l, s, ls. So the answer is ls.
	Question: A coin is heads up. Ka does not flip the coin. Sal flips the coin. Is the coin still heads up?
COINFLIP	**Gold:** The coin started heads up. Ka does not flip the coin, so it becomes heads up. Sal flips the coin, so it becomes tails up. So the answer is no. **Mask1:** The coin started heads up. Ka does not flip the coin, so it becomes _ up. Sal flips the coin, so it becomes tails up. So the answer is no. **Mask2:** The coin started heads up. Ka does not flip the coin, so it becomes heads up. Sal flips the coin, so it becomes _ up. So the answer is no. **Incorrect:** The coin started heads up. Ka does not flip the coin, so it becomes tails up. Sal flips the coin, so it becomes heads up. So the answer is yes. **No NL:** heads, heads, tails. So the answer is no.
	Question: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
GSM	**Gold:** Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32+42=74 chocolates. 35 have been eaten. So in total they still have 74-35=39 chocolates. The answer is 39. **Mask1:** Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32+42=_ chocolates. 35 have been eaten. So in total they have _-35=39 chocolates. The answer is 39. **Mask2:** Leah had 32 chocolates and Leah's sister had 42. That means there were originally _ chocolates. 35 have been eaten. So in total they have _ chocolates. The answer is 39. **Incorrect:** Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32+42=62 chocolates. 35 have been eaten. So in total they have 62-35=27 chocolates. The answer is 27. **No NL:** 32+42=74, 74-35=39. The answer is 39.

Figure 2: Examples of gold explanations and perturbed explanations. We perturb the trace in gold explanations (colored) by masking intermediate states or substituting them with incorrect values.

We experiment on both LLMs trained with vanilla language modeling objectives (`OPT`, `davinci`) and LLMs that are aligned with human expectations via different forms of instruction tuning (`text-davinci-001`, `text-davinci-002`).

3.1 Explanations or Triggers?

We start by investigating whether actual computation traces matter. If the correctness of the in-context demonstration is unimportant, then that serves as evidence that explanations act as triggers that induce LLMs to follow certain patterns and perform slot-filling. To study this, we prompt LLMs with perturbed explanations, i.e., by perturbing computational traces and measuring the impact on performance in an ICL setting.

Figure 2 shows concrete examples of how we perturb the gold explanations. We experiment with two ways of perturbing the inputs. The first way is

	LETCAT				COINFLIP				GSM			
	OPT	davinci	txt-01	txt-02	OPT	davinci	txt-01	txt-02	OPT	davinci	txt-01	txt-02
Standard	8.5	8.5	10.5	16.0	51.5	83.0	68.0	99.0	5.5	7.5	11.0	26.5
Gold	50.0	59.0	85.0	100.	94.0	89.5	100.	100.	32.5	26.0	25.0	57.5
Mask1	11.0	16.0	21.5	100.	71.0	88.0	61.5	100.	19.0	21.0	12.5	29.5
Mask2	32.5	49.5	68.0	100.	84.0	91.5	99.0	100.	10.0	16.0	11.5	27.5
Random	10.0	25.0	28.0	13.0	52.5	54.5	67.0	69.0	3.0	3.0	1.0	34.5
Incorrect	40.0	53.0	67.5	99.5	60.5	86.0	52.0	100.	18.5	17.0	10.0	16.5
No NL	29.0	15.0	46.5	100.	59.5	86.0	99.0	100.	8.0	19.5	14.5	45.5

Table 1: In-context performance obtained using various perturbed explanations on three datasets. Perturbed explanations achieve inferior performance than complete ones, but many of the perturbed explanations still grant performance gains over standard prompting.

masking out the intermediate states by replacing a state s_i (or several states) in $s_1, \ldots, s_n$ with a mask token (e.g., empty string), which tests whether LLMs can implicitly infer the intermediate states. The second way is to replace a state s_i with an incorrect one, which tests whether LLMs can recover the correct computation from corrupted traces.

Construction of Perturbed Explanations We carefully design the way to mask out the intermediate states. We experiment with various choices of masks in our preliminary experiments. We do not observe large variance caused by different masks, and we choose to use empty string which leads to less performance degradation in general, as our goal here is to probe LLMs' maximum capabilities in recovering the reasoning process from perturbed explanations. More details on the preliminary experiments on choosing masks can be found in Appendix B. When constructing incorrect explanations, we also experiment with different sets of random values used to substitute the correct ones. Furthermore, we also include complete random explanations (taken from other exemplars), which replace the whole gold traces with incorrect ones.

