AdaFuse: Adaptive Ensemble Decoding with Test-Time Scaling for LLMs

Chengming Cui[1†] **Tianxin Wei**[1†] **Ziyi Chen**[1] **Ruizhong Qiu**[1] **Zhichen Zeng**[1]
Zhining Liu[1] **Xuying Ning**[1] **Duo Zhou**[1] **Jingrui He**[1]
[†]Equal contribution
[1]University of Illinois Urbana-Champaign
{ccui12, twei10, jingrui}@illinois.edu

Abstract

Large language models (LLMs) exhibit complementary strengths arising from differences in pretraining data, model architectures, and decoding behaviors. Inference-time ensembling provides a practical way to combine these capabilities without retraining. However, existing ensemble approaches suffer from fundamental limitations. Most rely on fixed fusion granularity, which lacks the flexibility required for mid-generation adaptation and fails to adapt to different generation characteristics across tasks. To address these challenges, we propose ADAFUSE, an adaptive ensemble decoding framework that dynamically selects semantically appropriate fusion units during generation. Rather than committing to a fixed granularity, ADAFUSE adjusts fusion behavior on the fly based on the decoding context, with words serving as basic building blocks for alignment. To be specific, we introduce an uncertainty-based criterion to decide whether to apply ensembling at each decoding step. Under confident decoding states, the model continues generation directly. In less certain states, ADAFUSE invokes a diversity-aware scaling strategy to explore alternative candidate continuations and inform ensemble decisions. This design establishes a synergistic interaction between adaptive ensembling and test-time scaling, where ensemble decisions guide targeted exploration, and the resulting diversity in turn strengthens ensemble quality. Experiments on open-domain QA, arithmetic reasoning, and machine translation demonstrate that ADAFUSE consistently outperforms strong ensemble baselines, achieving an average relative improvement of 6.88%. The code is available at https://github.com/CCM0111/AdaFuse.

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a wide range of natural language processing tasks (Hendrycks et al., 2020; Cobbe et al., 2021; Ning et al.; Wei et al., 2025; Ai et al., 2025; Zhang et al., 2025; Chen et al., 2024) and are increasingly deployed in real-world applications (Grattafiori et al., 2024; OpenAI, 2024). However, model performance is not uniform across tasks: differences in pretraining data, model architectures, and decoding strategies lead individual models to exhibit heterogeneous strengths(Chu et al., 2025). Models that excel at structured or multi-step reasoning may perform less reliably on open-domain or commonsense question answering, while models optimized for such queries may struggle with complex reasoning (Joshi et al., 2017; Kwiatkowski et al., 2019; Huang and Chang, 2023; Guo et al., 2025). As a result, inference-time ensembling has emerged as a practical approach for improving robustness by combining complementary model strengths without retraining (Jiang et al., 2023b; Liu et al., 2024; Yao et al., 2024; Wang et al., 2023b).

Existing inference-time ensemble methods differ primarily in the granularity at which model outputs are fused (Chen et al., 2023). Sample-level approaches combine complete model responses via reranking or response fusion (Jiang et al., 2023b; Shnitzer et al., 2023; Lu et al., 2023; Wang et al., 2023a), but perform fusion only after generation has finished, preventing any mid-generation correction. Span-level methods assemble multi-token segments produced by different models (Liu et al., 2024; Xu et al., 2024; Lv et al., 2024; Fu et al., 2025), yet rely on predefined or fixed span boundaries that limit flexibility across tasks and reasoning depths. Token-level approaches aggregate next-token distributions at each decoding step (Yao et al., 2024; Huang et al., 2024; Yu et al., 2024; Zeng et al., 2025), but typically require token alignment across models, restricting applicability under heterogeneous tokenizers; related byte-level alternatives alleviate this mismatch at the cost of increased computation and weakened semantic structure (Phan et al., 2024; Sathe et al., 2024). Despite

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 42644–42657

operating at different granularities, most existing methods rely on a *fixed* fusion resolution, which restricts their ability to adapt the fusion scope to evolving semantic context, task demands, and reasoning uncertainty during generation (Yao et al., 2023).

These trade-offs motivate the need for an adaptive inference-time ensemble framework that overcomes the limitations imposed by fixed-granularity generation. We adopt word-level units as the basic building blocks, as they provide a natural abstraction that supports step-wise integration and semantic coherence, while avoiding explicit token alignment across heterogeneous tokenizers. Based on this design choice, we propose ADAFUSE, an adaptive word-level ensembling framework that enables flexible inference-time fusion and mid-generation correction at natural word boundaries, which preserves semantic integrity across models.

Concretely, ADAFUSE adopts an adaptive decoding strategy that balances effectiveness and efficiency during generation. We introduce an uncertainty-based criterion to decide whether to apply ensembling at each decoding step. Under confident decoding states, the model proceeds with direct generation. In less certain states, ADAFUSE invokes a diversity-aware scaling strategy to explore alternative candidate continuations and support more effective ensemble decisions. This design is guided by the evolving decoding context rather than predefined rules, and establishes a synergistic interaction between adaptive ensembling and test-time scaling. In particular, selective exploration under uncertainty improves test-time scaling effectiveness, while the resulting diversity further strengthens ensemble quality.

In summary, our paper contributes the following:

- **Confidence-Guided Adaptive Decoding**: ADAFUSE uses model confidence to decide when to commit longer word spans and when to consider more candidate continuations, enabling adaptive mid-generation correction without relying on fixed-length spans.
- **Diversity-Aware Ensemble Scaling**: Building on confidence-guided decoding, ADAFUSE employs an exploration strategy to generate diverse candidate continuations for ensemble scoring only when uncertainty arises, while avoiding unnecessary computation in confident states.
- **Strong Empirical Performance Across Diverse Tasks**: ADAFUSE achieves consistent gains over the strongest ensemble baselines, with an average relative improvement of 6.88% across open-domain question answering, arithmetic reasoning, and machine translation benchmarks.

2 Related Works

Ensemble methods in LLMs have evolved significantly, with prominent approaches categorized into three main levels: *Sample-level, Span-level, Token-level ensembling*, each designed to harness complementary model strengths yet plagued by its own drawbacks.

