Hit the Sweet Spot! Span-Level Ensemble for Large Language Models

Yangyifan Xu[1,2], **Jianghao Chen**[1,2], **Junhong Wu**[1,2], **Jiajun Zhang**[1,2,3,4]*
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]Institute of Automation, Chinese Academy of Sciences
[3]Wuhan AI Research, [4]Shanghai Artificial Intelligence Laboratory, Shanghai, China
{xuyangyifan2021, chenjianghao2022, wujunhong2021}@ia.ac.cn, jjzhang@nlpr.ia.ac.cn

Abstract

Ensembling various LLMs to unlock their complementary potential and leverage their individual strengths is highly valuable. Previous studies typically focus on two main paradigms: sample-level and token-level ensembles. Sample-level ensemble methods either select or blend fully generated outputs, which hinders dynamic correction and enhancement of outputs during the generation process. On the other hand, token-level ensemble methods enable real-time correction through fine-grained ensemble at each generation step. However, the information carried by an individual token is quite limited, leading to suboptimal decisions at each step. To address these issues, we propose **SWEETSPAN**, a span-level ensemble method that effectively balances the need for real-time adjustments and the information required for accurate ensemble decisions. Our approach involves two key steps: First, we have each candidate model independently generate candidate spans based on the shared prefix. Second, we calculate perplexity scores to facilitate mutual evaluation among the candidate models and achieve robust span selection by filtering out unfaithful scores. To comprehensively evaluate ensemble methods, we propose a new challenging setting (ensemble models with significant performance gaps) in addition to the standard setting (ensemble the best-performing models) to assess the performance of model ensembles in more realistic scenarios. Experimental results in both standard and challenging settings across various language generation tasks demonstrate the effectiveness, robustness, and versatility of our approach compared with previous ensemble methods.[1]

1 Introduction

Recently, large language models (LLMs) have rapidly developed, leading to the emergence of numerous diverse models (Touvron et al., 2023; Chiang et al., 2023). These LLMs differ in datasets, architectures, and training methodologies, each exhibiting its own strengths and weaknesses (Jiang et al., 2023). Therefore, ensembling these LLMs to unleash their complementary potential and leverage their individual strengths is highly valuable (Jiang et al., 2023; Lu et al., 2023; Shnitzer et al., 2023). Model ensembling has thus become a key focus and an active topic of current LLM research (Lu et al., 2024).

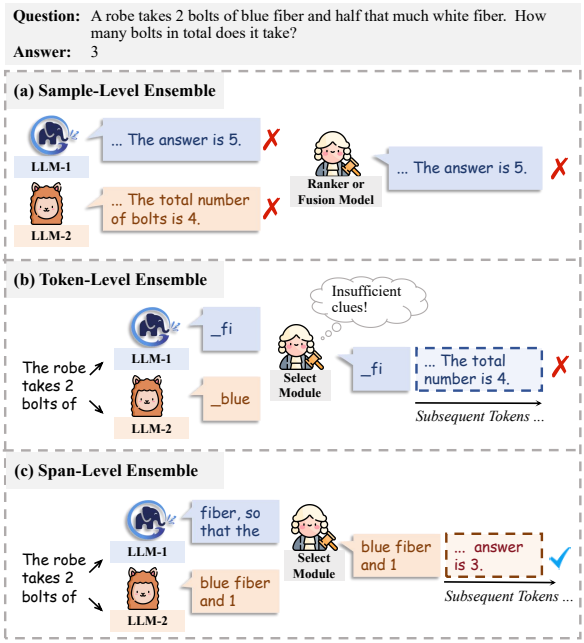

Figure 1: **Motivation of SWEETSPAN.** Sample-level ensemble methods struggle to produce a correct answer when all candidate outputs are flawed, while token-level ensemble methods make suboptimal choices at each generation step due to inadequate information. SWEETSPAN balances the flexibility needed for real-time adjustments and the information required for accurate ensemble decisions at each step.

Existing approaches can be classified into two categories: 1) Sample-level ensemble methods use

* Corresponding Author
[1]Our code is available in https://github.com/xydaytoy/SweetSpan

Proceedings of the 31st International Conference on Computational Linguistics, pages 8314–8325
January 19–24, 2025.

an additional model to either select (Lu et al., 2023; Shnitzer et al., 2023) or blend (Jiang et al., 2023) fully generated outputs. As a result, these methods are unable to correct and enhance outputs during the generation process. As illustrated in Figure 1(a), they struggle to produce a correct answer when all candidate outputs are flawed. Moreover, the reliance on an additional model poses challenges for generalization to unseen data distributions. 2) Token-level ensemble methods achieve fine-grained ensemble at each generation step through output distribution alignment and thus enable real-time correction (Xu et al., 2024; Huang et al., 2024; Yu et al., 2024). However, as shown in Figure 1(b), since tokens are smaller units than words and vary across models, the information they provide is often insufficient and inaccurate. This leads to suboptimal decisions at each step, hindering the achievement of globally optimal outputs.

To tackle the above issues, we propose a training-free span-level ensemble method named **SWEETSPAN**. Our method strikes a balance between the flexibility needed for real-time adjustments and the information required for accurate ensemble decisions at each step, hitting the sweet spot for the model ensemble. Specifically, in each generation round, we first have each candidate model independently generate a span based on the shared prefix. Subsequently, we facilitate mutual evaluation among the candidate models by calculating perplexity scores and filtering out unfaithful results to prevent underperforming models from skewing the evaluation. Finally, we choose the span with the lowest average perplexity as the ensemble result and attach it to the prefix for subsequent generations.

