Integrate the Essence and Eliminate the Dross: Fine-Grained Self-Consistency for Free-Form Language Generation

Anonymous ACL submission

Abstract

Self-consistency (SC), leveraging multiple samples from LLMs, shows significant gains on various reasoning tasks but struggles with free-form generation due to the difficulty of aggregating answers. Its variants, UCS and USC, rely on sample selection or voting mechanisms to improve output quality. These methods, however, face limitations due to their inability to fully utilize the nuanced consensus knowledge present within multiple candidate samples, often resulting in suboptimal outputs. We propose Fine-Grained Self-Consistency (FSC) to addresses these limitations by extracting and integrating segment-level commonalities from candidate samples, enhancing the performance of LLMs both in open-ended and reasoning tasks. Based on this, we present two additional strategies: candidate filtering, which enhances overall quality by identifying highly similar candidate sets, and merging, which reduces input token requirements by combining similar samples. The effectiveness of FSC is demonstrated through extensive experiments on various tasks, including summarization, code generation, and mathematical reasoning, using GPT-3.5-turbo and GPT-4. The results indicate significant improvements over baseline methods, showcasing the potential of FSC to optimize output quality by effectively synthesizing fine-grained consensus knowledge from multiple samples.

1 Introduction

The remarkable success of large-scale language models (LLMs) have transformed the landscape of natural language processing, showcasing significant improvements across a wide range of tasks, from reasoning tasks (Wei et al., 2022) with distinct answers like arithmetic and commonsense reasoning to free-form generation tasks like code generation (Austin et al., 2021) and summarization (Goyal et al., 2022).

However, LLMs may still generate suboptimal samples in challenging tasks. Efforts to improve output quality involve selecting the best response from multiple samples based on specific criteria. This includes using trained models for reranking outputs (Ravaut et al., 2023) and employing LLMs to evaluate the responses (Liu et al., 2023b). However, both approaches require additional models and overlook the knowledge present among the candidates. Wang et al. (2023) introduce self-consistency (SC) to improve performances without additional models, which mitigates noise from individual sampling by employing a voting mechanism across multiple samples. Unfortunately, SC is limited to tasks where the final answer can be aggregated through precise matching. How to aggregate the answers for free-form problems remains unclear.

Recently, some works seek to evolve the idea of self-consistency into open-ended generative tasks. UCS (Jain et al., 2023) calculates the overlap of unigrams between candidates and then selects the final answer with highest value. Alternatively, USC (Chen et al., 2023) leverages the capabilities of LLMs instead of rule-based criteria to choose the most consistent one. However, both approaches still rely on selection or voting mechanisms, which do not align well with the nature of free-form generation tasks, i.e., the final quality is determined by the entirety of the output content, rather than specific individual tokens. Therefore, sample-level selection methods only yield suboptimal outputs, primarily due to two reasons: (1) They are unable to incorporate consensus knowledge from unselected samples, which, despite their lower overall quality, may contain locally valuable information to enhance the quality of selected samples. (2) They cannot eliminate low-quality segments from selected samples to further improve overall quality. In a word, such coarse-grained selection methods suppress or overlook the role of fine-grained con-

sensus within the candidate samples. Figure 1 and table 1 depict scenarios that select-based SC methods struggle to address. The former indicates that both methods perform poorly when the quality of candidates is low for code generation tasks. The latter illustrates with case the scenario where, in summarization tasks, the semantic information of the ground truth cannot be fully covered by any one of candidate samples. Thus, regardless of the selection, only suboptimal outputs can be obtained.

Distribution	USC	UCS	Random
5 / 0	0.0 %	0.0 %	0.0 %
4 / 1	13.9 %	13.9 %	20.0 %

Table 1: Accuracy on code generation benchmark HumanEval with gpt-3.5-turbo. We generate 5 samples for each problem. The term "5/0" distribution pertains to cases where all five generated codes are erroneous, while "4/1" distribution indicates that one sample is correct while the other four are incorrect.

To address this issue and better leverage the consensus knowledge among multiple samples for open-ended problems, we propose Fine-Grained Self-Consistency (FSC). Specifically, after generating multiple candidate samples, FSC extracts the segment-level common elements within them by taking full advantage of LLM's text comprehension and contrasting capabilities. Subsequently, it synthesizes these consensus elements to produce an optimized output, effectively integrating the essence from the candidate samples. Based on this, we further propose two strategies: candidates filtering and merging. The former employs automated metrics to identify candidate sets that exhibit high similarity, thereby enhancing the overall quality of the candidates. The latter strategy merges samples that are highly similar, effectively reducing the quantity of input tokens required.

A wide range of tasks, including summarization, code generation, and formal mathematical reasoning, are evaluated on gpt-3.5-turbo and gpt-4 for proposed FSC. The results show that proposed method outperform baselines in a large margin. Additional experiments indicate that filter strategy can further enhance performance by selecting better candidates and merge strategy can reduce cost while maintaining the performance.

To summarize, this work includes the following contributions:

- We propose Fine-Grained Self-Consistency, designed to fully leverage the consensus knowledge extracted within multiple samples for tasks involving free-form generation.
- Two strategies, candidates filtering and merge, are devised to improve performance and minimize costs.
- Extensive experiments show that our proposed method can surpasses competitive baselines.