Sample-Level Ensembling aggregates entire generated responses from multiple LLMs. LLM-Blender (Jiang et al., 2023b) trains a *pairwise ranker* to compare and select the strongest candidate, then applies a *generative fusion* module to produce a final answer. Routing methods (Shnitzer et al., 2023; Lu et al., 2023) assign each query to the most suitable expert model, but their effectiveness is capped when every candidate output contains errors (Hu et al., 2024). Fusion networks (Wang et al., 2023a) go one step further by learning to merge several complete outputs into a single consolidated response, boosting overall quality but still facing challenges in generalizing across diverse tasks and overlooking valuable token-level probability cues generated during decoding (Lin et al., 2025; Li et al., 2025).

Span-Level Ensembling combines output segments. Cool-Fusion (Liu et al., 2024) merges model outputs once they reach common word boundaries, while SweetSpan (Xu et al., 2024) generates and merges spans based on perplexity scores. SpecFuse (Lv et al., 2024) aggregates multiple LLM outputs by predicting the next segment and combining them. RLAE (Fu et al., 2025) uses reinforcement learning to adjust weights dynamically during the merging process. While span-level methods improve coherence and fluency, they rely on the quality of the generated segments, making them susceptible to errors in weak segments. Furthermore, span-level methods offer limited flexibility due to their fixed fusion granularity and lack of adaptive control, making them less effective on novel or diverse inputs.

Token-Level Ensembling merges each model's next-token distributions at every decoding step. UniTE (Yao et al., 2024) forms the union of each

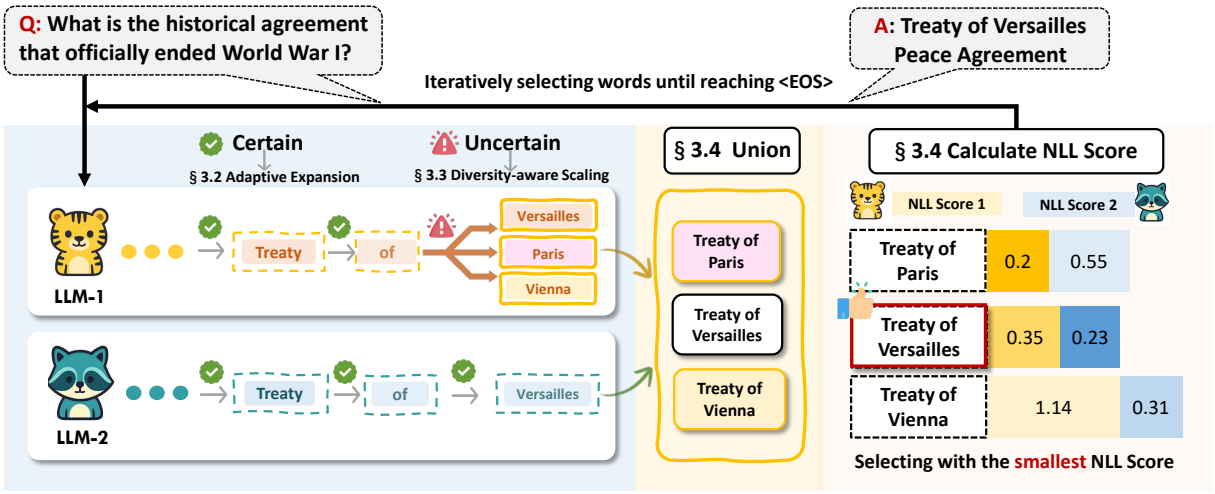

Figure 1: Illustration of the AdaFuse framework. AdaFuse ensembles multiple LLMs during decoding by dynamically adjusting fusion behavior. It expands in confident decoding states and applies diversity-aware scaling under uncertainty. Candidate word sequences are unified and selected using NLL scores, enabling adaptive and stable generation.

model's top-k token candidates at each step, avoiding full-vocabulary alignment and cutting inference cost. DeePEn (Huang et al., 2024) projects each model's logits into a universal relative space via anchor-token similarity, aggregates them, and applies a search-based inverse transformation to recover logits in a main model's vocabulary, enabling training-free fusion of heterogeneous LLMs. GAC (Yu et al., 2024) builds dense mapping matrices over the union vocabulary, projects each model's probability vector into this joint space, and averages them to select the next token, optionally cascading to cheaper ensembling when confidence is high. Despite their effectiveness, token-level approaches operate over large vocabularies at every step, leading to substantial computational overhead. They also struggle with heterogeneous tokenizations and may overlook low-probability yet semantically important tokens. Byte-level ensembling addresses vocabulary mismatch by merging distributions at the byte granularity (Phan et al., 2024), but introduces constant per-step conversion overhead and obscures word-level semantic structure. These limitations motivate our adaptive ensembling framework, which performs confidence-guided fusion at natural word boundaries, enabling precise step-wise integration with reduced overhead and improved semantic alignment.

3 Methodology

We present AdaFuse, an adaptive ensemble decoding framework that addresses a core limitation of existing inference-time ensembling methods: their reliance on fixed fusion granularity and rigid decoding control, which prevents fusion behavior from adapting to evolving uncertainty.

At each decoding step, AdaFuse adaptively generates word-level candidate segments from each model, regulates the breadth of candidate consideration based on model confidence, and applies joint scoring across models to select the most appropriate continuation. This design enables real-time integration of ensemble preferences at a linguistically meaningful granularity, while flexibly balancing robustness and efficiency during decoding. An overview is shown in Figure 1.

3.1 Candidate Word Proposal

AdaFuse generates candidate word completions from each model by extending the current decoding prefix until a word boundary is reached. Starting from the current prefix $\mathbf{y}_{1:t}$, each model m_k generates tokens sequentially according to its next-token distribution until a complete word is formed. The resulting token sequence is decoded into a word segment, denoted as $\mathbf{w}$. This design avoids mid-word truncation and provides a consistent unit for alignment and scoring across models during ensemble decoding.

3.2 Adaptive Word Commitment

In word-level ensemble decoding, we need an effective rule to decide whether the model is sufficiently confident to continue generating a longer span or to stop and re-score candidate segments. Because a full word or phrase typically spans multiple sub-

word tokens, its confidence cannot be reliably assessed before generation completes. We therefore use the first-token signal as a practical proxy for how ready the model is to commit to extending the current word.

Start-of-Word Confidence. We assess confidence using the margin between the top-ranked candidates in the first-token distribution at the start of a word. Specifically, let $p_{(1)}$ and $p_{(2)}$ be the top-1 and top-2 probabilities of the *first token* at the start of a word. We define the confidence margin as $\Delta_{m_k}(\mathbf{y}) = p_{(1)} - p_{(2)}$.