We evaluate our method on various language generation tasks, including commonsense reasoning, arithmetic reasoning, code generation, and machine translation. We first ensemble the best-performing LLMs as a standard setting to assess the upper-bound performance of different ensemble methods. SWEETSPAN consistently achieves significant performance improvements across multiple tasks compared to previous methods, demonstrating its effectiveness and versatility. We then ensemble LLMs with significant performance gaps as a challenging setting to evaluate the noise resistance of ensemble methods. Unlike previous methods that collapse on most tasks, SWEETSPAN, aided by an effective filtering strategy, consistently delivers positive improvements, highlighting its robustness.

Briefly, our contributions can be summarized from the perspectives of effectiveness, robustness, and versatility:

- We propose a span-level ensemble method that balances the flexibility needed for real-time adjustments and the information required for accurate ensemble decisions at each step. Experimental results demonstrate the effectiveness of our method.
- Beyond the standard setting, we introduce a new challenge setting that tests the noise resistance of ensemble methods by ensembling models with significant performance gaps. We observe that previous ensemble methods collapse on most tasks under this setting, while SWEETSPAN achieves stable positive improvements, highlighting its robustness.
- SWEETSPAN is not constrained by vocabulary, model architecture, or task, and can be directly applied to any LLM ensemble without additional parameter training, showcasing strong versatility.

2 Our Method

To achieve span-level ensemble for LLMs, we have to figure out two key issues. First, *how to define spans?* Second, *how to identify the optimal span?* We delve into these issues in the following subsections.

2.1 How to define spans?

As shown on the left side of Figure 2, in each generation round, we first have each candidate model independently generate a text span, which has the following two characteristics: 1) Spans are composed of words rather than tokens. We ensure that spans do not cross word boundaries to prevent subsequent evaluations from being affected by different tokenizers across models. 2) All candidate spans have the same predefined length, such as 4. On the one hand, a longer span provides more information during the span selection phase, increasing the probability of making the optimal choice in the current round. On the other hand, a shorter span allows for more ensemble corrections over multiple rounds, reducing the probability of local errors. Therefore, an appropriate span length is crucial for balancing these competing demands and achieving optimal ensemble results.

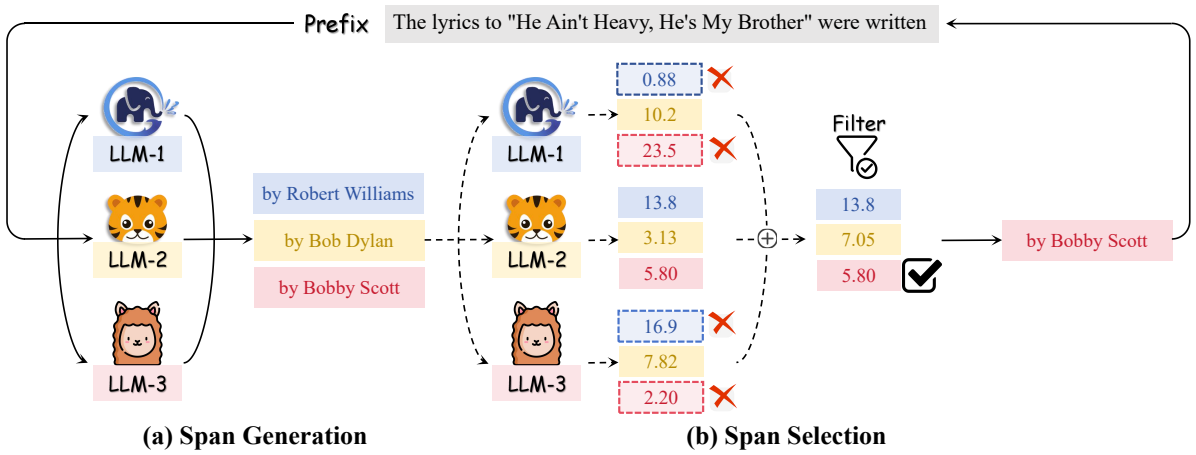

Figure 2: **The SWEETSPAN framework.** SWEETSPAN consists of two steps. (a) First, we have each candidate model generate a span based on the shared prefix. (b) Next, we facilitate mutual evaluation among the candidate models by calculating perplexity and achieve robust span selection by filtering out unfaithful scores.

2.2 How to identify the optimal span?

We choose the simple and widely used perplexity as the evaluation metric for the span selection phase. Perplexity measures how well a language model fits a given span of text. A lower perplexity score indicates that the span aligns more closely with the internal knowledge of the language model, making it a reliable metric for span evaluation.

Considering a set of N candidate models (denoted as $\mathcal{M}$), we first have each candidate model ($m \in \mathcal{M}$) independently calculate the perplexity score for all candidate spans:

$$\mathrm{PPL}_m(\mathbf{s}_j) = \exp\left(-\frac{1}{|\mathbf{s}_j|}\sum_{t\in\mathbf{s}_j}\log p(t)\right),$$

where $\mathbf{s}_j$ is the candidate span generated by the j-th candidate model m_j. Since all spans contain only complete words, vocabulary discrepancies do not affect the perplexity calculation.