2 Related Work

Consistency-Based Response Selection Approaches. The literature presents a variety of consistency-based response selection approaches, typically incorporating a voting procedure to select the most frequently occurring response (Wang et al., 2023; Zhou et al., 2022; Portillo Wightman et al., 2023; Yue et al., 2023). The self-consistency approach proposed by (Wang et al., 2023) demonstrates that by generating multiple responses for the same task and selecting the reasoning path leading to the most common final answer, the performance of chain-of-thought reasoning can be improved. Candidate responses can also be derived from different prompt variants corresponding to the same problem (Zhou et al., 2022; Portillo Wightman et al., 2023; Yue et al., 2023). For open-ended generation tasks, Jain et al. (2023) propose the n-gram consistency score to measure the pairwise similarity between candidate responses. The consistency score for each response is calculated as the sum of the pairwise similarity scores. Chen et al. (2023) propose leverage LLMs themselves to directly select the most consistent answer among multiple candidates without an explicit definition of the pairwise similarity. In this work, we take a closer look into the coarse-grained limitations faced by selection based self-consistency methods on open-ended generation tasks, and propose Fine-Grained Self-Consistency to fully leverage the segment-level consensus knowledge extracted within multiple samples so as to gain better consistency output.

Response improvement with multiple candidates. Recent studies show that the LLM can enhance its own prediction output based on candidate responses. Zheng et al. (2023) demonstrate that the given a trajectory of previously generated solutions, the LLM can iteratively produce superior solutions for an optimization task. Other researches focus on aggregating multiple reasoning

Completions:
(1)Andy Murray, British No 1 and two-time Grand Slam champion, is set to marry his girlfriend of almost 10 years, Kim Sears, in his hometown of Dunblane, Scotland.
(2)When they were teenagers at the US Open in New York in 2005, Andy Murray and his long-time girlfriend Kim Sears met and they will get married in the Scottish town of Dunblane on Saturday.
(3)Andy Murray and his girlfriend Kim Sears are set to get married in Dunblane, Scotland this Saturday. The couple, who have been together for over nine years, announced their engagement in November of last year.
(4)Murray and Sears, who met when they were teenagers at the US Open in New York in 2005, confirmed their engagement last November after more than nine years together.
(5)Andy Murray and his long-term girlfriend Kim Sears are set to get married in the Scottish town of Dunblane on Saturday.

Ground truth: Andy Murray and long-time girlfriend Kim Sears will tie the knot in the Scottish town of Dunblane on Saturday. The British No 1 and his partner met when they were teenagers at the US Open in New York in 2005. Murray and Sears confirmed their engagement last November after more than nine years together.

FSC: Andy Murray, the British No 1 and two-time Grand Slam champion, is set to marry his girlfriend of almost 10 years, Kim Sears, in his hometown of Dunblane, Scotland. The couple met as teenagers at the US Open in New York in 2005 and got engaged last November.

Figure 1: Case from DailyMail with gpt-3.5-turbo. Each color represents a distinct semantic information. The input document is provided in the Appendix A.1 due to space constraints.

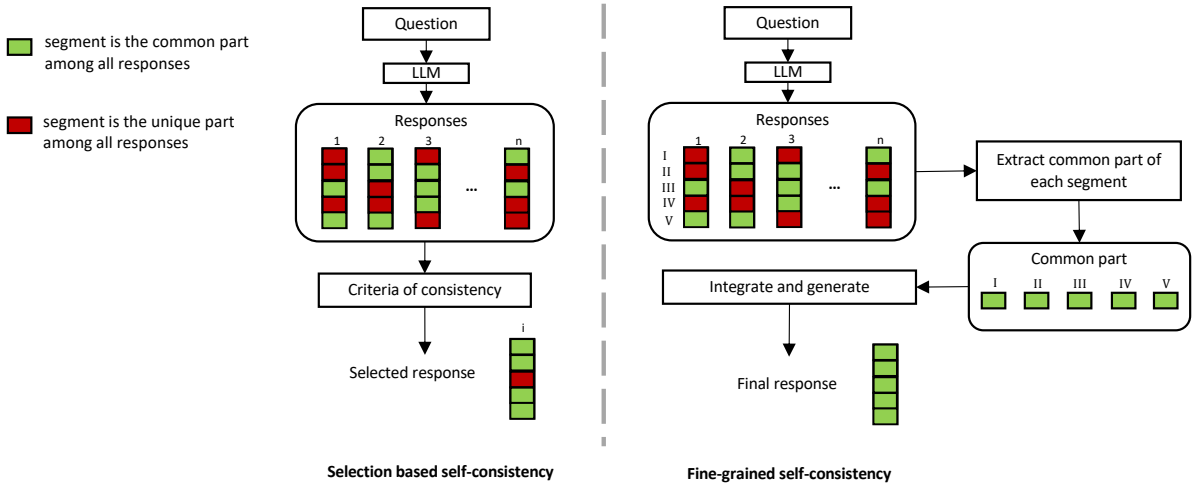


Figure 2: Illustration of selection based self-consistency and proposed fine-grained self-consistency.

chains and prompting the LLM to generate a better final response, which shows performance enhancement in multi-hop question answering (Yoran et al., 2023), mathematical reasoning (Yang et al., 2023), machine translation (Zhang et al., 2024) and code generation (Huang et al., 2023). Instead of prompting the LLM to generate a superior response, our proposed FSC focuses on extracting and integrating consistency, eliminating the need for post-answer guidance. We demonstrate that FSC effectively leverages multiple responses to enhance performance across a range of tasks.

