Alignment-Augmented Speculative Decoding with Alignment Sampling and Conditional Verification

Anonymous ACL submission

Abstract

Recent works have revealed the great potential of speculative decoding in accelerating the autoregressive generation process of large language models. The success of these methods relies on the alignment between draft candidates and the sampled outputs of the target model. Existing methods mainly achieve draft-target alignment with training-based methods, e.g., EAGLE, Medusa, involving considerable training costs. In this paper, we present a training-free alignment-augmented speculative decoding algorithm. We propose alignment sampling, which leverages output distribution obtained in the prefilling phase to provide more aligned draft candidates. To further benefit from high-quality but non-aligned draft candidates, we also introduce a simple yet effective flexible verification strategy. Through an adaptive probability threshold, our approach can improve generation accuracy while further improving inference efficiency. Experiments on 8 datasets (including question answering, summarization and code completion tasks) show that our approach increases the average generation score by 3.3 points for the LLaMA3 model. Our method achieves a mean acceptance length up to 2.39 and speed up generation by 2.23×.

1 Introduction

The enormous size of state-of-the-art autoregressive models (Anthropic, 2024; Meta-AI, 2024; OpenAI, 2024) demands substantial memory and processing power, making real-time applications challenging. In scenarios such as interactive text generation, these models require vast amounts of computation, leading to slower response times and increased energy consumption. Hence, more efficient decoding algorithms are urgently needed to reduce inference costs significantly while maintaining or improving generation quality.

Speculative decoding (SD) (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2023, 2024; Li et al., 2024) has emerged as an effective solution to the inefficient decoding process of large autoregressive models. It uses a "draft and verify" mechanism to achieve lossless acceleration. A lightweight drafting model generates candidate tokens, which are then verified in parallel by the target model. This enables the generation of multiple tokens in a single step, thus accelerating decoding while preserving the output distribution. Speculative decoding algorithms that combine large and small models (Miao et al., 2023; Li et al., 2024) have shown promising results. However, these approaches often require additional training and extra parameters, complicating deployment in inference systems (Contributors, 2023; Kwon et al., 2023).

As an alternative, the training-free retrieval-based speculative decoding (Yang et al., 2023; Saxena, 2023; He et al., 2024) uses external knowledge sources such as databases or historical text to retrieve n-grams as drafts for generation. It is more flexible and scalable to large models, as the drafting cost is independent of model size. However, two issues have been largely overlooked in previous work on retrieval-based speculative decoding. First, poor alignment between the retrieved drafts and the model's output distribution leads to a low acceptance rate. Second, in many generation scenarios, the input context and the generated content have strong correlation. As a result, drafts retrieved from the input context tend to be of higher quality and do not require strict verification to ensure consistency with the target model's output.

In this paper, we propose **A**lignment-**A**ugmented **S**peculative **D**ecoding (AASD), a plug-and-play generation algorithm that improves the draft-target alignment in prompt-based speculative decoding. To address the first issue, we introduce alignment sampling, where we sample additional tokens from the output distribution of the prefilling phase for poorly aligned tokens, thereby improving the overall draft quality and increasing the chances of suc-

cessful alignment with output distribution of the target model. To address the second issue, we propose a conditional verification strategy. It applies heuristic probability thresholds for each token, based on its information entropy, allowing the model to autonomously utilize the input text during decoding through speculative decoding. It improves both generation accuracy and efficiency. Both the two techniques improve the alignment between the retrieved draft candidates and the target model output distribution. Alignment sampling makes the drafts more aligned with the target model distribution, and conditional verification makes the target model more aligned with the high-quality drafts.

We conduct comprehensive experiments with two different models on 8 datasets in long context generation scenarios, including question answering, summarization, and code completion tasks, in Section 4 to evaluate the effectiveness of AASD. The results show that AASD outperforms common sampling methods in terms of generation performance, improving the average score from 44.69 to 47.98 compared to greedy sampling for the LLaMA3 model (Meta-AI, 2024). In terms of generation efficiency, AASD achieves the highest acceptance rate and average decoding throughput among existing retrieval-based speculative decoding algorithms. It achieves a decoding speedup of up to 2.23 times.

In summary, we highlight the importance of draft-target alignment in retrieval-based speculative decoding in this paper. We propose a training-free method called alignment-augmented speculative decoding to improve this alignment, which consists of alignment sampling and conditional verification. The proposed method improves generation accuracy and efficiency compared to baseline methods.