Results Table 1 shows the results obtained using prompts with various perturbations on the three tasks. First, *LMs are indeed relying on the actual computation traces*. Perturbing the traces of the explanations will lead to performance degradation in various degrees on all these three tasks for `OPT`, `davinci`, and `text-davinci-001`. `text-davinci-002` does not exhibit performance degradation when being prompted with incomplete explanations on the simple tasks, LETCAT and COINFLIP: when the trace is straightforward, a powerful enough LLM is able to "shortcut" some particular steps. While for the more challenging tasks, i.e., GSM, `text-002` is also affected by perturbations in explanations. Nonetheless, *LLMs can still benefit from partially complete or partially correct explanations* and outperform standard prompting without using explanations. In particular, on the LETCAT task, even completely irrelevant random explanations can be beneficial, although they lag gold explanations. Overall, incorrect and incomplete or even totally irrelevant explanations are able to elicit reasoning, but LLMs do rely on gold explanations to work well.

3.2 Is Natural Language Necessary?

Next, we question whether the natural language (NL) is really necessary and test whether LLMs can infer the reasoning steps from the computation traces alone. We perturb gold explanations by not wrapping computation traces with natural language transformation L, as shown in the examples from Figure 2, and only retain the traces.

Results We show the performance obtained by using these prompts in Table 1. *Natural language also plays an essential role in making effective explanations*. Removing the NL leads to substantially worse performance. On LETCAT, the accuracy of `OPT`, and `davinci` drops by more than 20, respectively, compared to using gold explanations. On GSM, removing NL consistently leads to performance degradation. Meanwhile, including intermediate states without NL can still improve the performance compared to not using any explanations.

3.3 Discussion

As suggested in the experimental results in Section 3.1 and Section 3.2, LLMs do generally follow the explanations in the prompts. Both concrete computation traces and natural language contribute to making effective explanations for ICL. Perturbing certain parts of the explanations will accordingly result in performance degradation, but partial explanations are still beneficial to LMs.

By contrast, recent work shows LLMs are not sensitive to perturbations on the ground-truth input-label mapping in the standard prompting paradigm that does not use explanations (Min et al., 2022c). Our work shows that LLMs are sensitive to perturbations in the input-explanation mapping and other more subtle perturbations in the explanations. Using explanations in prompts is a promising way to guide LLMs in learning a new task via ICL.

4 What Makes A Good Exemplar Set?

Our probing experiments have established how the general factors, computation trace and natural language, impact explanations' effectiveness in ICL. We now study how a set of exemplars, as a whole, functions together in solving a particular test query. We study this problem from two angles, the interplay between exemplars and the interplay between the query and the exemplars.

4.1 Exemplar-Exemplar Interplay

As in Section 3, LLMs can learn to follow the reasoning processes as specified in exemplars. As reasoning processes can be composed, we hypothesize that LLMs might also be able to fuse the reasoning processes of different exemplars together to solve a test query. We design a set of probing experiments that successfully verify this hypothesis.

Experiment Design At the abstract level, we compare the performance of LLMs when being prompted with three sets of exemplars. The first and second set of exemplars each focuses on a particular part of the reasoning process, and these two parts are disjoint. That is, for a computation trace $s_1, \ldots, s_n$, the first and second set contain exemplars where s_i and s_j are perturbed, and $i \neq j$. The third set of exemplars includes the mixture from the first and second sets. We test the ICL performance of the prompts constructed from these three types of exemplar sets on the test set that requires combining two types of reasoning. If the third type gives superior performance than the first two types, that means LLMs can pick up the disjoint reasoning and fuse them in-context.

To better illustrate such a hypothesis, we give a concrete example as follows. We have introduced two different types of masked explanations in Figure 2 for LETCAT, where the first type masks the last letter extraction part, and the second type masks the letter concatenation part. These two different masked explanations specify two steps of the reasoning process needed; combining these two steps will yield the complete reasoning steps needed for solving this task. We test whether LLMs can combine these two reasoning steps in-context if being prompted with a mixture of these two corresponding types of masked prompts.