3 Method

The core idea behind fine-grained self-consistency is to measure and extract the commonality of each response generated by LLM at the segment level, and then fuse to generate the final response with superior commonality. Figure 2 illustrates the idea of select-based self-consistency and the proposed fine-grained self-consistency (FSC). Specifically, select-based self-consistency ranks the responses generated by LLM according to specific consistency criteria (Jain et al., 2023; Chen et al., 2023), and the top-ranked response is selected as the output. However, these methods do not assess the consistency at a more fine-grained level within the response. On the other hand, FSC first measures the commonality of each segment of the responses generated by LLM, and extracts the corresponding common part, then fuses the common part and generates the final output, achieving a more refined consistency sample output.

We present the overall workflow of Fine-grained self-consistency (FSC) in Figure 3, Including FSC with two plugins, candidates filtering and merge.

3.1 Fine-Grained Self-Consistency

Considering that it is difficult to divide and measure the consistency of the segments in the responses

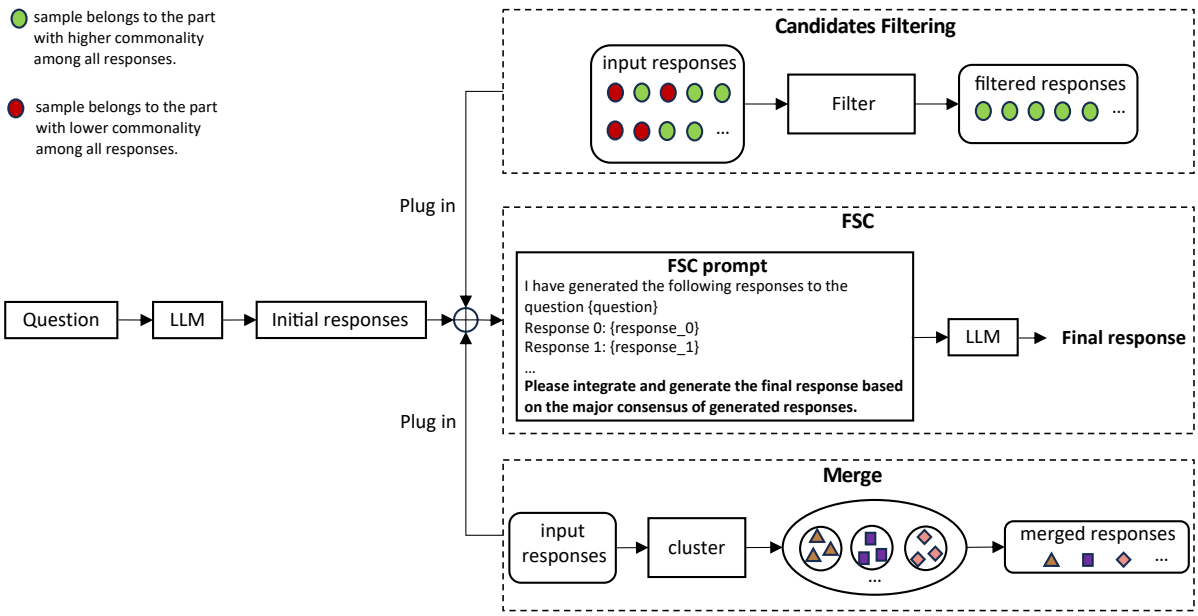


Figure 3: Overview of the proposed fine-grained self-consistency (FSC) workflow.

based on prior knowledge, we seek to utilize the comparative and integrate capabilities of LLMs to inherently extract the fine-grained common part and integrate consistency knowledge from different responses. As shown in Figure 3, For the multiple input responses, we concatenate them all and construct a prompt with an instruction asking the LLM to extract the major consensus of generated responses, then integrate and generate the final response. In this way, FSC measures commonality at a finer granularity, utilizing, and integrating the consistency knowledge from different responses, alleviating the coarse-grained limitations of the selection based self-consistency.

3.2 Candidates filtering

As shown in Figure 3, Considering the varying quality of responses generated by LLM, We propose candidates filtering strategy to measure the consistency of generated responses at the sample level, eliminating responses with low consistency to ensure the overall quality of the candidate responses. Specifically, Given the similarity function Sim, we can define a sample-level self-consistency score $SC_{Sim}(i)$ for each response i, given by $SC_{Sim}(i) = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N} Sim(i)$. Here, N represents the number of responses, and we use Rouge (Lin, 2004) for our similarity function Sim. In the end, we take the Top-k responses according to SC_{Sim} as the filtered candidates set, where k is a hyperparameter.