A larger margin corresponds to higher confidence, allowing the decoder to continue generating a longer word span, whereas a smaller margin indicates uncertainty and triggers early re-scoring.

Adaptive Expansion Strategy. Based on the confidence signal defined above, ADAFUSE adaptively decides whether to extend generation beyond the current word or to stop and perform ensemble re-scoring. For model m_k at prefix $\mathbf{y}$, we allow continued generation only when the margin criterion exceeds a predefined threshold τ_Δ:

$$\Delta_{m_k}(\mathbf{y}) \geq \tau_\Delta. \tag{1}$$

When this condition is satisfied, the decoder proceeds to continue generating the current word using greedy decoding, treating the current continuation as a confident span extension. Otherwise, generation is halted after the current word and candidate segments are returned for cross-model scoring.

To avoid over-commitment while preserving step-wise correction, we limit each decoding round to at most M generated words. In practice, we set $M = 3$ in all experiments, which is sufficient to capture most compact semantic units (e.g., named entities, short phrases, and collocations), while preventing error accumulation and delayed cross-model feedback.

Overall, this adaptive rule maintains the efficiency of greedy decoding while adjusting the degree of commitment based on model confidence, striking a practical balance between accuracy, fluency, and computational cost.

3.3 Diversity-Aware Ensemble Scaling

Building on the adaptive word-length expansion strategy, ADAFUSE explores the role of lexical diversity in word-level ensemble decoding. To balance diversity and efficiency, we adopt an adaptive mechanism that expands the candidate space only when additional exploration is warranted, rather than enforcing uniform diversification. Since beam search operates at the token level and prioritizes high-probability continuations, directly applying it at the word level often yields redundant candidates with limited effective diversity. To address this mismatch, ADAFUSE employs a two-stage word-level search that separates *exploration* and *exploitation*, enabling controlled diversity while preserving fluency.

Adaptive Trigger. Diversity is introduced only when the first-token distribution at the current decoding prefix $\mathbf{y}$ fails to satisfy the confidence criterion in Eq. (1), indicating elevated uncertainty in initiating a new word. In such cases, ADAFUSE activates the two-stage diversity search to enumerate multiple plausible word-level continuations. When the criterion is satisfied, decoding proceeds without diversification, ensuring that diversity is applied only when beneficial while maintaining computational efficiency.

Exploration. In the first stage, we select the top-B *distinct* initial tokens based on their conditional likelihood given the current decoding prefix $\mathbf{y}$. We refer to B as the *branching factor*, which determines how many paths are explored per decoding step to encourage intra-round diversity:

$$\{z_1^{(1)}, \ldots, z_1^{(B)}\} = \text{Top-}B\left(p(v \mid \mathbf{y})\right), \tag{2}$$

Each $z_1^{(b)}$ serves as a distinct lexical entry point for the generation of candidate words.

Exploitation. Each selected initial token $z_1^{(b)}$ is then greedily extended autoregressively to form a complete word:

$$z_i^{(b)} = \arg\max_v \; p_{m_k}\left(v \mid \mathbf{y}, z_1^{(b)}, \ldots, z_{i-1}^{(b)}\right), \quad i = 2, 3, \ldots \tag{3}$$

Decoding continues until a whitespace character is produced, at which point we obtain the complete word sequence candidates: $\mathbf{w}^{(b)} = (z_1^{(b)}, \ldots, z_L^{(b)})$, where L denotes the position at which decoding yields a full word. This two-stage procedure produces diverse yet fluent candidate words for subsequent ensemble scoring.

3.4 Cross-Model Scoring and Candidate Integration

The final step in ADAFUSE is to select the most suitable continuation from a pooled set of word-level

span candidates constructed across models. Within each decoding round, each model generates a span candidate by concatenating the words it commits in that round. We denote the b-th span candidate as $\mathbf{s}^{(b)} = \left(\mathbf{w}_1^{(b)}, \ldots, \mathbf{w}_c^{(b)}\right)$, where $c \leq M$ is the adaptive number of committed words in the round. When the confidence criterion remains satisfied throughout the round, the model produces a single span candidate. Otherwise, diversity-aware ensemble scaling is triggered to enumerate B alternative span candidates: $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(B)}$.

ADAFUSE forms a candidate pool S by taking the *union* of $\mathbf{s}^{(b)}$ generated by all participating models. To compare candidates fairly, we compute the normalized negative log-likelihood (NLL) of each word as its average token-level surprise. NLL reflects model confidence over each candidate word, enables fair comparison across variable-length segments, and supports stable aggregation across models.

For each model m_k, we compute the normalized negative log-likelihood (NLL) of a span candidate $\mathbf{s}^{(b)}$ as the average token-level surprise:

$$\mathrm{NLL}_{m_k}(\mathbf{s}^{(b)}) = -\frac{1}{T_b} \sum_{t=1}^{T_b} \log p_{m_k}\left(z_t^{(b)} \mid \mathbf{y}, z_{<t}^{(b)}\right), \tag{4}$$

where $(z_1^{(b)}, \ldots, z_{T_b}^{(b)})$ denotes the token sequence obtained by concatenating all tokens in the span $\mathbf{s}^{(b)}$, and T_b is the total number of tokens in the span.

We then average these scores across the ensemble of K models to obtain the fusion score:

$$F(\mathbf{s}^{(b)}) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{NLL}_{m_k}(\mathbf{s}^{(b)}), \tag{5}$$

Finally, we select the best candidate by minimizing the fusion score:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}^{(b)}} F(\mathbf{s}^{(b)}), \tag{6}$$

and append it to the decoding prefix:

$$\mathbf{y} \leftarrow \mathbf{y} \| \mathbf{s}^*. \tag{7}$$

This scoring mechanism enables ADAFUSE to act as a dynamic feedback system, selecting candidates that are not only contextually appropriate for the current prefix but also jointly supported across diverse model predictions. The complete procedure is outlined in Algorithm 1.