Add Only	**Q:** Marion received 20 more turtles than Mia at the animal rescue center. If Mia received 40 turtles, how many turtles did they receive together? **A:** Since Marion received 20 more turtles than Mia, she had 20 + 40 = 60 turtles. The two received 60 + 40 = 100 turtles. The answer is 100.
Mul Only	**Q:** Super Clean Car Wash Company cleans 80 cars per day. They make $5 per car washed. How much money will they make in 5 days? **A:** Each day they will make 80 * $5 = $400. They will make $400 * 5 = $2000 in 5 days. The answer is 2000.
Add & Mul	**Q:** Peter purchased 20 popsicles at $0.25 each. He also purchased 4 ice cream bars at $0.50 each. How much did he pay in total in dollars? **A:** The popsicles cost 0.25 * 20 = 5 dollars. The ice cream bars cost 0.5 * 4 = 2 dollars. He paid 5 + 2 = 7 dollars. The answer is 7.

Figure 3: Examples of GSM data points that involve only addition operators, only multiplication operators, and both of them at the same time. We note that Add & Mul are only used at the test time.

For GSM, we use a more organic way to partition the reasoning process. We separate the reasoning skills needed for a test query based on the operators (addition and multiplication) that are used in the steps. Concretely, we filter the GSM dataset by looking at the provided explanations paired with examples, and obtain disjoint sets that 1) only involves addition operators in the explanation 2) only involves multiplication operators in the explanation (See Figure 3 for examples). Next, we test the performance on a test set consisting of examples that require both operators at the same time (Add and Mul in Figure 3). This forms a test-bed for investigating whether LLMs can better learn to solve problems where both operators are present at the same time while being prompted with the mixture of these two operators, even if no explicit combinations are shown in the prompts.

Setup We experiment on the same three datasets as used in Section 3. On LETCAT and COINFLIP, we test whether LLMs can combine the reasoning steps specified in two different types of masked explanations; on GSM, we test whether LLMs can compose addition and multiplication. The mixture type prompts include half of the exemplars from the first type and second type which bear different reasoning. For each setting of GSM, we experiment with 4 different sets of randomly drawn exemplars and report the average. More details about the setting can be found in Appendix C.

		OPT	davinci	txt-01	txt-02
LETCAT	Mask1	11.0	16.0	21.5	100.
	Mask2	32.5	49.5	68.0	100.
	Mixture	**37.0**	**56.5**	**82.0**	100.
COIN	Mask1	71.0	88.0	61.5	100.
	Mask2	84.0	91.5	99.0	100.
	Mixture	**93.5**	91.0	100.	100.
GSM	AddOnly	6.8	13.5	14.1	50.3
	MulOnly	4.7	17.2	16.7	50.1
	Mixture	**7.0**	**18.9**	**18.2**	**52.0**

Table 2: The accuracy of prompting LLMs with exemplars focusing on single parts of the reasoning or a mixture of them. LLMs achieve better performance when being prompted with exemplars covering multiple aspects of the reasoning process.

Results As in Table 2, on LETCAT, the prompts with mixed explanations largely surpass Mask1, and outperform Mask2 by 6.0, 7.0, and 12.0 on `OPT`, `davinci`, `text-davinci-001`, respectively. Particularly, the mixture prompts is able to perform roughly on par with the complete prompt (Gold-Expl in Table 1) for LETCAT on `davinci`, and `text-001`. On COINFLIP, using mixture prompts also leads to improvements on `OPT`.

On the realistic GSM dataset, prompting `text-davinci-002` with only addition or multiplication exemplars leads to a performance of 50.3 and 50.1, respectively, whereas prompting with a mixture of these two types of exemplars achieves a better performance of 52.0. On `davinci`, and `text-001`, addition exemplars give worse performance than multiplication exemplars. Nevertheless, including these inferior addition exemplars in prompts together with multiplication exemplars still leads to better performance, as they can complement the reasoning. In general, results on three datasets suggest LLMs are able to fuse the reasoning process that is spread over different exemplars. Therefore, we can expect the exemplars to be able to complement each other and collaborate together to solve the reasoning needed in the test query.

4.2 Query-Exemplar Interplay

Next, we explore how the interplay between the query and exemplars impacts the ICL performance. Recent work has studied how to make good in-context exemplar sets for a given query in the standard prompting setting: choosing nearest neighbors that are more similar to the query leads to better performance (Liu et al., 2021; Shin et al., 2021). Our work investigates how choosing relevant exemplars impact the performance in the setting when using explanations in prompts. We compare the performance obtained by constructing prompts using nearest neighbors against using randomly selected exemplars. The results verify that choosing the nearest neighbors is also beneficial in this setting.