Next, we filter the raw evaluation scores. Models lacking relevant knowledge of the current sample may assign unjustifiably perplexity scores, giving correct spans excessively high perplexity while assigning overly low perplexity to their own incorrect spans. As shown in Figure 2, under the current prefix, LLM-1, which fails to answer the question correctly, assigns an unjustifiably high perplexity score of 23.5 to the correct span "*by Bobby Scott*", while giving its own incorrect span "*by Robert Williams*" a low perplexity score of 0.88. As a result, without filtering, the correct span ranks below the incorrect ones, leading to an incorrect output. To address this issue, we design an adaptive filtering strategy that identifies outliers based on the discrepancy in evaluation scores for each span. If the maximum score exceeds the minimum score by a factor of λ, we remove the highest and lowest scores for that span. As shown in Figure 2, by filtering out the maximum value, we prevent LLM-1 from excessively increasing the perplexity for the correct span, thus avoiding an overly negative assessment. By filtering out the minimum value, we avoid LLM-1's overconfidence in the incorrect span. Our proposed filtering method effectively removes unjustified outliers, thereby preventing them from skewing the evaluation.

$$\mathcal{R}_j = \left\{\operatorname*{argmax}_{m\in\mathcal{M}}\mathrm{PPL}_m(\mathbf{s}_j),\ \operatorname*{argmin}_{m\in\mathcal{M}}\mathrm{PPL}_m(\mathbf{s}_j)\right\}$$

$$\mathcal{C}_j = \begin{cases} \mathcal{M}\setminus\mathcal{R}_j, & \text{if } \dfrac{\max\limits_{m\in\mathcal{M}}\mathrm{PPL}_m(\mathbf{s}_j)}{\min\limits_{m\in\mathcal{M}}\mathrm{PPL}_m(\mathbf{s}_j)} > \lambda \\ \mathcal{M}, & \text{otherwise} \end{cases}$$

Finally, we aggregate the filtered perplexity scores and select the span with the lowest average perplexity as the ensemble result for the current generation round.

$$\mathbf{s}^* = \operatorname*{arg\,min}_{\mathbf{s}_j, 1\le j\le N}\frac{1}{|\mathcal{C}_j|}\sum_{m\in\mathcal{C}_j}\mathrm{PPL}_m(\mathbf{s}_j)$$

3 Experimental Settings

3.1 Tasks and Datasets

To demonstrate the versatility of our method, we explore a wide range of language generation tasks:

Commonsense Reasoning via Natural Question (NQ) (Kwiatkowski et al., 2019): Models are tested to answer questions originating from real queries issued to the Google search engine. The evaluation metric is Exact Match.

Arithmetic Reasoning via GSM8K (Cobbe et al., 2021): Models are tasked to solve math problems that require multi-step reasoning. These problems are at the grade school level. The evaluation metric is Accuracy.

Code Generation via MBPP (Austin et al., 2021): Models are requested to generate code that solves basic Python programming problems. These problems are designed to be solvable by entry-level programmers, covering fundamentals and standard library functionality. The evaluation metric is Pass@1.

Machine Translation via Flores-101 (Goyal et al., 2022): Machine translation is a traditional NLP task, which demonstrates the multilinguality of LLMs. We use the German (De)↔English (En) split and Romanian (Ro)↔English split for evaluation. The evaluation metric is BLEU (Post, 2018).

3.2 Candidate LLMs

We choose seven open-source chat LLMs, each approximately 7B in size, as the candidate LLMs to form our ensemble: LLaMA2-7B-Chat (Touvron et al., 2023), ChatGLM2-6B (Zeng et al., 2022), Baichuan2-7B-Chat (Baichuan, 2023), InternLM-7B-Chat (Team, 2023), TigerBot-7B-Chat-V3 (Chen et al., 2023b), Vicuna-7B-V1.5 (Chiang et al., 2023), ChineseAlpaca2-7B (Cui et al., 2023).

Each model leverages large-scale, high-quality data to establish a strong knowledge base and is aligned by supervised instruction tuning, thereby achieving high performance on public benchmarks. Originating from distinct institutions, these models exhibit inherent diversity, which provides opportunities for effective ensemble.

3.3 Baselines

We compare SWEETSPAN with strong baseline methods, including both sample-level and token-level ensemble approaches.

PairRanker Jiang et al. (2023) utilize a pairwise comparison to identify subtle distinctions between different candidate outputs.

LLM-Blender Jiang et al. (2023) employ a 3B-parameter model fine-tuned on an instruction dataset to combine the ranking results from PairRanker and produce the final output.

EVA Xu et al. (2024) utilizes overlapping tokens as anchors to map the output distributions of heterogeneous LLMs into a unified space, enabling a fine-grained token-level ensemble.

3.4 Implement Details

We utilize greedy decoding in all experiments since it generally produces higher-quality outputs (Uzan et al., 2024). Empirically, we set $\lambda = 10$, with a detailed analysis provided in Appendix A. For machine translation tasks, we employ a 4-shot in-context learning setting, while for other tasks, we perform zero-shot inference. Furthermore, we follow the required format for each chat model and use task-specific prompts, including the specific incorporation of a chain of thought prompt in the arithmetic reasoning task.