3.3 Merge

As shown in Figure 3, To reduce the computation cost of FSC, we propose merging semantically similar candidates to decrease the number of responses to be integrated by the LLM. Furthermore, due to the limitations of the LLM input length, the number of samples inputted to the LLM for consistency extraction is limited. By merging, we can provide the LLM with samples that contain more diverse knowledge, broadening the LLM's knowledge field of view. specifically, To merge similar responses, we employ the K-Medoids clustering algorithm based on their semantic similarity. We categorize all responses into c clusters, each encompassing a set of similar results. Then we select the centroids of each cluster as representative responses and discard the remaining ones. It ensures the selected response has diverse knowledge and reduces the cost of FSC.

4 Experiment

4.1 Evaluation setup

4.1.1 Benchmarks

We evaluate FSC on the following variety of tasks:

Code generation We conduct experiments on three widely used code generation benchmarks,

Model	Method	HumanEval	HumanEval+	BIRD-SQL	
		Execution Accuracy	Execution Accuracy	Execution Accuracy	Efficiency
gpt-3.5-turbo	Random	69.8 -	63.2 -	43.1 -	43.9 -
	UCS	70.1 ↑0.3	63.2 ↑0.0	43.6 ↑0.5	44.5 ↑0.6
	USC	70.5 ↑0.4	66.3 ↑3.1	43.6 ↑0.5	44.6 ↑0.7
	FSC	**74.5** ↑4.7	**68.4** ↑5.2	**45.3** ↑2.2	**46.0** ↑2.1
gpt-4	Random	82.5 -	76.4 -	49.5 -	50.6 -
	UCS	82.3 ↓0.2	76.7 ↑0.3	50.5 ↑1.0	51.8 ↑1.2
	USC	86.1 ↑3.6	80.9 ↑4.5	50.9 ↑1.4	51.6 ↑1.0
	FSC	**87.1** ↑4.6	**82.8** ↑6.4	**51.4** ↑1.9	**52.2** ↑1.6

Table 2: The results on three code generation benchmarks. The improvements are calculated between each methods and Random. The best performance for each dataset are shown in **bold**.

including HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023a) for Python code generation and BIRD-SQL dataset (Li et al., 2023) for text-to-SQL generation. HumanEval (Chen et al., 2021) is a hand-written Python programming problem, which is further enhanced by HumanEval+ (Liu et al., 2023a) through the addition of more unit tests. BIRD-SQL (Li et al., 2023) is a much more challenging dataset consisting of text-to-SQL tasks across 37 professional domains, derived from 95 databases with a total size of 33.4 GB.

Text summarization We conduct our experiments on two widely used text summarization datasets: DailyMail for short text summarization (Nallapati et al., 2016) and SummScreen for long text summarization (Chen et al., 2022). In the DailyMail dataset, each input is an ~ 800 words news article, and each reference output is a human-written summary of the article with ~ 55 words. In SummScreen, every input is a transcript of a TV show episode with $\sim 5,600$ words, and each reference output is a ~ 100 words human-written recap of the episode. We follow Nallapati et al. (2016) and measure ROUGE 1, ROUGE 2, and ROUGE-L[1] which measure n-gram overlap with the reference summary, and we also measure BERTScore F1[2] (Zhang et al., 2019).

Mathematical reasoning We introduce the widely used GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al.) datasets to verify the generalizability of the proposed method on tasks with answer of fixed form. GSM8K consists of 8,500 grade school math word problems, and MATH consists of 12,500 challenging mathematics problems from high school competitions.

4.1.2 Baselines

We compare FSC to the following self-consistency methods: (1) Random selects one answer randomly from multiple responses with temperature > 0; (2) UCS (Jain et al., 2023) calculates the overlap of unigrams between candidates and then selects the final answer with highest value;[3] (3) USC (Chen et al., 2023) utilizes LLMs to choose the most consistent one as the final answer. (4) SC (Wang et al., 2023) is the standard self-consistency decoding with answer extraction, which mitigates noise from individual sampling by employing a voting mechanism across multiple samples. Specifically, random select represents the performance of the LLM itself when the temperature > 0, while UCS and USC are two strong baselines for selection based self-consistency methods. Since the outputs of code generation and summarization tasks are in free-form, we evaluate SC on mathematical reasoning benchmarks where the final answers can be compared through exact match.

4.1.3 Implementation details

We conduct experiments using gpt-3.5-turbo and gpt-4 models. We set the temperature as 0.8 for both gpt-3.5-turbo (ChatGPT)[4] and gpt-4 (Achiam et al., 2023) models to generate 50 initial responses For all benchmarks. For summarization and python code generation, the initial samples are generated

[1]We use the implementation of https://github.com/pltrdy/rouge.

[2]We use the bert-base-uncased version for evaluation.

[3]As token probabilities cannot be obtained from gpt-3.5-turbo and gpt-4, we implemented UCS based on its unigram version.