Algorithm 1 ADAFUSE Decoding

Require: Prompt P, models $\{m_k\}_{k=1}^{K}$, margin threshold τ_Δ, max words M, branching factor B

$\mathbf{y} \leftarrow P$ ▷ Committed decoding prefix
while not Terminate **do** ▷ until EOS or length limit
  $S \leftarrow \emptyset$ ▷ Candidate span pool
  for each model m_k **do**
    $\mathbf{s} \leftarrow \emptyset, c \leftarrow 0$
    while $c < M$ **do**
      $\mathbf{w} \leftarrow \textsc{GenWord}(m_k, \mathbf{y} \| \mathbf{s})$ ▷ § 3.1
      if $\Delta_{m_k}(\mathbf{y} \| \mathbf{s}) \geq \tau_\Delta$ **then** ▷ § 3.2
        $\mathbf{s} \leftarrow \mathbf{s} \| \mathbf{w}, c \leftarrow c + 1$
      else ▷ § 3.3
        $\{\mathbf{w}^{(b)}\}_{b=1}^{B} \leftarrow \textsc{GenWord}(m_k, \mathbf{y} \| \mathbf{s}, B)$
        $\mathbf{s} \leftarrow \mathbf{s} \| \{\mathbf{w}^{(b)}\}_{b=1}^{B}$
        break
      end if
    end while
    $S \leftarrow S \cup \{\mathbf{s}\}$
  end for
  for each candidate $\mathbf{s} \in S$ **do** ▷ § 3.4
    $F(\mathbf{s}) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mathrm{NLL}_{m_k}(\mathbf{s})$
  end for
  $\mathbf{s}^* \leftarrow \arg\min_{\mathbf{s} \in S} F(\mathbf{s})$
  $\mathbf{y} \leftarrow \mathbf{y} \| \mathbf{s}^*$
end while
return $\mathbf{y}$

4 Experiments

In this section, we conduct a series of experiments to systematically evaluate the effectiveness, and design decisions of ADAFUSE. Specifically, we aim to answer the following research questions:

- **RQ1**: Does ADAFUSE consistently enhance the performance of base models across a wide range of tasks?
- **RQ2**: How does the adaptive word-commitment mechanism in ADAFUSE affect the effectiveness of word-level ensemble decoding?
- **RQ3**: How does incorporating diversity-aware ensemble scaling in ADAFUSE influence overall performance as the number of candidates increases?
- **RQ4**: How does ADAFUSE balance efficiency and effectiveness, and what qualitative behaviors can be observed from example analyses?

4.1 Setup

Models. We conduct our experiments using four recent and widely adopted open-source chat and instruction-tuned models:LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023a), Qwen3-8B (Yang et al., 2025), and InternLM3-8B-Instruct (Cai et al., 2024). These models cover diverse, practical size regimes and represent the latest publicly released versions available during our experiments.

Table 1: Evaluation results on six benchmarks for base models and LLM ensemble methods. Baseline ensemble methods use a fixed LLaMA-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3 pair. ADAFUSE is evaluated with both a fixed base pair and an oracle top-2 base selection. The overall best result is shown in **bold**, while the best result among baseline ensemble methods is underlined. The improvement row reports the relative percentage gains of ADAFUSE over the best-performing baseline method.

Model	NQ	SQuAD	TriviaQA	GSM8K	Flores En→De	Flores De→En	Avg
Base Models							
Mistral-7B-Instruct-v0.3	32.05	83.49	77.27	61.71	29.99	40.64	54.19
LLaMA-3.1-8B-Instruct	34.24	80.13	78.97	81.05	31.33	41.78	57.92
Qwen3-8B	23.85	76.72	64.72	89.99	17.45	36.95	51.61
InternLM3-8B-Instruct	27.15	81.77	65.38	76.72	28.17	37.37	52.76
LLM Ensemble Methods							
LLM-BLENDER	36.56	82.13	75.58	62.55	34.45	41.56	54.97
DEEPEN	37.42	71.51	73.85	67.63	37.56	42.33	55.05
SWEETSPAN	37.59	86.58	80.75	63.38	33.74	42.85	59.16
UniTE	38.95	82.17	81.38	64.59	34.98	41.67	57.29
Our Ensemble Method							
ADAFUSE (Top-2 Base)	**42.85**	**90.38**	**82.17**	**90.25**	**39.83**	**45.25**	**65.12**
ADAFUSE (Fixed Base)	**42.85** (+10.01%)	90.15 (+4.12%)	**82.17** (+0.97%)	79.15 (+17.03%)	**39.83** (+6.04%)	**45.25** (+5.60%)	63.23 (+6.88%)

Baselines. We compare our method against four representative ensembling baselines. 1) **LLM-BLENDER** (Jiang et al., 2023b) uses a reward model (`PairRanker`) and a fusion model (`GenFuser`) to rerank and merge responses. 2) **DEEPEN**(Huang et al., 2024) projects candidate outputs into a shared latent space using vocabulary alignment and relative representation modeling. 3) **SWEETSPAN** (Xu et al., 2024) is a span-level ensemble method that selects from model-generated span candidates via perplexity-based scoring. 4) **UNITE** (Yao et al., 2024) applies a top-k union filtering strategy for robust vocabulary alignment and token selection.

Benchmarks. We evaluate six benchmarks across three task categories. 1) **Knowledge-intensive QA:** `NaturalQuestions` (5-shot) (Kwiatkowski et al., 2019), a large-scale open-domain QA dataset built from real Google search queries; `SQuAD` (5-shot) (Rajpurkar et al., 2016), reading-comprehension questions over Wikipedia passages; and `TriviaQA` (5-shot) (Joshi et al., 2017), trivia questions paired with evidence documents harvested from the web. 2) **Arithmetic Reasoning:** `GSM8K` (4-shot with chain-of-thought prompting) (Cobbe et al., 2021), grade-school math word problems requiring multi-step reasoning. 3) **Machine Translation:** `FLores` (Goyal et al., 2022) English(En)→German(De) (0-shot) and German(De)→English(En) (0-shot), covering translation quality on both high- and low-resource language pairs.

Experimental Setup. For the main two-model ensembling setup, we choose Mistral-7B-Instruct-v0.3 and LLaMA-3.1-8B-Instruct as our fixed pair, given their strong, complementary performance profiles. We also include Qwen3-8B and InternLM3-8B-Instruct to test scalability and cross-architecture robustness. In our experiments, the confidence threshold is set to $\tau_\Delta = 0.7$ (see Appendix D.2 for hyperparameter analysis). Within each decoding round, we cap the maximum number of consecutively committed words at $M = 3$, and bound the total context length by a task-dependent limit of up to 512 tokens. Performance is evaluated using standard metrics for each task: exact match accuracy on NQ, SQuAD, and TriviaQA, accuracy on GSM8K, and spBLEU for machine translation. Unless otherwise specified, Section 3.3 is disabled in the main experiments to ensure a fair comparison under comparable computational budgets, and is examined separately in the ablation study. We additionally examine how ADAFUSE scales with the number of base models by expanding the ensemble beyond the primary two-model setup; detailed results are reported in Appendix E.