4 Experimental Results and Analysis

4.1 Standard Setting: Ensembling Top-Performing Models

For each task, we select the top-performing four models out of seven for the ensemble, following a setup similar to previous work (Xu et al., 2024). We conduct experiments with span lengths of 1, 2, 4, 8, 16, and 32, and the trends in ensemble performance are shown in Figure 3.[2]

4.1.1 SWEETSPAN Demonstrates Superiority

Our proposed SWEETSPAN consistently outperforms individual models and prior ensemble methods across all types of tasks at nearly all span lengths, demonstrating the effectiveness and cross-task versatility of our approach. Notably, in the GSM8K task, SWEETSPAN achieves a significant 12.21-point improvement compared with the best-performing individual model. We attribute this success to the appropriate granularity, which effectively balances the competing demands of information richness and flexibility.

4.1.2 Guidelines for Determining Appropriate Span Length

We observe that the optimal span length varies based on the specific characteristics of each task.

[2]Span length refers to the number of words within the span.

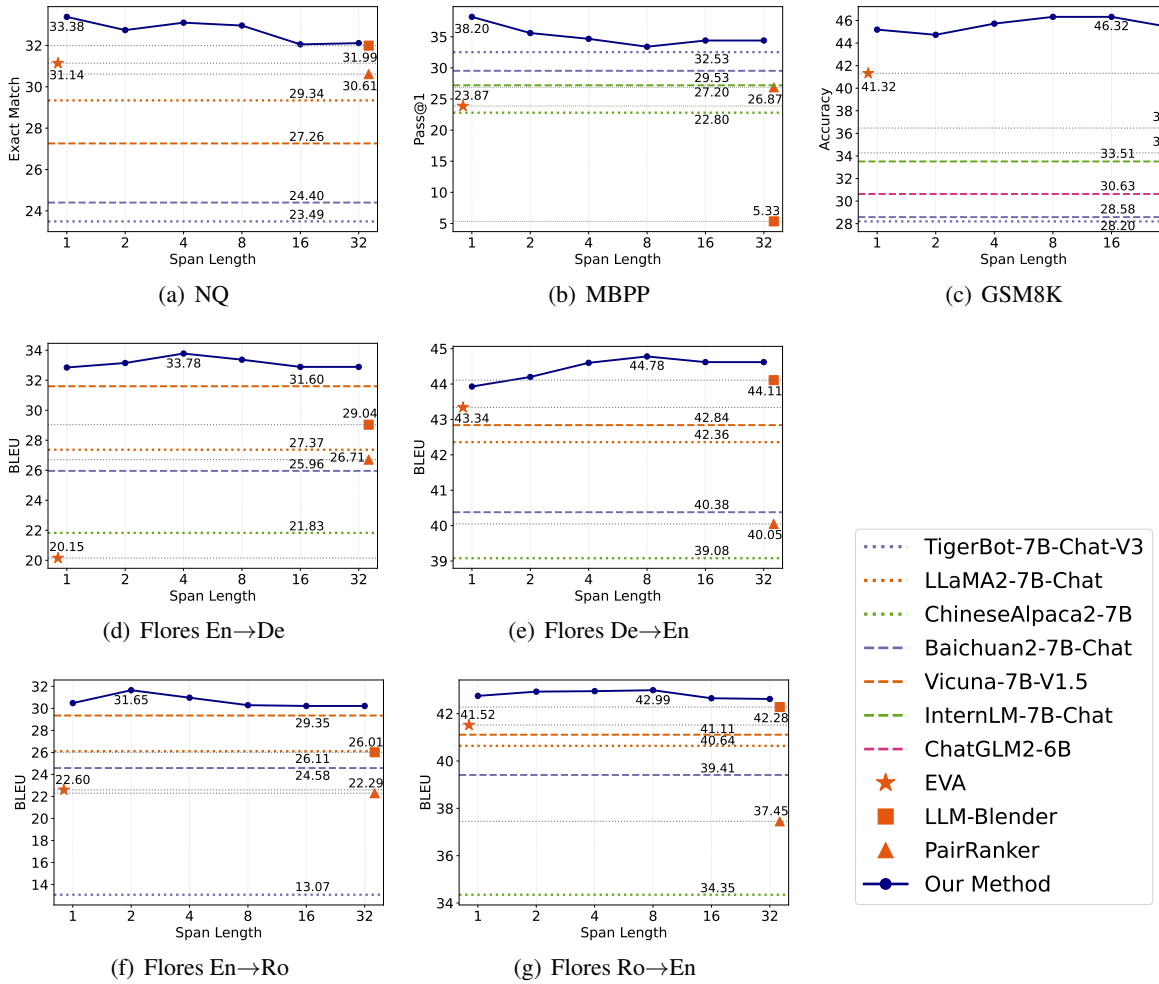

Figure 3: Main results on various language generation tasks under the standard setting.[3]

The following guidelines provide recommendations for selecting an appropriate span length tailored to different tasks:

Deterministic tasks prefer shorter spans. Deterministic tasks are characterized by having precisely defined acceptable answers. NQ, where the answer is a fixed phrase, and MBPP, where the answer is code, are examples of such tasks, both achieving optimal ensemble performance with a span length of 1. As shown in Table 1, for the MBPP task, the strict syntax of code leads to a narrow range of correct answers, where a two-word span is already sufficient to confirm that the second span violates the syntax structure. In this case, longer spans do not provide additional useful information for the current decision and may even hinder timely correction of errors made during independent generation by candidate models.