Similarity Measurements We test three different ways to measure the similarity $\mathcal{S}(q, q_i)$ between a test query q and an exemplar q_i.[2] **1) CLS-based:** Liu et al. (2021) use smaller LMs (e.g., BERT (Devlin et al., 2019)) to extract the CLS embedding of the input q and q_i and then use cosine similarity to score the embedding pairs, i.e., $\cos(\text{CLS}(q), \text{CLS}(q_i))$. **2) LM-based:** the similarity is given as the probability of generating the query when the language model is conditioned on the exemplar, i.e., $LM(q \mid q_i)$ (Shin et al., 2021; Rubin et al., 2022). **3) BERTScore:** we also experiment with using BERTScore (Zhang et al., 2020) as the similarity score, in addition to the two approaches that are commonly used in prior work.

It is worthwhile to note that measuring similarity using large LLMs is expensive. As it requires querying a large number of query-exemplar pairs.[3]

Setup We experiment on three realistic datasets, GSM, ECQA, and E-SNLI. leaving out synthetic tasks which feature formulaic explanations that are all similar to each other. We set the number of exemplars to be 8 for all three test datasets. We compare the performance of selecting nearest exemplars against that of selecting random exemplars.

Given the intensive cost of querying LLMs, we set the train exemplars pool size to be 512, and allocate computational resources to experimenting over 4 sets of randomly selected 512 exemplar pools to alleviate the influence of randomness. We focus on more capable LLMs, including `code-davinci-001`, and `code-davinci-002`, and `text-davinci-002`, leaving out `OPT` and `davinci` which have inferior performance. We note that we do not use `text-002` to measure similarity, owing to its high cost. Rather, we take the similarity scores computed by `code-002` and use those for `text-002`. So the performance of `text-002` when

[2]The similarity is measured only based on the input part and excludes the explanations part, as we do not have access to the explanation of the query in the test phase.

[3]For instance, calculating the similarity between 500 queries and a pool of 500 exemplars for a dataset whose typical question token number is 50, would cost $500 using GPT-3 API (rate: $0.02/1000 tokens).

	code-davinci-001				code-davinci-002				text-davinci-002			
	GSM	ECQA	E-SNLI	AVG	GSM	ECQA	E-SNLI	AVG	GSM	ECQA	E-SNLI	AVG
Random	16.3	53.6	47.2	39.0	64.6	74.7	74.9	71.3	48.8	71.9	75.1	65.3
CLS	16.5	55.0	54.1	41.8	65.4	74.9	74.8	71.7	50.4	72.1	77.4	66.6
LLM	**18.5**	**56.0**	**57.4**	**43.9**	65.8	**76.8**	**81.6**	**74.7**	**52.0***	**74.3***	**83.9***	**70.0**
BERTScore	**18.5**	54.6	53.7	42.3	**66.7**	75.9	75.6	72.8	51.0	72.8	78.7	67.6

Table 3: Comparison between the performance obtained by choosing relevant exemplars using CLS embedding, LM, or BERTScore. AVG denotes the average across the three datasets. Selecting relevant exemplars leads to performance improvements, especially when using LLMs themselves to measure the similarity. Using Bertscore also consistently improves the performance across all tasks, even surpassing LM-based scores on GSM. We note that the results on `text-davinci-002` use the LM-based scores provided by `code-davinci-002` (denoted by *).

using the LM-based measure might be suboptimal given the discrepancy.

Results As shown in Table 3, choosing relevant exemplars is also useful in the setting that includes explanations in prompts. Using the LM-based similarity measurements brings performance improvements across all three datasets, and has the most significant impacts on E-SNLI, though this is achieved with non-negligible computation cost. Using CLS-embeddings for selecting exemplars mildly improves the performance on GSM but does not result in any performance gains on ECQA. The limited improvements can be attributed to the size of the exemplar pools that we use. In our experiments, the size is 512, which is significantly smaller than that in Liu et al. (2021) (typically tens of thousands of exemplars). Nevertheless, this size is large enough for the LM-based method and BERTScore to take advantage of.