4.2 Main Results (RQ1)

As shown in Table 1, ADAFUSE achieves an average score of 63.23 across six benchmarks, outperforming the strongest prior ensemble method (SWEETSPAN, 59.16) by 4.07 points (6.88% rela-

tive improvement).

ADAFUSE delivers particularly strong gains on knowledge-intensive QA tasks. Compared to the single model LLaMA-3.1-8B-Instruct, it improves performance by +8.61 pts on NQ, +10.25 pts on SQuAD, and +3.20 pts on TriviaQA. It also consistently outperforms all competing ensemble methods on both translation benchmarks, demonstrating robust cross-task generalization.

On the arithmetic-reasoning benchmark GSM8K, ADAFUSE attains an accuracy of 79.15, slightly below LLaMA-3.1-8B-Instruct (81.05). We attribute this gap to the large performance disparity between the underlying base models. When ensembling the two strongest base models on GSM8K, **Qwen3-8B** and **LLaMA-3.1-8B-Instruct**, ADAFUSE achieves 90.25 accuracy, confirming that base-model compatibility plays a critical role in ensemble effectiveness. Notably, ADAFUSE still surpasses all other baseline ensemble methods on GSM8K by at least +11.52 pts.

We further analyze performance by fusion granularity to contextualize these gains. Sample-level ensembling (LLM-BLENDER) provides only modest improvements (e.g., +2.32 pts on NQ and +3.12 pts on Flores En $\rightarrow$ De), indicating limited robustness when candidate quality is uniformly low. Span-level methods (SWEETSPAN) achieve strong gains on extractive QA (+6.45 pts on SQuAD), but yield smaller improvements on open-domain QA and translation. Token-level approaches (UNITE) further improve open-domain QA (up to +4.71 pts on NQ), yet underperform on translation and incur substantially higher computational overhead. In contrast, ADAFUSE consistently delivers the largest gains across tasks (e.g., +8.61 pts on NQ and +8.50 pts on Flores En $\rightarrow$ De), striking a favorable balance between accuracy, robustness, and efficiency.

4.3 Ablation Study (RQ2)

To assess the importance of adaptive word commitment in ADAFUSE, we conduct an ablation study in which we replace the adaptive word commitment with fixed-length word commitment, while keeping all other components unchanged.

Analysis of Adaptive Word Commitment. We study the role of adaptive word commitment on the Natural Questions dataset using ADAFUSE with the main fixed base-model pair. As shown in Figure 2, fixed-length decoding with word lengths of 1, 2, or 3 consistently underperforms adaptive ADAFUSE. This is because fixed-length strategies tend to fragment semantically coherent units such as named entities and collocations, whereas adaptive commitment preserves complete semantic segments when confidence is high. Specifically, we select an example from the Natural Questions dataset and examine the same input question, as shown in Figure 3. With fixed-length decoding, the ensemble converges to an incorrect answer ("Terrence"). In contrast, adaptive word commitment allows one model to generate the correct multi-word entity ("Kelly Reno") within a single decoding round, which is then selected by the subsequent cross-model scoring step.

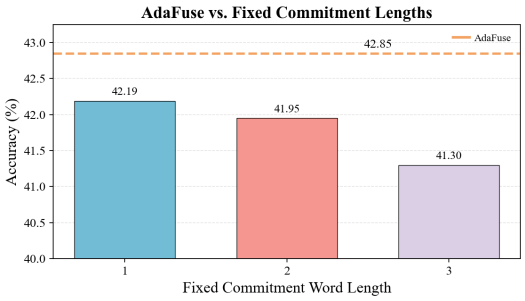

Figure 2: Comparison of Fixed-Length Decoding and ADAFUSE on Natural Questions Dataset.

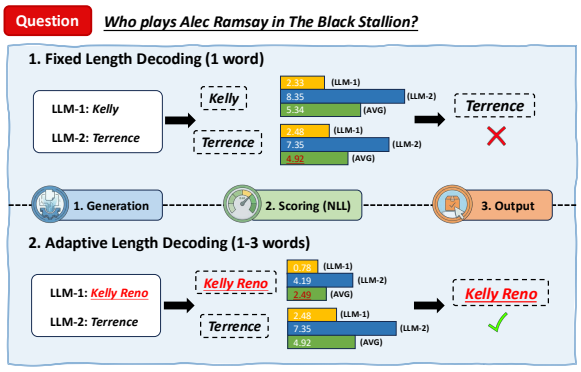

Figure 3: A Qualitative Example from Natural Questions: Fixed-Length Decoding vs. ADAFUSE

Distribution of Words per Decoding Round. We further analyze the distribution of the number of words generated per decoding round across different datasets. On Natural Questions, the proportions of rounds generating 1, 2, and 3 words are [80.29, 15.53, 4.18]%, respectively; on SQuAD, they are [57.64, 21.27, 21.09]%; and on De$\rightarrow$En translation, they shift to [34.02, 19.89, 46.09]%. These results indicate that ADAFUSE adapts the length of word-level commitment to the characteristics of different tasks, rather than relying on a fixed commitment length.

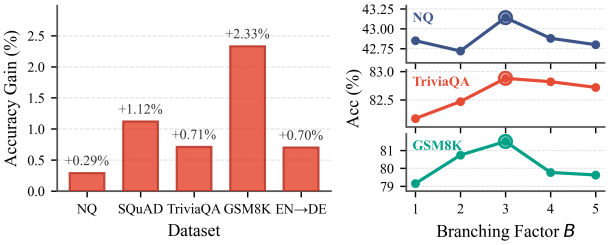

Figure 4: Diversity-Aware Ensemble Scaling Analysis

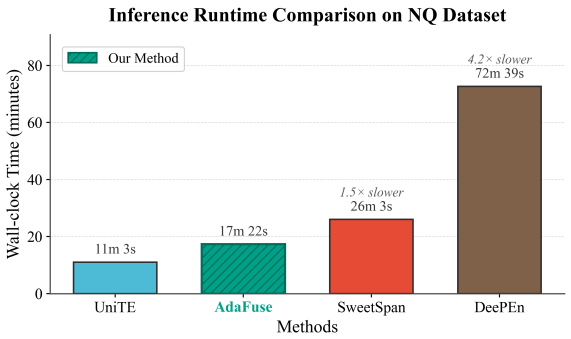

Figure 5: End-to-end runtime comparison of ensemble decoding methods on the Natural Questions (NQ) dataset under a unified multi-GPU inference setting.