In contrast, for the GSM8K task, which uses chain-of-thought prompting, intermediate reasoning steps can be expressed in various ways. A two-word span often doesn't provide enough information to make the best decision, and early incorrect choices can cause subsequent tokens to deviate from the intended meaning. In this case, a longer span is necessary to compare different reasoning paths. This aligns with the performance observed in GSM8K, where optimal ensemble performance is achieved with a span length of 16.

An intermediate span length is effective across all tasks. As illustrated in Figure 3, although different tasks exhibit varying trends, an intermediate span length, such as 4, consistently delivers significant improvements over candidate models. The intermediate span length effectively balances the need for real-time adjustments and the information required for accurate ensemble decisions at each

[3]Please refer to appendix C for detailed information on the models used for each task and the experimental results presented in tabular form.

Code Generation Task: MBPP		
Prefix	def sort_matrix(matrix):\n n = len(matrix)\n for	
2-word-candidate spans	i in	| \n i
8-word-candidate spans	i in range(n):\n for j in	| \n i in range(n):\n for j
Arithmetic Reasoning Task: GSM8K		
Prefix	1. Janet's ducks lay 16 eggs per day.	
2-word-candidate spans	She eats	| She bakes
8-word-candidate spans	She eats three for breakfast every morning so	| She bakes muffins with 4 eggs so she

Table 1: Examples of spans with different lengths in GSM8K and MBPP tasks.

ensemble step. We advise adopting a span length of 4 when applying our method to unknown tasks.

4.1.3 SWEETSPAN Avoid the Inherent Limitations of Token-Level and Sample-Level Ensemble Methods

We observe that previous ensemble methods underperform compared to the best candidate model in the Flores En→Ro, Flores En→De, and MBPP tasks. This underperformance stems from the inherent limitations of these methods, specifically:

Token-level ensemble is constrained by insufficient and inaccurate information. Firstly, since tokens, as units smaller than words, carry very limited information, it becomes extremely challenging to determine which option at the current step is most beneficial for the overall outcome. Secondly, token-level ensemble methods rely on overlapping tokens as anchors to align the output distributions from heterogeneous LLMs into the same space, which introduces additional noise. As shown in Figures 3 (d) and (f), the performance of EVA is significantly lower than that of the best candidate model and also falls well short of SWEETSPAN with one-word span. Since tokens in non-English languages like German and Romanian are less familiar to candidate LLMs, the negative impact of noise outweighs the benefits of the ensemble approach. In summary, token-level ensembles with insufficient and inaccurate information lead to suboptimal results at each generation step, hindering the achievement of globally optimal outcomes.

Our proposed SWEETSPAN avoids crossing word boundaries, offering a simple and effective way to eliminate the noise caused by different tokenizations across models. In addition, it ensures that the necessary information is captured through an appropriate span length, resulting in consistent performance improvements across tasks.

Sample-level ensemble is constrained by training data exposure. Sample-level ensemble methods rely on training an additional reward or fusion model to select or combine all candidate answers, which poses significant challenges in generalizing to unseen data distributions. For benchmarks in unfamiliar domains, such as code generation or non-English language translation tasks, their performance can degrade significantly. As shown in Figure 3 (b), LLM-Blender performs very poorly on the code generation task, indicating that its generative fusion model lacks the capability to effectively generate code.

In contrast, our proposed SWEETSPAN is a training-free method, thereby avoiding the generalization issues associated with training data exposure and offering superior generalization across tasks.

4.2 Challenging Setting: Ensembling Models with Substantial Performance Gaps

Previous work focuses on ensembling models with similar performance (Huang et al., 2024). However, in real-world scenarios, it is inevitable to encounter underperforming models in an ensemble when dealing with unknown tasks. To investigate this, we introduce a new challenge setting, ensembling models with significant performance gaps, to evaluate the noise resistance of ensemble methods. Specifically, we perform experiments under two challenging settings[4]: one combining the top three

[4]The scenario most representative of real-world situations involves randomly selecting four models for ensembling. Recognizing the impracticality of exhaustively evaluating every possible combination, we select models with the largest performance gaps to simulate the most challenging scenario among all potential situations.

System	MBPP			GSM8K		
	4 Good	3 Good & 1 Bad	2 Good & 2 Bad	4 Good	3 Good & 1 Bad	2 Good & 2 Bad
LLaMA2-7B-Chat	17.93	17.93	17.93	25.02	25.02	25.02
ChatGLM2-6B	16.60	16.60	16.60	30.63	30.63	30.63
Baichuan2-7B-Chat	29.53	29.53	29.53	28.58	28.58	28.58
InternLM-7B-Chat	27.20	27.20	27.20	33.51	33.51	33.51
TigerBot-7B-Chat-V3	32.53	32.53	32.53	28.20	28.20	28.20
Vicuna-7B-V1.5	-	-	-	18.12	18.12	18.12
ChineseAlpaca2-7B	22.80	22.80	22.80	10.08	10.08	10.08
PairRanker (Jiang et al., 2023)	26.87(- *5.66*)	19.47(*-13.06*)	17.60(*-14.93*)	36.47(+ *2.96*)	33.28(- *0.23*)	28.28(- *5.23*)
LLM-Blender (Jiang et al., 2023)	5.33(*-27.20*)	5.13(*-27.40*)	4.00(*-28.53*)	34.27(+ *0.76*)	29.04(- *4.47*)	28.96(- *4.55*)
EVA (Xu et al., 2024)	23.87(- *8.66*)	28.99(- *3.54*)	23.08(- *9.45*)	41.32(+ *7.81*)	37.15(*+3.64*)	26.08(- *7.43*)
SweetSpan (*ours*)	**38.20(+ *5.67*)**	**37.07(+ *4.54*)**	**35.53(+ *3.00*)**	**46.32(*+12.81*)**	**39.20(*+5.69*)**	**36.32(*+2.81*)**