[4]https://chat.openai.com

Model	Method	DailyMail				Summscreen			
		Rouge1	Rouge2	RougeL	BertScore	Rouge1	Rouge2	RougeL	BertScore
gpt-3.5-turbo	Random	37.3	14.1	38.4	60.6	18.3	2.1	16.8	48.8
	UCS	36.9	14.0	38.2	60.5	16.9	2.0	16.4	48.3
	USC	37.7	14.3	38.7	60.8	19.3	**2.3**	17.2	49.3
	FSC	**38.6**	**14.4**	**39.0**	**60.9**	**20.1**	2.1	**17.4**	**49.6**
gpt-4	Random	37.5	14.5	38.9	61.0	19.1	2.6	17.3	49.5
	UCS	36.9	14.3	38.6	60.9	18.6	2.4	17.0	49.3
	USC	37.6	14.6	39.0	61.1	19.3	2.7	17.5	49.6
	FSC	**38.1**	**14.7**	**39.3**	**61.2**	**19.5**	**2.9**	**17.8**	**50.0**

Table 3: Results on summarization benchmarks. FSC consistently improves over the baselines on summary quality.

Model	Method	GSM8K	MATH
gpt-3.5-turbo	Random	75.9	35.0
	UCS	77.2	35.6
	USC	80.6	39.3
	SC	**82.0**	**41.9**
	FSC	<u>81.0</u>	<u>39.5</u>
gpt-4	Random	87.5	50.2
	UCS	87.6	50.7
	USC	88.1	54.8
	SC	<u>88.8</u>	<u>55.5</u>
	FSC	**91.3**	**56.0**

Table 4: Accuracy on mathematical reasoning benchmarks. FSC performance is comparable to SC. The best and second best performance for each dataset are shown in **bold** and <u>underline</u>.

with zero-shot prompting, thus the output formats are diverse. For BIRD-SQL, we used the 1-shot chain-of-thought prompt in Li et al. (2023), which improves the performance. Considering the cost of experiments, we randomly select 1,000 samples from the test splits of DailyMail and SummScreen respectively to form our text summarization benchmarks. We set the temperature of FSC and USC as 0 to ensure the reproducibility of the results. Unless otherwise specified, we set the default number of input responses as 5 For all baselines. All experiments are repeated five times and the average performance is reported.

4.2 Main results

4.2.1 Code generation

As shown in Table 2, we present the results on HumanEval, HumanEval+ and BIRD-SQL respectively. Besides the execution accuracy, we follow (Li et al., 2023) to also evaluate the valid efficiency score on BIRD-SQL, which measures the efficiency of the generated SQL queries. We show that FSC outperforms all baselines on execution accuracy by a significant margin across all datasets, while also generating more efficient SQL code.

4.2.2 Text summarization

Table 3 presents the results for summarization benchmarks. In both datasets FSC consistently improves over the baselines across all metrics, which demonstrates that FSC can improve performance in both short and long text summarization tasks simultaneously.

4.2.3 Mathematical reasoning

As shown in Table 4, Besides selection based self-consistency methods, we compare FSC against the standard self-consistency (SC). For SC, we employ a regular expression matching to extract the final answer on GSM8K, and reuse the answer parsing code from (Li et al., 2024) for MATH. Overall, FSC consistently improves over the selection based self-consistency methods UCS and USC, and the performance is generally comparable to SC, which needs answer parsing to perform the voting. surprisingly, FSC outperform SC on gpt-4, which demonstrates that FSC is not simply dependent on statistical measures of final reasoning answers, and its analysis and integration of various reasoning paths are effective. These results suggest that FSC could be further generalized to tasks where answer extraction is feasible for voting.

4.3 Effect of candidates filtering

As shown in Table 5, We compare the performance of FSC combined with candidates filtering strategy (denoted as FSC+Filter) with FSC itself and selection based self-consistency baselines. Specifically, FSC+Filter performs candidate filtering on the initial N responses to obtain $\frac{N}{2}$ filtered responses, and

Method	N	HumanEval	HumanEval+	DailyMail			
		Accuracy	Accuracy	Rouge1	Rouge2	RougeL	BertScore
UCS	10	70.7	62.5	38.2	14.4	39.2	60.9
USC	10	71.0	65.9	38.5	14.5	39.1	**61.1**
FSC	5	74.5	68.4	38.6	14.4	39.0	60.9
FSC+Filter	10	**75.5**	**69.2**	**38.7**	**14.6**	**39.3**	61.0

Table 5: Results of candidates filtering on HumanEval, HumanEval+ and DailyMail with gpt-3.5-turbo. N represents the number of input responses. The best performance for each dataset is shown in **bold**.

Method	N	HumanEval	HumanEval+	Save	DailyMail				Save
		Accuracy	Accuracy		Rouge1	Rouge2	RougeL	BertScore	
FSC	5	74.5	68.4	-	38.6	**14.4**	**39.0**	**60.9**	-
FSC	4	74.3	67.8	20.0%	**38.8**	14.2	**38.8**	60.8	20.0%
FSC+Merge	5	75.3	68.0	25.4%	**38.8**	14.0	38.6	60.8	26.5%
FSC+Filter+Merge	5	**75.6**	**68.5**	**40.0%**	38.6	**14.4**	38.9	**60.9**	**53.2%**

Table 6: Results of merge on HumanEval, HumanEval+ and DailyMail with gpt-3.5-turbo. "Save" represents how many candidate responses inputted into the LLM are saved compared to the default settings of FSC (N=5). We calculate "Save" through relative difference percentage. The best performance for each dataset is shown in **bold**.

then applies FSC to the filtered responses to get the final output. The results show that candidates filtering consistently improves FSC performance across all test benchmarks, indicates that candidates filtering obtains a higher-quality response candidate set through screening. On the other hand, the performance of FSC+Filter surpasses UCS and USC under the same number of response inputs on all test datasets (except for BertScore on DailyMial), demonstrating the superiority of FSC combined with candidates filtering strategy.