4.4 Scaling Analysis (RQ3)

We further examine the impact of introducing diversity-aware ensemble scaling into ADAFUSE. Using two main models, we evaluate how adaptive branching influences performance across multiple datasets. As shown in Figure 4 (right), increasing the branching factor B from 1 to 5 on NQ, TriviaQA, and GSM8K leads to a generally upward trend in accuracy, suggesting that moderate branching helps the decoder explore more informative candidate paths. We then fix B=3 and compare ADAFUSE with and without diversity-aware ensemble scaling across datasets. As illustrated in Figure 4 (left), enabling diversity-aware ensemble scaling consistently improves performance on five benchmarks. These results indicate that, despite additional computational cost, ADAFUSE provides a reliable accuracy gain by enhancing diversity during decoding.

In Appendix F, we compare diversity-aware ensemble scaling with a beam-search-based scaling variant that ignores diversity in Figure 7. The latter leads to a clear performance drop, while our approach remains stable and effective as the number of candidates increases.

4.5 Case Study (RQ4)

Runtime Analysis. Figure 5 reports wall-clock inference time for different ensemble decoding strategies under a standardized setting (batch size 1, up to 10 new tokens per decoding call) on 4× NVIDIA A100 80GB GPUs using the main fixed base-model pair. ADAFUSE achieves runtime performance comparable to UniTE, which relies primarily on lightweight tokenizer-level transformations and introduces minimal per-step overhead. At the same time, ADAFUSE is substantially faster than span-level methods such as SweetSpan, whose fixed-span generation and scoring incur higher latency, and also outperforms token-level approaches like DeepEn that suffer from dense vocabulary alignment and cross-space projection costs. Overall, ADAFUSE strikes a favorable balance between efficiency and quality, delivering improved accuracy while maintaining practical inference efficiency. Importantly, the trailing-whitespace boundary detection used in ADAFUSE does not introduce wasted generation at each decoding round: although the decoder generates one additional whitespace token to detect the end of the current word, that whitespace corresponds to the separator that would appear in the final output anyway. In practice, we discard it for boundary detection and then append the whitespace once the word is committed, so these two operations effectively offset each other rather than introducing an extra useless token per round.

Case Analysis. To better characterize the behavior of ADAFUSE in longer-form generation, we additionally present a concrete generation example from the De→En dataset in Example 1. In this example, each parenthesized span corresponds to one decoding round. As shown, ADAFUSE frequently commits semantically cohesive word groups in longer sequences, reflecting its tendency to adapt word-level commitment to meaningful lexical units.

Example 1: De→En Translation Instance

Source (De): Singapur ist im Allgemeinen ein äußerst sicherer Ort, an dem man sich gut zurechtfindet und nach der Ankunft fast alles kaufen kann.

Output (En): [Singapore] [is generally a] [very safe place] [where] [you] [can] [easily get around] [and buy] [almost everything] [after arrival].

Note: [ ] denotes a word span generated in each decoding round.

Generalization to Harder Reasoning Tasks To further evaluate whether ADAFUSE generalizes to more challenging reasoning settings, we conduct additional experiments on GPQA-Diamond and include a thinking-style model, Qwen3-4B-Thinking-2507. Compared with the benchmarks in our main experiments, this setting involves longer reasoning chains and more sustained decoding uncertainty.

Table 2: Performance comparison between ADAFUSE and SweetSpan on the GPQA-Diamond benchmark.

Method	Accuracy (%)
LLaMA-3.1-8B	30.30
Qwen3-4B-Thinking	34.34
SweetSpan	32.83
ADAFUSE	**35.86**

ADAFUSE achieves the best performance, outperforming both single-model baselines and SweetSpan. Notably, the performance gap over SweetSpan becomes more pronounced compared to easier tasks, suggesting that problems with longer and more complex reasoning benefit more from adaptive fusion than fixed-span strategies. These results indicate that the confidence-guided and adaptive decoding mechanism generalizes to harder reasoning tasks and remains effective when combined with thinking models.

5 Conclusion

We introduced ADAFUSE, an adaptive ensemble decoding framework that enables confidence-guided fusion of multiple large language models at inference time. Experiments across open-domain question answering, arithmetic reasoning, and machine translation show that ADAFUSE consistently outperforms strong ensemble baselines under comparable computational budgets. Further analyses demonstrate the effectiveness of adaptive word commitment and controlled diversity in improving robustness and scalability.

More broadly, ADAFUSE highlights the value of adaptive control for coordinating heterogeneous models during generation, allowing ensemble behavior to respond flexibly to uncertainty across diverse tasks and evolving generation contexts.

Limitations

Our approach assumes access to token-level likelihoods and tokenizer outputs for cross-model scoring, which may not be available in closed-source or black-box LLM APIs. As a result, the applicability of ADAFUSE is currently limited to open or fully accessible models, and extending the framework to settings with restricted model interfaces remains an open direction for future work.

Acknowledgments

This work is supported by the National Science Foundation under Award No. IIS-2117902. The views and conclusions expressed in this paper are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the U.S. government.

References

Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. 2025. Resmoe: Space-efficient compression of mixture of experts llms via residual restoration. *arXiv preprint arXiv:2503.06881*.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, and 1 others. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

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

Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, Deqing Yang, and Hanghang Tong. 2024. Wapiti: A watermark for finetuned open-source llms. *arXiv preprint arXiv:2410.06467*.

Seong Yeub Chu, Jong Woo Kim, and Mun Yong Yi. 2025. Think together and work better: Combining humans' and llms' think-aloud outcomes for effective text evaluation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–23.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuqian Fu, Yuanheng Zhu, Jiajun Chai, Guojun Yin, Wei Lin, Qichao Zhang, and Dongbin Zhao. 2025. Rlae: Reinforcement learning-assisted ensemble for llms. *arXiv preprint arXiv:2506.00439*.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Ting Liu, and Bing Qin. 2024. Ensemble learning for heterogeneous large language models with deep parallel collaboration. *Advances in Neural Information Processing Systems*, 37:119838–119860.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de Las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, and 1 others. 2023a. Mistral 7b. corr, abs/2310.06825, 2023. doi: 10.48550. *arXiv preprint ARXIV.2310.06825*, 10.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Chaudhary, Lijie Hu, and Jiayi Shen. 2025. Smoa: Improving multi-agent large language models with s parse m ixture-o f-a gents. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 54–65. Springer.