In addition, the results suggest that choosing relevant exemplars using BERTScore is also able to improve the performance across all datasets. Specifically, BERTScore-based exemplar selection achieves an accuracy of 66.7 on GSM using `code-002`, which even surpasses the performance of LM-based exemplar selection. While using BERTScore lags the LM-based on ECQA and E-SNLI, it still outperforms choosing random exemplars or CLS-based exemplar selection. Overall, using BERTScore to select the closest exemplars can lead to credible performance improvements while does not require heavy overheads caused by using LLMs to score query-exemplar pairs.

5 MMR for Exemplar Selection

We have established that emplar-exemplar interplay together with the query-exemplar interplay impacts the performance of using explanations in ICL. This leads us to rethink how to select good exemplars for a given query. Based on our prior analysis on the effects of complementarity and relevance in Section 4.1, we argue that a good set should consist of *relevant* exemplars that collaboratively cover the reasoning skills required for solving the query.

Algorithm 1 MMR-Based Exemplar Selection

procedure MMRSELECT($D, q, k, \mathcal{S}$)
input: exemplar pool $D = \{q_1 ... q_n\}$, test query q, number of shots m and similarity measurement $\mathcal{S}$
output: selected exemplars $T = \{q_1 ... q_m\}$
  $\mathbb{S} := [[\mathcal{S}(q_i, q_j)]]_{q_i, q_j \in D}$; ▷ the pairwise similarity between exemplars in D
  $\mathbb{Q} := [\mathcal{S}(q, q_i)]_{q_i \in D}$; ▷ the similarity between query and exemplars in T
  $T := \{\}$;
  while $|T| < k$ **do**
    $\hat{q} :=$ Equation(1); ▷ get the next exemplar based on Eq (1)
    T.add($\hat{q}$)
  return T;

The prominent paradigm, i.e., NN-based exemplar selection strategy, only considers the relevance between the exemplars and the query. Yet, selecting nearest neighbors could result in mostly similar exemplar sets, which can possibly limit collaboration. We argue that complementarity should also be considered in the exemplar selection process, so that the selected set could have a higher chance to illustrate the required reasoning processes.

In practice, it is tricky to decide whether the reasoning underlying a set of exemplars is complementary categorically. We therefore use diversity as a proxy, since a set of less similar exemplars is arguably more likely to exhibit complementarity. To that end, we propose a maximal-marginal-relevance (Carbonell and Goldstein, 1998) (MMR) based exemplar selection strategy. The idea is to select exemplars that are relevant to the query while being diverse enough to be collaborative. Suppose for the given query q, we have already selected a set of exemplars $T = \{q_i\}$, then we will pick up the next exemplar according to:

	code-davinci-001				code-davinci-002				text-davinci-002			
	GSM	ECQA	E-SNLI	AVG	GSM	ECQA	E-SNLI	AVG	GSM	ECQA	E-SNLI	AVG
LLM NN	18.5	56.0	57.4	43.9	65.8	76.8	**81.6**	74.7	52.0*	74.3*	**83.9***	70.0*
LLM MMR	18.7	**57.2**	**59.5**	**45.1**	67.0	77.4	81.5	**75.3**	**52.8***	**75.3***	83.7*	**70.6***
BERTScore NN	18.5	54.6	53.7	42.3	66.7	75.9	75.6	72.8	51.0	72.8	78.7	67.6
BERTScore MMR	**19.4**	56.3	53.9	43.2	**68.2**	**78.1**	77.8	74.7	52.0	73.7	78.2	68.0

Table 4: Results of using MMR-based exemplar selection strategy on three datasets (AVG denotes the average). Using MMR generally selects better exemplars on all datasets, using either LM-based method or BERTScore. The results on `text-davinci-002` use the LM-based scores provided by `code-davinci-002` (denoted by *).

$$\arg\max_{q_j \in D/T} \lambda \mathcal{S}(q, q_j) - (1-\lambda) max_{q_i \in T} \mathcal{S}(q_j, q_i) \quad (1)$$

where $\mathcal{S}$ denotes similarity and λ is a parameter that controls the balance between relevance and diversity. We rely on MMR to iteratively select exemplars from the exemplar pool, as shown in Algorithm (1). Note that this requires scoring all exemplar pairs within the pool. To run inference over m queries using a pool of n exemplars, MMR requires to score the similarity of $nn + mn$ pairs.

Results We apply the MMR strategy on top of LM-based method and BERTScore, leaving out the CLS-based approach which has inferior performance. The experimental setup largely follows Section 4.2, please refer to Appendix D for details.