4.4 Effect of merge

We present the results of FSC combined with merge strategy in Table 6. The results show that the Merge strategy can significantly reduce the number of FSC input responses (20.0% on HumanEval and HumanEval+; 26.5% on Daily Mail), with minimal performance loss. Compared to saving costs by reducing the number of input responses ($N = 4$ for FSC), the Merge strategy saves more costs and maintains better performance (except for Rouge2 and RougeL on DailyMail), demonstrating its effectiveness. Furthermore, we combine FSC with both Filter and Merge strategies and achieve the best performance on HumanEval and HumanEval+, saving 40.0% of the cost. Besides, we save 53.2% of the cost on DailyMail with minimal performance drop. The results demonstrate the superiority of the combination of two proposed strategies.

4.5 Discussion

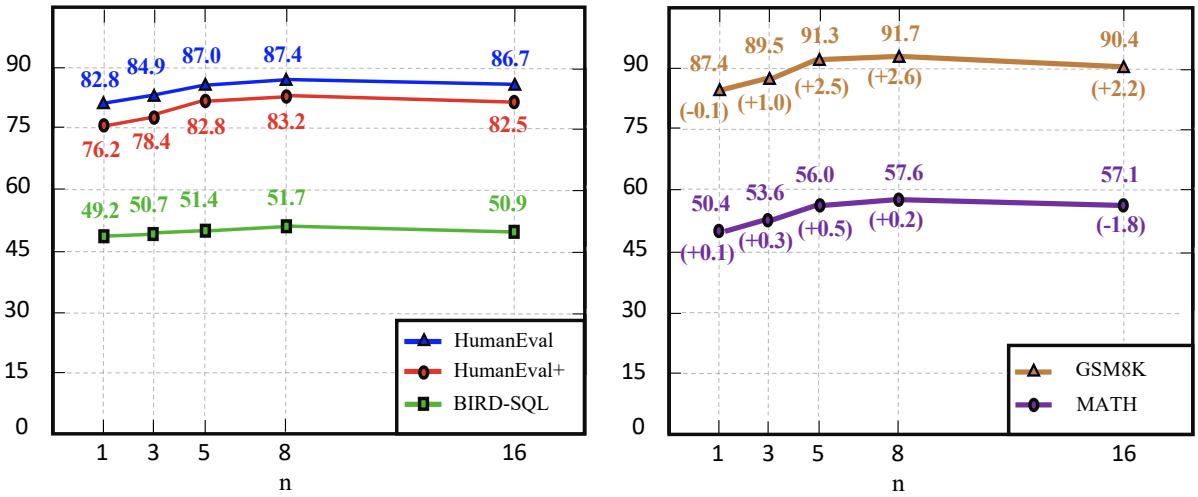
4.5.1 Different number of input responses

As shown in Figure 4, we examine the effect of using different numbers of responses (denoted as n) in FSC on gpt-4 model.[5] The results show that FSC consistently benefits from more input responses with $n \leq 8$. However, the performance of FSC decreases with $n = 16$, which could be due to the difficulty in understanding long-context when the prompt includes a larger number of input responses, as well as the length constraint of LLMs. Nevertheless, we believe that using a limited number of input responses (e.g., 5) strikes an ideal balance between task accuracy and computational cost. In such case, FSC reliably enhances performance across all benchmarks.

4.5.2 Robustness to noise in input responses.

Table 7 shows the accuracy on the HumanEval benchmark under different distributions. We define noise as the the proportion of erroneous examples in the input responses, (Distribution 5/0 corresponds to the maximum noise). While the performance of UCS and USC lag behind random select when the input noise is high, FSC consistently surpasses random select by a large margin under all distributions, proving that FSC has better robustness. Furthermore, the accuracy of FSC is greater than 0 when the distribution is 5/0, indicat-

[5] Due to budget constraints, we do not conduct the experiment on text summarization benchmark.



(a) Results on code generation.

(b) Results on mathematical reasoning. The top numbers are FSC accuracies, and the bottom numbers are the differences to SC accuracies.

Figure 4: FSC results on gpt-4 model with different number of input responses.