Qiqi Lin, Xiaoyang Ji, Shengfang Zhai, Qingni Shen, Zhi Zhang, Yuejian Fang, and Yansong Gao. 2025. Life-cycle routing vulnerabilities of llm router. *arXiv preprint arXiv:2503.08704*.

Cong Liu, Xiaojun Quan, Yan Pan, Liang Lin, Weigang Wu, and Xu Chen. 2024. Cool-fusion: Fuse large language models without training. *arXiv preprint arXiv:2407.19807*.

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*.

Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Yue Yu, and Ping Luo. 2024. Specfuse: Ensembling large language models via next-segment prediction. *arXiv preprint arXiv:2412.07380*.

Xuying Ning, Dongqi Fu, Tianxin Wei, Mengting Ai, Jiaru Zou, Ting-Wei Li, and Jingrui He. Mc-search: Benchmarking multimodal agentic rag with structured reasoning chains. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Buu Phan, Brandon Amos, Itai Gat, Marton Havasi, Matthew Muckley, and Karen Ullrich. 2024. Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles. *arXiv preprint arXiv:2410.09303*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ashutosh Sathe, Divyanshu Aggarwal, and Sunayana Sitaram. 2024. Improving self consistency in llms through probabilistic tokenization. *arXiv preprint arXiv:2407.03678*.

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*.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. 2023a. Fusing models with complementary expertise. *arXiv preprint arXiv:2310.01542*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, and 1 others. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*.

Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. 2024. Hit the sweet spot! span-level ensemble for large language models. *arXiv preprint arXiv:2409.18583*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yuxuan Yao, Han Wu, Mingyang Liu, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. 2024. Determine-then-ensemble: Necessity of top-k union for large language model ensembling. *arXiv preprint arXiv:2410.03777*.

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*.

Zhichen Zeng, Qi Yu, Xiao Lin, Ruizhong Qiu, Xuying Ning, Tianxin Wei, Yuchen Yan, Jingrui He, and Hanghang Tong. 2025. Harnessing consistency for robust test-time llm ensemble. *arXiv preprint arXiv:2510.13855*.

Yuheng Zhang, Dian Yu, Tao Ge, Linfeng Song, Zhichen Zeng, Haitao Mi, Nan Jiang, and Dong Yu. 2025. Improving llm general preference alignment via optimistic online mirror descent. *arXiv preprint arXiv:2502.16852*.

A Potential Risks

ADAFUSE adopts an adaptive decoding design that may increase GPU energy consumption, primarily due to a higher number of forward passes during inference. Compared to standard decoding, the framework repeatedly performs forward evaluations to generate candidate word spans and to score them across models, which increases the cumulative computational workload on the GPU. Nonetheless, relative to *fixed-length* word-level ensembling, ADAFUSE can reduce the total number of decoding rounds by committing longer multi-word segments when confidence is high. As a result, the additional per-round computation is offset by fewer decoding rounds and synchronization steps, often leading to improved end-to-end throughput despite the increased number of forward passes.

B Use Or Create Scientific Artifacts

Our work builds directly on publicly available benchmarks and contributes novel code artifacts to the community. Specifically, we use standard QA and translation datasets (NaturalQuestions, SQuAD, TriviaQA, GSM8K, FLORES) for evaluation, without modifying their contents. In addition, we develop and release the ADAFUSE codebase, providing in a well-organized repository with clear module structure.

B.1 Cite Creators Of Artifacts

All externally sourced artifacts are cited to their original publications and repositories. The NaturalQuestions (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2016), and TriviaQA (Joshi et al., 2017) benchmarks are credited to their respective authors; the GSM8K dataset is attributed to (Cobbe et al., 2021); and the FLORES evaluation suite is acknowledged to its creators,(Goyal et al., 2022).

Each model family used (LLaMA-3.1, Mistral-7B, Qwen3-8B, InternLM3-8B) is referenced via its official technical report or HuggingFace model card, ensuring that credit is properly given to all upstream contributors.

B.2 Discuss The License For Artifacts

We adhere to the licenses of all artifacts we use or release. All question-answering and translation benchmarks are distributed under Creative Commons or public-domain terms as specified by their maintainers.

Model checkpoints carry their original open-source licenses (for example, MIT for Qwen3 and CC-BY-NC for InternLM3), and our own code and generated data are released under the MIT license. License files are included in our repository and documented in the supplementary README.

B.3 Artifact Use Consistent With Intended Use

We confirm that our use of each dataset and pre-trained model aligns with its intended scope. All benchmarks are used strictly for inference and evaluation, consistent with their terms of service, and no modified model weights are redistributed. Our released code operates in inference mode only and does not enable fine-tuning or commercial redistribution of the original checkpoints.

B.4 Data Contains Personally Identifying Info Or Offensive Content

Though benchmarks include public text with personal names or colloquial language, we:

- Restrict to publicly released benchmark passages only.
- Apply keyword filtering to remove or mask offensive content in our analysis summaries.

B.5 Documentation Of Artifacts

All released artifacts are accompanied by clear and sufficient documentation to support reproducibility and reuse. The codebase is organized into modular components, with each module containing inline comments that describe its functionality and key parameters.

B.6 Use of AI Assistants

AI assistants (e.g., ChatGPT) were used to support writing clarity, language refinement, and minor code-level assistance during the preparation of this manuscript. All technical contributions, including method design, experimental setup, implementation decisions, and result analysis, were conceived and conducted by the authors. The use of AI assistants did not influence the scientific claims or empirical conclusions reported in this work.

C Statistics For Data

The following descriptive statistics summarize the datasets used in this study:

C.1 **Flores (English to German)** (Machine Translation)

- Number of entries: 1012
- Average question length: 130.4 characters
- Average answer length: 152.0 characters

C.2 **Flores (German to English)** (Machine Translation)

- Number of entries: 1012
- Average question length: 152.0 characters
- Average answer length: 130.4 characters

C.3 **GSM8K** (Arithmetic Reasoning)

- Number of entries: 1319
- Average question length: 239.9 characters
- Average answer length: 272.3 characters

C.4 **NaturalQuestions** (Open-Domain Question Answering)

- Number of entries: 3610
- Average question length: 47.7 characters
- Average answer length: 24.7 characters

C.5 **SQuAD 2500** (Reading Comprehension QA)

- Number of entries: 2500
- Average question length: 877.5 characters
- Average answer length: 66.1 characters

C.6 **Wikipedia Test 6000** (Open-Domain QA)

- Number of entries: 6000
- Average question length: 78.8 characters
- Average answer length: 267.9 characters

D Computational Experiments

All computational experiments described in this work are fully reproducible and documented in both the main text and the supplementary materials.