We show the results in Table 3. In the setting that uses BERTScore, MMR-based selection successfully improves the performance for almost all LLMs for all datasets, compared to using nearest neighbors. On LM-based method, MMR is also able to improve the performance for GSM and ECQA across all LMs, and only marginally underperforms NN for E-SNLI.

In particular, using the MMR-based selection strategy achieves an accuracy of 68.2 and 78.1 on GSM and ECQA respectively, even outperforming LM-based method that requires a large number of queries to the LM. This suggests that BERTScore and MMR as a combination are able to construct effective explanation-infused prompts that approach that of actually querying LLMs. Furthermore, the fact that LLMs achieve better performance from the exemplars selected using our MMR-based method is congruent with our analysis in the previous section: LLMs can exploit complementary explanations.

5.1 Analysis

Impacts of the Trade-off Between Relevance and Diversity We conduct an analysis to investigate how the trade-off between diversity and relevance impacts the performance. We test the performance under varying λ on `code-davinci-002` with BERTScore as the similarity metric. We note this is done on one pool of training exemplars. Generally, when λ is large (0.8), the performance is similar to NN ($\lambda = 1.0$). MMR typically works well with a λ of 0.6 or 0.5 (roughly balancing the two terms). The performance starts to degrade while decreasing λ from 0.4 to 0, as the selected exemplars are not relevant enough.

λ	GSM	ECQA	E-SNLI
1.0	66.7	75.9	75.6
0.8	66.9	75.6	76.6
0.6	**68.2**	77.9	**78.1**
0.5	**68.2**	**78.1**	77.8
0.4	66.8	75.7	76.0
0.2	65.9	75.9	74.9
0.0	63.5	75.5	75.5

Table 5: The performance of MMR exemplar selection strategy with varying λ.

	GSM	ECQA	E-SNLI
Random	$65.4_{1.3}$	$74.1_{0.5}$	$74.0_{1.2}$
NN	$68.6_{0.7}$	$75.4_{0.5}$	$75.9_{1.1}$
MMR	$69.4_{1.0}$	$77.8_{0.7}$	$77.8_{0.9}$

Table 6: $\text{Mean}_{\text{variance}}$ of the performance across 5 random order. Using better exemplars has more significant impact than varying exemplar order.

Sensitivity to Different Order We have shown choosing exemplar sets using MMR can lead to better ICL performance, which could be affected by other confounders such as the order of exemplars. We conduct experiments to show that using better exemplar sets has more impact than reordering exemplars. Specifically, we experiment with 5 random orders of the exemplar sets for each query and report averaged performance and variance of the accuracy. We note this is done on one pool of training exemplars for each dataset using

code-davinci-002 with BERTScore as the metric. As shown in Table 6, MMR is still substantially better than NN and Random under varying order.

6 Related Work

The growing scale of pretrained language models has granted them the ability to learn a new task from a few examples via in-context learning (Brown et al., 2020). Various approaches have been proposed to improve ICL in recent years, including meta-tuning LLMs (Min et al., 2022b; Chen et al., 2022), calibration of ICL (Zhao et al., 2021; Han et al., 2022), automatically determining the orders of exemplars (Lu et al., 2022), and alternative formulation of ICL based on PMI (Holtzman et al., 2021) or noisy-channel (Min et al., 2022a). More closely related to our work, prior research also contributes to better understanding ICL as Bayesian inference (Xie et al., 2022) or experiments that study what makes in-context learning works (Min et al., 2022c). Our work focuses on understanding the usage of explanations in ICL, as opposed to standard prompting where the LLMs are presented with only input-output pairs.

In particular, our work is connected to prior research on effective ways for selecting in-context exemplars (Shin et al., 2021; Liu et al., 2021; Rubin et al., 2022; Qiu et al., 2022; Su et al., 2022). While past work primarily focuses on the effectiveness of using relevant examples in the standard prompting paradigm, we examine the benefits of the complementary exemplars when prompting with explanations. We also propose an MMR-based strategy, which is more effective than the NN-based approach on various LLMs across three tasks.