Table 2: Main results on the code generation task (MBPP) and the arithmetic reasoning task (GSM8K) under the challenge setting. Best results are highlighted with bold and the models employed within the ensemble are distinguished by color according to their performance. Vicuna is excluded from the MBPP task because we find that it cannot generate properly fomatted executable code.

System	NQ			Flores De→En		
	4 Good	3 Good & 1 Bad	2 Good & 2 Bad	4 Good	3 Good & 1 Bad	2 Good & 2 Bad
LLaMA2-7B-Chat	29.34	29.34	29.34	42.36	42.36	42.36
ChatGLM2-6B	14.68	14.68	14.68	34.70	34.70	34.70
Baichuan2-7B-Chat	24.40	24.40	24.40	40.38	40.38	40.38
InternLM-7B-Chat	16.79	16.79	16.79	33.18	33.18	33.18
TigerBot-7B-Chat-V3	23.49	23.49	23.49	36.21	36.21	36.21
Vicuna-7B-V1.5	27.26	27.26	27.26	42.84	42.84	42.84
ChineseAlpaca2-7B	22.83	22.83	22.83	39.08	39.08	39.08
PairRanker (Jiang et al., 2023)	30.61(*+1.27*)	30.69(*+1.35*)	30.55(*+1.21*)	40.05(- *2.79*)	36.82(- *6.02*)	35.24(- *7.60*)
LLM-Blender (Jiang et al., 2023)	31.99(*+2.65*)	31.52(*+2.18*)	**30.89(*+1.55*)**	44.11(*+1.27*)	43.90(*+1.06*)	43.42(*+0.58*)
EVA (Xu et al., 2024)	31.14(*+1.80*)	30.22(*+0.88*)	28.64(- *0.70*)	43.34(*+0.50*)	43.69(*+0.85*)	42.44(- *0.40*)
SweetSpan (*ours*)	**33.38(*+4.04*)**	**32.55(*+3.21*)**	30.30(*+0.96*)	**44.78(*+1.94*)**	**44.82(*+1.98*)**	**43.76(*+0.92*)**

Table 3: Main results on the commonsense reasoning task (NQ) and the machine translation task (Flores De→En) under the challenge setting.

models with the worst model, and another combining the top two with the bottom two models in each task. Experimental results on NQ, GSM8K, MBPP, and Flores-De→En are shown in Table 2 and Table 3.

4.2.1 SweetSpan Demonstrates Robustness

We observe that previous ensemble methods collapse on most tasks under this setting, while our proposed SweetSpan consistently delivers stable performance improvements over individual models and token-level ensemble methods across the four tasks, demonstrating its robustness and versatility. Notably, SweetSpan also outperforms LLM-Blender, which utilizes an additional 3B-parameter fusion model, on three of four tasks, further demonstrating the effectiveness of our approach. We attribute the robustness of our method to the effective filtering strategy that excludes outliers with abnormal perplexity, thereby preventing unfaithful evaluations from impacting the ensemble results.

4.3 Ablation Study on Filtering Strategy

To validate the effectiveness of the filtering strategy in SweetSpan, we conduct ablation studies in both standard and challenging settings. Experimental results on MBPP, GSM8K, NQ, and Flores-De→En are presented in Table 4.

Overall, the filtering strategy in SweetSpan consistently enhances performance across tasks. As the settings become increasingly challenging, the impact of the filtering strategy becomes more pronounced. We observe that the filtering strategy yields modest improvements in tasks with relatively stable performance, such as machine translation. However, for tasks significantly affected by underperforming models, such as MBPP, the strategy results in substantial gains. Notably, when the ensemble includes the two lowest-performing models, it achieves a 10.93% increase in Pass@1 on MBPP compared to using no filtering, highlighting its critical role in maintaining SweetSpan's robustness in challenging scenarios.

Methods	MBPP	GSM8K	NQ	Flores De→En
4 Good				
SweetSpan	38.20	46.32	33.38	44.78
w/o Filter	36.80	46.02	31.39	44.78
3 Good & 1 Bad				
SweetSpan	37.07	39.20	32.55	44.82
w/o Filter	34.00	38.51	30.97	44.79
2 Good & 2 Bad				
SweetSpan	35.53	36.32	30.30	43.76
w/o Filter	24.60	35.33	25.37	43.26

Table 4: Ablation study on filtering strategy

5 Related Work

Ensemble learning is a widely adopted technique to enhance performance on specific tasks and ensure robust generalization by combining multiple complementary systems (Zhou et al., 2017; Liu et al., 2018; Lu et al., 2024). Based on the granularity of the ensemble, existing ensemble methods can be categorized into two categories: sample-level ensemble and token-level ensemble.