D.1 Model Size And Budget

For each LLM used, we specify its total parameter count and approximate GPU memory footprint: LLaMA-3.1-8B, Qwen3-8B, and InternLM3-8B each contain roughly 8 billion parameters and require about 30GB of GPU memory in 16-bit precision, while Mistral-7B has 7 billion parameters. The total compute budget for all experiments is approximately 500 A100 GPU-hours, as detailed in Section 4.

D.2 Experimental Setup And Hyperparameters

We describe every experimental setting in Section 4 and Appendix C. All experiments are conducted on NVIDIA A100 80 GB GPUs. Key hyperparameters include a confidence threshold $\tau_\Delta = 0.7$, a maximum of 3 consecutively committed words per decoding round, and a task-dependent context length limit of up to 512 tokens.

In addition, we conducted a targeted sensitivity analysis on the confidence threshold τ_Δ, which governs the adaptive decision about word-commitment in AdaFuse.

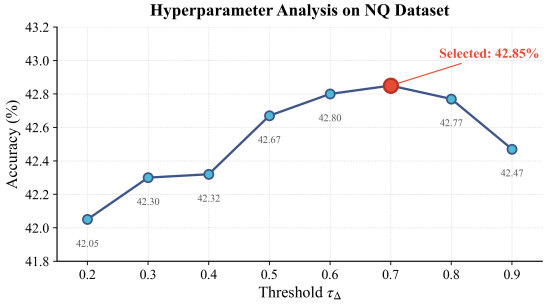

Figure 6: Effect of the confidence threshold τ_Δ on AdaFuse performance on the Natural Questions dataset.

As shown in Figure 6, the performance of the model degrades noticeably when τ_Δ is set to very small or very large values, indicating an increased sensitivity under extreme thresholds. In contrast, accuracy remains relatively stable in a broad region around $\tau_\Delta = 0.7$, with only minor fluctuations across nearby settings. Based on this observation, we adopt $\tau_\Delta = 0.7$ as a unified threshold for all datasets in our main experiments.

D.3 Descriptive Statistics

For each result table in the main text (Table 1), we report results from a deterministic decoding setting.

D.4 Parameters For Packages

The software environment used in our experiments is fully specified in the supplementary materials. We use PyTorch (v2.4.1, CUDA 12.1) as the core deep-learning framework. Model loading, inference, and tokenization are implemented using HuggingFace Transformers (v4.51.3) and Tokenizers (v0.21.0).

For multi-GPU inference and distributed execution, we rely on Accelerate (v1.6.0). The data set handling and evaluation pipelines are implemented with the HuggingFace Datasets library (v3.0.2).

Regarding evaluation, we report task-specific

Table 3: Impact of ensemble size on NQ accuracy.

Model & Ensemble Setting	NQ Acc.
InternLM3-8B-Instruct	27.15
LLaMA-3.1-8B-Instruct	34.24
Qwen3-8B	23.85
Mistral-7B-Instruct-v0.3	32.05
AdaFuse (2 models, B=3)	40.39
+ Mistral-7B-Instruct-v0.3 (3 models)	**42.63**
+ Qwen3-8B (4 models)	40.55

metrics consistent with prior work: exact match accuracy for question answering tasks (e.g., Natural Questions, SQuAD, TriviaQA), answer accuracy for GSM8K, and BLEU scores for machine translation benchmarks. In addition, we use the `BERTScore` package (v0.3.13) to compute BERTScore as a semantic similarity metric for generation quality assessment, where applicable.

E Effect of Ensemble Size

We further examine how AdaFuse scales with the number of models in the ensemble on NQ dataset. Table 3 reports the Natural Questions accuracy obtained by incrementally adding models to the ensemble, starting from a two-model configuration and extending to three and four models.

The results show that AdaFuse achieves a clear improvement when moving from single-model baselines to a two-model ensemble. Adding a third model (Mistral-7B-Instruct-v0.3) further increases accuracy, indicating that incorporating a strong and complementary model can provide additional gains. In contrast, extending the ensemble to four models by including Qwen3-8B leads to a slight performance drop, which can be attributed to the relatively weaker standalone performance of Qwen3-8B on NQ.

F Diversity-Aware Scaling vs. Beam-Search Scaling

To further evaluate the effectiveness of diversity-aware ensemble scaling, we conduct an additional analysis on the Natural Questions (NQ) dataset using the same main fixed base-model pair as in the primary experiments, and compare AdaFuse with a beam-search-based scaling variant. Conventional beam search is formulated at the token level, while AdaFuse operates at the word level, where the number of subword tokens required to complete a word varies across lexical forms and tokenizers. This discrepancy prevents a direct alignment between beam-search steps and word-level decoding rounds. To enable a fair comparison, we implement a token-level beam search that generates a fixed number of new tokens per decoding round (set to 5 in our experiments), followed by decoding and truncation to the first complete word. As shown in Figure 7, this beam-search-based variant exhibits degraded scaling behavior as the branching factor increases. We attribute this outcome to the absence of adaptive word commitment and the lack of an explicit diversity-aware exploration–exploitation mechanism, which together result in highly redundant initial word candidates and limited effective diversity at the word level.

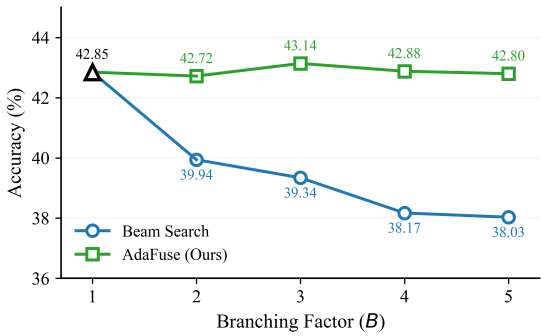

Figure 7: Comparison of Diversity-Aware Ensemble Scaling and Beam-Search-Based Scaling on the Natural Questions dataset.