Lastly, including textual explanations in prompts has exhibited remarkable benefits for LLMs to learn various reasoning tasks. Using Scratchpad (Nye et al., 2021) or Chain-of-Thought (Wei et al., 2022) significantly boosts ICL performance on multi-step reasoning tasks such as arithmetical reasoning and symbolic reasoning. Using free-text rationales is also helpful for more unstructured tasks like QA and NLI (Ye and Durrett, 2022; Wang et al., 2022a). While recent work largely aims to find better ways to prompt LLMs with explanations (Kojima et al., 2022; Zhou et al., 2022a; Press et al., 2022; Zhou et al., 2022b), we focus on analyzing the role of explanations in ICL and what makes effective explanations.

7 Conclusion

We have presented a series of studies on what makes effective explanations for in-context learning. We first investigated the impacts of computation traces and natural language in explanations. Through a set of probing experiments, we found that LLMs rely on both of them to effectively learn from explanations. We further examined the interplay among exemplars within prompts and the interplay between exemplars and the query. Our analysis uncovered the benefits of constructing prompts by selecting complementary explanations that are relevant to the query. Lastly, we proposed an MMR-based exemplar selection strategy, which successfully improved the end task performance across three important datasets.

8 Limitations

The models chosen in this work are selected to represent the state-of-the-art at the time the work was conducted, and in some cases omit weaker models. For example, our exemplar selection experiments do not cover those LLMs trained with vanilla language models objectives, namely OPT and davinci, as we find their performance substantially lags code-davinci-002 and text-davinci-002. For the same reason, we only consider the substantially large language models, omitting LLMs of smaller scales (e.g., text-curie-001). Running experiments using smaller LMs or vanilla LMs may provide insights into how scale or instruction finetuning impacts the ability of LMs in learning from explanations, but our investigation mainly focus on selecting exemplars to achieve the best in-context learning performance with state-of-the-art models.

In addition, certain aspects of our approach are computationally intensive, particularly using LM-based similarity scores. However, we think this is still feasible in practice: if practitioners are deploying a real-world system, investing more computation upfront to improve its performance is likely in reach for those deploying LLMs in practice.

Finally, our experiments consider a certain subset of NLP reasoning tasks written in English. While we believe the results here transfer to other tasks in this vein which have been frequently used to evaluate LLMs, it is unknown how well they handle other languages, dialects, or genres of text such as social media data.

Acknowledgments

Thanks to anonymous reviewers for their helpful feedback and colleagues at Meta AI for helpful discussions. This work was partially supported by NSF CAREER Award IIS-2145280 and the NSF AI Institute for Foundations of Machine Learning (IFML).

References

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for CommonsenseQA: New Dataset and Models. In *Proceedings of the Annual Conference of the Association for Computational Linguistics (ACL)*.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. *arXiv*, abs/2202.01279.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Association for Computing Machinery.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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 Baindoor Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. 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 García, 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 Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Zhi Han, Yaru Hao, Li Dong, and Furu Wei. 2022. Prototypical calibration for few-shot learning of language models. *ArXiv*, abs/2205.10183.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916.

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022b. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022c. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

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*, abs/2210.03350.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. 2022. Evaluating the impact of model scale for compositional generalization in semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the Annual Conference of the Association for Computational Linguistics (ACL)*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *ArXiv*, abs/2205.01068.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022a. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625.

Hattie Zhou, Azade Nova, H. Larochelle, Aaron C. Courville, Behnam Neyshabur, and Hanie Sedghi. 2022b. Teaching algorithmic reasoning via in-context learning. *ArXiv*, abs/2211.09066.

A Details of the Explanations Used for LETCAT and COINFLIP on OPT

LETCAT	**Question:** Take the last letters of the words in "Bill Gates" and concatenate them.
	Ours: Add space to "Bill" and get "B i l l", the last letter is l. Add space to "Gates" and get "G a t e s", the last letter is s. Concatenating l and s is ls. So the answer is ls. **Wei et al. (2022):** The last letter of Bill is l. The last letter of Gates is s. Concatenating l and s is ls. So the answer is ls.
COINFLIP	**Question:** A coin is heads up. Shaunda does not flip the coin. Shalonda flips the coin. Is the coin still heads up?
	Ours: The coin started heads up. Shaunda does not flip the coin, so it becomes heads up. Shalonda flips the coin, so it becomes tails up. So the answer is no. **Wei et al. (2022):** The coin was flipped by Shalonda. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.

Figure 4: Examples of original chain-of-thoughts from Wei et al. (2022) and ours used for OPT.

	LETCAT	COINFLIP
Standard	8.5	51.5
(Wei et al., 2022)	29.5	61.0
Ours	50.0	94.0

Table 7: Performance of original chain-of-thoughts and our explanations used for OPT.