Distribution	gpt-3.5-turbo				gpt-4			
	Random	UCS	USC	FSC	Random	UCS	USC	FSC
5 / 0	0.0%	0.0%	0.0%	**1.1%**	0.0%	0.0%	0.0%	**2.2%**
4 / 1	20.0%	13.9%	13.9%	**27.9%**	20.0%	23.1%	16.6%	**29.1%**
3 / 2	40.0%	35.8%	47.1%	**62.2%**	40.0%	31.0%	51.7%	**55.1%**
2 / 3	60.0%	68.3%	56.6%	**70.0%**	60.0%	65.9%	**87.2%**	82.9%
1 / 4	80.0%	82.1%	86.9%	**91.6%**	80.0%	73.6%	91.6%	**97.2%**

Table 7: Accuracy on code generation benchmark HumanEval with gpt-3.5-turbo and gpt-4. Please refer to Table 1 for the definition of distribution. We mark values lower than random performance in red and values higher than random in green. The best performance is highlighted in **bold**.

ing that FSC can still recover the correct answer when all input responses are wrong. This demonstrates that FSC is capable of integrating the correct knowledge from different input responses and eliminating the wrong part to achieve the correct final response.

4.5.3 Consistency of FSC output

To gain a more intuitive understanding of the working mechanism of FSC, and to conduct a qualitative analysis of the consistency of FSC's final output, we present examples of FSC in code generation tasks in Section A.2. Specifically, for all error input responses, FSC incorporates consensus knowledge from each input and eliminates low-quality segments, ultimately recovering the correct solution. Our analysis indicates that FSC is capable of achieving fine-grained commonality extraction and obtaining outputs with better consistency compared to selection based self-consistency methods.

5 Conclusion

In this work, we propose Fine-Grained Self-Consistency (FSC), which fully leverage the segment-level consensus knowledge extracted within multiple samples to overcome the coarse-grained limitations faced by selection based self-consistency methods. To improve performance and minimize costs, we further propose two strategies called candidates filtering and merge. Extensive experiments demonstrate that FSC notably boosts the performance on diverse range of tasks, exhibits superior robustness against noise in input responses, and can be generalized to those tasks where answer extraction is feasible through voting. Additional experiments confirm that the proposed candidates filtering and merge strategies can further enhance the performance of FSC while reducing the required computational cost.

Limitations

Despite the remarkable performance gain on variety of tasks, the current implementation of FSC still suffers from the following two limitations:

1. As illustrated in Section 4.5.1, While self-consistency can be applied to any number of samples, the number of samples supported by FSC is bounded by the context length of the underlying LLM. That is to say, FSC would be limited in tasks that require lengthy responses, such as story generation, long text translation, etc.

2. FSC relies on LLM's own text understanding, comparison, and integration capabilities. For models with fewer parameters, the effectiveness of FSC may be limited due to their weaker capabilities. We will explore the generalizability of FSC on small models in the future work.

Ethics Statement

All of the datasets used in this study were publicly available, and no annotators were employed for our data collection. We confirm that the datasets we used did not contain any harmful content and was consistent with their intended use (research). We have cited the datasets and relevant works used in this study.

References

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

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *CoRR*, abs/2311.17311.

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

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of GPT-3. *CoRR*, abs/2209.12356.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *Sort*, 2(4):0–6.

Baizhou Huang, Shuai Lu, Weizhu Chen, Xiaojun Wan, and Nan Duan. 2023. Enhancing large language models in coding through multi-perspective self-consistency. *arXiv preprint arXiv:2309.17272*.

Siddhartha Jain, Xiaofei Ma, Anoop Deoras, and Bing Xiang. 2023. Self-consistency for open-ended generations. *CoRR*, abs/2307.06857.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. 2023. Strength in numbers: Estimating confidence of large language models by prompt agreement. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada. Association for Computational Linguistics.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2023. Unsupervised summarization re-ranking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8341–8376, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

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

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

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-contrast: Better reflection through inconsistent solving perspectives. *arXiv preprint arXiv:2401.02009*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Prompt consistency for zero-shot task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2613–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Appendix

A.1 Input document for Figure 1

Input: "The wedding of the year in Scotland takes place on Saturday when British No 1 and two-time Grand Slam champion Andy Murray marries Kim Sears, his girlfriend of almost 10 years, in his hometown of Dunblane. Murray and Sears, both aged 27, met when the pair were teenagers during the US Open in 2005. Murray was playing in only his second Grand Slam tournament, while Sears was travelling with her tennis coach father Nigel. The couple got back together in 2010 following a brief split and after having to field constant questions over the years on when he would propose, their engagement was confirmed last November. Andy Murray kisses his new girlfriend Kim Sears in the crowd after winning his first ATP World Tour title in San Jose in February 2006 . Murray pictured walking alongside Sears on the streets of London during the Wimbledon Championships in June 2006 . Murray watches his brother Jamie in action at London‘s Queen's Club in June 2007 alongside Sears and mother Judy (left) Murray watches his brother in action at Wimbledon in 2007 alongside Sears and Carlos Mier (right), who will be one of Murray‘s three best men . Murray and Sears watching British boxer Amir Khan in action during a title fight at the ExCel Arena in London in February 2008 . Murray and Sears attend the exhibition match held to mark the launch of the new Wimbledon Centre Court roof in May 2009 . Murray and Sears attend a Burberry fashion show alongside Serena Williams (second left) and Sarah Jessica Parker (left) in September 2010 . Murray was the best man for the wedding of his brother Jamie (right) and wife Alejandra (second right) at Cromlix House in October 2010 . Television viewers are well used to the sight of Sears, pictured here at Wimbledon in 2011, showing her emotions during Murray's matches . Murray looks dejected as he and Sears wait for transport after the Brit lost the Wimbledon 2012 final to Roger Federer . There were happier moments just weeks later though as Murray celebrates with Sears after beating Federer to win