Sample-level Ensemble Sample-level ensemble can be further categorized into selection-based ensemble and fusion-based ensemble. Selection-based ensemble methods select the best LLM for specific examples before inference or select the best output from multiple outputs after inference. Shnitzer et al. (2023) uses benchmark datasets to train a router model that selects the best LLM out of a collection of models for a given task. Lu et al. (2023) introduce ZOOTER, a system that uses a reward model to score query-output pairs, then trains a router via knowledge distillation to select the optimal LLM based on input queries. FrugalGPT (Chen et al., 2023a) sequentially calls LLMs until a scoring model deems the output acceptable, efficiently utilizing multiple LLMs. Such methods are limited by the output quality of the candidate models and are unable to generate outputs superior to those of existing models.

Unlike selection-based methods, fusion-based ensembles bypass the limitations of complete outputs, often yielding better results. Jiang et al. (2023) select the top K outputs with a pair ranker, then use a fusion model to combine them. However, this approach relies on the generative capacity of the fusion model, which is limited by its exposure to training data. Moreover, the use of a fusion model considerably increases training and inference costs. For instance, Jiang et al. (2023) utilizes a 3B-sized fusion model.

Our proposed SweetSpan is a training-free approach, avoiding the generalization issues tied to training data exposure and providing superior generalization across tasks.

Token-level Ensemble Token-level ensemble methods combine the output distribution of candidate models at each generation step. Several studies fuse LLMs with specialized models that share the same vocabulary to enhance the specific capabilities of LLMs. Li et al. (2024) combine untrusted LLMs with a benign smaller LLM to mitigate issues such as copyright infringement, data poisoning, and privacy violations. Hoang et al. (2024) enhance translation performance by ensembling a machine translation model with an LLM.

However, most open-source LLMs are heterogeneous and have different vocabularies, hindering direct ensembling. Fu et al. (2023) and Wan et al. (2024) employ exact match constraints and minimum edit distance strategy, respectively, to align vocabularies. Xu et al. (2024), Huang et al. (2024) and Yu et al. (2024) utilize overlapping tokens as anchors to map the output distributions of heterogeneous LLMs into a unified space. Specifically, Xu et al. (2024) propose to directly learn the projection matrices between different vocabularies using the anchors as bridges, while Huang et al. (2024) and Yu et al. (2024) calculate the relative representations from anchors to different vocabularies, thereby indirectly achieving the vocabulary projection. The process of vocabulary alignment introduces noise, which constrains the effectiveness of these methods.

Our proposed SweetSpan does not cross word boundaries, eliminating the noise caused by varying tokenizers.

6 Conclusion

In this paper, we propose a span-level ensemble method, SweetSpan, which balances the flexibility needed for real-time adjustments and the information required for accurate ensemble decisions at each step. Our method has no limitations regarding vocabulary, model architecture, or task, and can be directly applied to any LLM ensemble without additional parameter training, making it a simple and versatile approach. Experimental results in both standard and challenging settings demonstrate the superiority of our approach, which stably and significantly improves overall performance across various natural language processing tasks.

Limitations

Efficiency Due to the inherent nature of ensembling, our approach, like previous ensemble methods, requires performing inference N times when ensembling N models. Additionally, we need to compute perplexity through forward propagation. However, we argue that the inferences for generating candidate spans and computing perplexity can be executed in parallel, respectively, as they are completely independent processes. Notably, our method demonstrates superior efficiency compared to the token-level approach, and the time overhead decreases as the span length increases. Please refer to Appendix B for detailed efficiency analysis.

LLM Evaluation Considering the expenses, we do not use human or GPT-4 evaluation. Human or GPT-4 evaluation could provide us with more reliable and comprehensive results. However, due to the number of models and datasets in our experiments, we cannot afford large-scale human evaluation.

Acknowledgments

This work is supported by National Key R&D Program of China 2022ZD0160602 and the Natural Science Foundation of China 62122088.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023a. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Ye Chen, Wei Cai, Liangmin Wu, Xiaowei Li, Zhanxuan Xin, and Cong Fu. 2023b. Tigerbot: An open multilingual multitask llm. *arXiv preprint arXiv:2312.08688*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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*.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Hieu Hoang, Huda Khayrallah, and Marcin Junczys-Dowmunt. 2024. On-the-fly fusion of large language models and machine translation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 520–532.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. 2024. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. *arXiv preprint arXiv:2404.12715*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Tianlin Li, Qian Liu, Tianyu Pang, Chao Du, Qing Guo, Yang Liu, and Min Lin. 2024. Purifying large language models by ensembling a small language model. *arXiv preprint arXiv:2402.14845*.

Yuchen Liu, Long Zhou, Yining Wang, Yang Zhao, Jiajun Zhang, and Chengqing Zong. 2018. A comparable study on model averaging, ensembling and reranking in nmt. In *Natural Language Processing and Chinese Computing: 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26–30, 2018, Proceedings, Part II 7*, pages 299–308. Springer.

Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. 2024. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. https://github.com/InternLM/InternLM.

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

Omri Uzan, Craig W Schmidt, Chris Tanner, and Yuval Pinter. 2024. Greed is all you need: An evaluation of tokenizer inference methods. *arXiv preprint arXiv:2403.01289*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. 2024. Bridging the gap between different vocabularies for llm ensemble. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7133–7145.

Yao-Ching Yu, Chun-Chih Kuo, Ziqi Ye, Yu-Cheng Chang, and Yueh-Se Li. 2024. Breaking the ceiling of the llm community by treating token generation as a classification for ensembling. *arXiv preprint arXiv:2406.12585*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–384.