For GSM, we directly use the gold explanations provided in Wei et al. (2022). For LETCAT and COINFLIP, we take the original explanations from Wei et al. (2022) and manually engineered them, as the original ones are sub-optimal for OPT and do not lead to credible gains compared to standard prompting.

We show examples of the original explanations (chain-of-thoughts) used in Wei et al. (2022) and the explanation we adapted for OPT in Figure 4. For LETCAT, we add another step of tokenizing the two words. For COINFLIP, we change the way of decomposing the problem. As shown in Table 7, our adapted explanations lead to more substantial performance improvements over standard prompting. We use engineered explanations for the probing experiments on OPT, which allows more distinguishable performance differences. We refer readers to the supplementary materials of Wei et al. (2022) for the complete set of exemplars and explanations.

B Details of the Choice of Masks

We conduct preliminary experiments on the LETCAT dataset using `davinci` to determine the choice of masks. We tested masking with "N/A", "[mask]", "?", "_", and empty string. The results obtained using different masks are shown in Table 8. Whatever masks are used, LLMs see performance degradation compared to gold explanations, but can still learn from partially complete explanations. We use an empty string as the mask token across all datasets, which leads to the least performance degradation.

	N/A	[mask]	?	_	Empty Str
Standard			8.5		
Gold			59.0		
Mask1	14.0	14.0	15.0	13.5	16.0
Mask2	48.0	48.0	48.5	43.0	49.5

Table 8: Results of using different mask tokens for LETCAT on OPT.

C Details of the Setup for Exemplar-Exemplar Interplay Experiments

For LETCAT, we experiment with 4 exemplars where the first steps are perturbed, 4 exemplars where the second steps are perturbed and a mixture of 2 from each of these explanations. For COINFLIP, we use 8 exemplars and follow the same setting. For the mixture type of prompts, we experiment with 4 random combinations for mixing two types of masked exemplars.

For GSM, we use three types of prompts constructed by 1) 8 addition-only exemplars, 2) 8 multiplication-only exemplars, and 3) a mixture of 4 exemplars from each of the two types. We note that unlike what's in LETCAT and COINFLIP which uses identical exemplars perturbed in different ways, the exemplars in the three sets, for GSM are drawn from different pools. We experiment with 4 different sets of randomly drawn examples and report the average in the setting. We note the test set for GSM that requires composing addition and multiplication contains 1,150 data points in total.

D Details of the Setup for MMR-Based Exemplar Selection Experiments

We evaluate the effectiveness of our MMR-based exemplar selection strategy on the three realistic datasets used in Section 4.2. Also, the experiments on `text-davinci-002` using the LM-based method rely on similarity scores obtained

from `code-davinci-002`, given the prohibitive cost needed to run these experiments. We do not tune λ in our experiments. λ is set to roughly balance the variance among the two terms in Equation (1), which is 0.5 across all datasets and methods, except for using the LM-based method on `code-davinci-001`. For this particular setting, we set lambda to be $\frac{2}{3}$, as we observe higher variance among the diversity with exemplars.

E Details of Prompts for Real-world Datasets

We showcase how we format the prompts for GSM, ECQA, and E-SNLI in Figure 5, with one exemplar for each of the three datasets. We note the prompt format for E-SNLI is taken from PromptSource (Bach et al., 2022).

GSM
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total? A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74 chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39.
ECQA
Q: Where can you get a bugle to take home with you? Answer Choices: (a) farmer's wife (b) music store (c) military base (d) military band (e) american army. A: Bugle is a musical instrument. Musical instruments are available in a music store. Music store is a building. So the answer is (b).
E-SNLI
Premise: "A man at a flea market browsing." Based on this premise, can we conclude the hypothesis "A man is sleeping at a flea market." is true? OPTIONS: - yes - no - not possible to tell A: One cannot be sleeping and browsing at the same time. The answer is no.

Figure 5: Detailed examples of prompts for GSM, ECQA, and E-SNLI.

F License of Datasets

- GSM (Cobbe et al., 2021): MIT license.
- E-SNLI (Camburu et al., 2018): MIT license.
- ECQA (Aggarwal et al., 2021):Community Data License Agreement - Sharing - Version 1.0.
- LETTER CONCATENATION and COINFLIP (Wei et al., 2022): MIT license.