Olympic gold in London . Murray then won his first Grand Slam title at the US Open in New York in September 2012 by beating Novak Djokovic in an epic final . Murray and Sears laugh with television host Jimmy Fallon before an appearance on the show following his US Open victory . Murray and Sears pose for photographers as they arrive for a Burberry fashion show during London Fashion Week in September 2012 . Murray leans over to kiss Sears after becoming the first British man in 77 years to win the men‘s singles title at Wimbledon in July 2013 . Murray and Sears pose with the famous trophy during the Wimbledon Champion’s Dinner at a hotel in Park Lane later that evening . Murray and Sears stand outside Buckingham Palace in October 2013 after the British No 1 was awarded an OBE by Prince William . Murray and Sears outside Dunblane High School after the local hero received the Freedom of Stirling at his former school in April 2014 . Murray and Sears watch golf as the couple stroll by the fairways of Ridgewood Country Club in New Jersey during The Barclays in August 2014 . Sears poked fun at the reaction to her foul-mouthed rant during the Australian Open this year by wearing this t-shirt for Murray‘s final.”

A.2 Case study of FSC

We present the case of FSC on HumanEval benchmark. Figure 5 shows the final output of FSC on HumanEval_130 when the input responses are all wrong. The corresponding question is as follows:

```
def tri(n):
    """Everyone knows Fibonacci sequence
        ↪ , it was studied deeply by
        ↪ mathematicians in
    the last couple centuries. However,
        ↪ what people don't know is
        ↪ Tribonacci sequence.
    Tribonacci sequence is defined by
        ↪ the recurrence:
    tri(1) = 3
    tri(n) = 1 + n / 2, if n is even.
    tri(n) = tri(n - 1) + tri(n - 2) +
        ↪ tri(n + 1), if n is odd.
    For example:
    tri(2) = 1 + (2 / 2) = 2
    tri(4) = 3
    tri(3) = tri(2) + tri(1) + tri(4)
           = 2 + 3 + 3 = 8
    You are given a non-negative integer
        ↪  number n, you have to a
        ↪ return a list of the
    first n + 1 numbers of the
        ↪ Tribonacci sequence.
    Examples:
    tri(3) = [1, 3, 2, 8]
    """
```

Response 1

```
def tri(n):
    # Define initial values
    tri_seq = [0, 3, 2]

    # Calculate remaining numbers
    for i in range(3, n + 1):
        if i % 2 == 0:
            tri_seq.append(1 + i / 2)
        else:
            tri_seq.append(tri_seq[i - 1] + tri_seq[i - 2] + tri_seq[i - 3])

    return tri_seq[:n + 1]
```

Response 2

```
def tri(n):
    # initialize the sequence with the base cases
    seq = [0, 3]
    # calculate the rest of the sequence
    for i in range(2, n + 1):
        if i % 2 == 0:
            # if i is even
            seq.append(1 + i / 2)
        else:
            # if i is odd
            seq.append(seq[i - 1] + seq[i - 2] + 2)
    return seq
```

Response 3

```
def tri(n):
    if n < 0:
        return []
    # Initialize the array with base cases
    sequence = [0] * (n + 2)
    sequence[0] = 1
    sequence[1] = 3
    for i in range(2, n + 2):
        if i % 2 == 0: sequence[i] = 1 + i // 2
        else:
            sequence[i] = sequence[i - 1] + sequence[i - 2] + sequence[i + 1]
    return sequence[:n + 1]
```

Response 4

```
def tri(n):
    # initialize the list with first two elements
    tribonacci = [1, 3]

    # calculate the next elements
    for i in range(2, n + 1):
        if i % 2 == 0:
            tribonacci.append(1 + i / 2)
        else:
            tribonacci.append(tribonacci[i - 1] + tribonacci[i - 2] + tribonacci[i - 1])

    return tribonacci
```

Response 5

```
def tri(n):
    if n == 1:
        return [3]
    elif n == 2:
        return [3, 2]
    elif n == 3:
        return [3, 2, 8]
    else:
        tri_list = tri(n - 1)
        if n % 2 == 0: tri_list.append(1 + n / 2)
        else:
            tri_list.append(tri_list[-1] + tri_list[-2] + (1 + (n + 1) / 2))
        return tri_list
```

FSC output

```
def tri(n):
    if n < 0:
        return "Invalid input"
    tribonacci = [1, 3]
    for i in range(2, n + 1):
        if i % 2 == 0:
            tribonacci.append(1 + i / 2)
        else:
            tribonacci.append(tribonacci[i - 1] + tribonacci[i - 2] + tribonacci[i + 1]

    return tribonacci[:n + 1]
```

Figure 5: The input responses and final output of FSC with gpt-3.5-turbo on HumanEval_130. We mark the correct parts in green and the wrong parts in red for each input response.