A Effect of Model Filtering Intensity

In Section 2.2, we introduced the hyperparameter λ to control the filtering threshold for the evaluation scores. A smaller λ results in more aggressive filtering. For example, when $\lambda = 0$, each span has its highest and lowest scores removed. In contrast, a larger λ results in a more lenient filtering, with filtering bypassed as λ becomes very large. We conduct experiments on the NQ and MBPP tasks to evaluate the effect of different λ values on the ensemble results. Since the optimal λ varies across tasks, we select $\lambda = 10$ as a balanced option for all experiments and observe consistent performance across various tasks.

λ	NQ	MBPP
0	33.05	40.93
10	33.38	38.20
20	33.30	37.53
40	33.38	37.80

Table 5: Effect of Model Filtering Intensity.

B Efficiency Analysis

The time overhead of ensemble methods during the generation process, such as span-level ensemble and token-level ensemble, is related to the length of the generated text. The longer the generated text, the greater the time overhead. As shown in Table 6, we compare the efficiency of SWEETSPAN with the token-level ensemble method EVA by calculating the average extra time required per token. Our method shows superiority in efficiency compared to the token-level approach, and the time overhead decreases as the span length increases.

Method	Average Extra Time Per Token
EVA	0.0777s
Span-1	0.0512s
Span-2	0.0224s
Span-4	0.0128s
Span-8	0.0079s
Span-16	0.0053s
Span-32	0.0077s

Table 6: Efficiency Comparison of Token-level and Span-level Ensembles.

C Main Results under the Standard Setting

The main results on various language generation tasks under the standard setting are presented in Table 7 and Table 8.

System	NQ	GSM8K	MBPP
LLaMA2-7B-Chat	29.34	25.02	17.93
ChatGLM2-6B	14.68	30.63	16.60
Baichuan2-7B-Chat	24.40	28.58	29.53
InternLM-7B-Chat	16.79	33.51	27.20
TigerBot-7B-Chat-V3	23.49	28.20	32.53
Vicuna-7B-V1.5	27.26	18.12	-
ChineseAlpaca2-7B	22.83	10.08	22.80
PairRanker	30.61(+*1.27*)	36.47(+*2.96*)	26.87(-*5.66*)
LLM-Blender	31.99(+*2.65*)	34.27(+*0.76*)	5.33(-*27.20*)
EVA	31.14(+*1.80*)	41.32(+*7.81*)	23.87(-*8.66*)
Span-1	**33.38(+*4.04*)**	45.19(+*11.68*)	**38.20(+*5.67*)**
Span-2	32.74(+*3.40*)	44.73(+*11.22*)	35.60(+*3.07*)
Span-4	33.10(+*3.76*)	45.72(+*12.21*)	34.67(+*2.14*)
Span-8	32.96(+*3.62*)	46.32(+*12.81*)	33.40(+*0.87*)
Span-16	32.05(+*2.71*)	**46.32(+*12.81*)**	34.40(+*1.87*)
Span-32	32.11(+*2.77*)	45.41(+*11.90*)	34.40(+*1.87*)

Table 7: Main results of NQ (measured by Exact Match), GSM8K (measured by Accuracy) and MBPP (measured by Pass@1). Best results are highlighted with bold and the model employed within the ensemble is underlined for distinction.

System	Flores-En→De	Flores-De→En	Flores-En→Ro	Flores-Ro→En
LLaMA2-7B-Chat	27.37	42.36	26.11	40.64
ChatGLM2-6B	13.20	34.70	12.44	31.30
Baichuan2-7B-Chat	25.96	40.38	24.58	39.41
InternLM-7B-Chat	9.20	33.18	10.03	30.23
TigerBot-7B-Chat-V3	20.78	36.21	13.07	33.32
Vicuna-7B-V1.5	31.60	42.84	29.35	41.11
ChineseAlpaca2-7B	21.83	39.08	6.96	34.35
PairRanker	26.71(-*4.89*)	40.05(-*2.79*)	22.29(-*7.06*)	37.45(-*3.66*)
LLM-Blender	29.04(-*2.56*)	44.11(+*1.27*)	26.01(-*3.34*)	42.28(+*1.17*)
EVA	20.15(-*11.45*)	43.34(+*0.50*)	22.60(-*6.75*)	41.52(+*0.41*)
Span-1	32.85(+*1.25*)	43.93(+*1.09*)	30.48(+*1.13*)	42.75(+*1.64*)
Span-2	33.15(+*1.55*)	44.20(+*1.36*)	**31.65(+*2.30*)**	42.93(+*1.82*)
Span-4	**33.78(+*2.18*)**	44.60(+*1.76*)	30.97(+*1.62*)	42.95(+*1.84*)
Span-8	33.37(+*1.77*)	**44.78(+*1.94*)**	30.29(+*0.94*)	**42.99(+*1.88*)**
Span-16	32.89(+*1.29*)	44.62(+*1.78*)	30.22(+*0.87*)	42.65(+*1.54*)
Span-32	32.89(+*1.29*)	44.62(+*1.78*)	30.22(+*0.87*)	42.62(+*1.51*)

Table 8: Main results of machine translation tasks (measured by BLEU). Best results are highlighted in bold and the model employed within the ensemble is underlined for distinction.