2 Backgrounds

2.1 Speculative Decoding

Given a prompt $q = (x_1, x_2, ..., x_l)$ and language model M, where $x_{i(i=1,2,...,l)}$ represents each token, and l is the prefix length, we input the current sequence into the model to obtain the next token in the autoregressive decoding manner:

$$y_{l+1} \sim p_M(x_{l+1}|(x_1, x_2, ..., x_l)), \quad (1)$$

where $p_M(\cdot|\cdot)$ represents the output probability distribution of M. The model can generate only one token in one forward propagation, resulting in low efficiency, particularly for long generation lengths..

Speculative decoding (SD) alleviates this problem by adopting a "draft and then verify" mechanism. At each decoding step, a lightweight model m generates a draft $d = (\hat{x}_{l+1}, \hat{x}_{l+2}, ..., \hat{x}_{l+l_d})$ for the next tokens based on the current input sequence, where $\hat{x}_i (i = l+1, l+2, ..., l+l_d)$ denotes the draft tokens and l_d represents the draft length. This draft is then simultaneously input into the target model for verification. Tokens in the draft that match the target model's output will be accepted. Note that if a token is rejected, all subsequent tokens in the draft are also rejected. This allows the model to generate multiple tokens in a single forward pass.

2.2 Retrieval-Based Drafting

An effective draft model must address two crucial aspects. First, it must minimize computational costs, ensuring faster and more efficient inference through a lightweight design. Second, it must closely align with the target model. Ideally, perfect alignment would result in a 100% acceptance rate, thus achieving the theoretical maximum speedup, defined as the throughput of the smaller model divided by that of the target model.

Approaches such as EAGLE (Li et al., 2024) strengthen the alignment between the draft and target models through alignment training, causing inconvenience in adapting to different models. Retrieval-based speculative decoding methods complete the drafting by retrieving n-grams from historically generated data or pre-provided databases based on the current sequence. However, existing retrieval-based speculative decoding methods (Yang et al., 2023; Saxena, 2023; He et al., 2024) ignore the importance of alignment. Although their drafting overhead is very small compared to methods that require additional modules, their acceptance rate is also relatively lower.

2.3 Impact of Draft-Target Alignment

Recent research (Bachmann et al., 2025) shows that even a high-quality draft will have a low acceptance rate if it does not align with the model output distribution. We also conduct an experiment to give an insight into the impact of draft-target alignment in retrieval-based speculative decoding. For the four-token input sequence "A, B, C, D", the model can predict the probability distribution of the next token for each input token in parallel. For each distribution, we select the token with the highest probability, that is, greedy sampling. Assume that the sampled tokens are B, C, F, E in order. The sam-

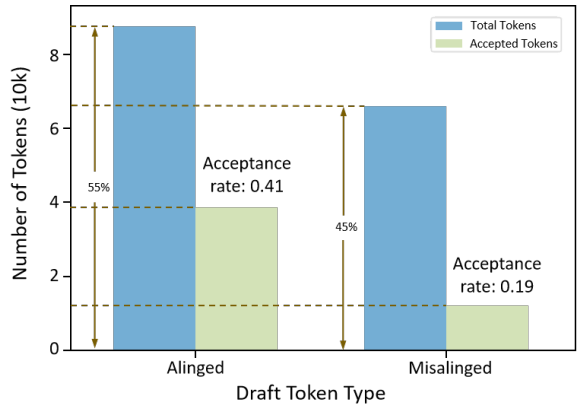

Figure 1: Acceptance rate of aligned and misaligned draft tokens in prompt-based speculative decoding.

pled tokens "B" and "C" match the input, while "F" does not match "D" in the input. We regard the matching tokens ("B" and "C") as aligned tokens and the unmatched tokens ("D") as misaligned tokens. We conduct prompt-based speculative decoding on 100 samples in NQ (Kwiatkowski et al., 2019) dataset with LLaMA3.1-8B-Instruct (Meta-AI, 2024) and calculate the acceptance frequency of aligned and misaligned draft tokens. Results are shown in Figure 1. About 45% retrieved draft tokens are misaligned with the model output. The acceptance rate of misaligned tokens is much lower than that of aligned tokens. Inspired by this, for the positions of misaligned tokens, we can sample additional tokens from the output distribution to enhance the alignment of the entire draft and the model, thus improving the acceptance rate.

2.4 Potential for Non-Strict Verification

Most speculative decoding methods adopts strict verification to achieve lossless generation acceleration. The recent research (Bachmann et al., 2025) claims that correct but non-aligned draft candidates can be accepted to pursue higher speed-up ratio. They trained an additional module on carefully designed data to judge whether to accept draft tokens. However, for prompt-based speculative decoding, as the draft source is highly relevant to the answer, a training-free verification strategy can be designed to realize a better trade-off between generation performance and efficiency. For example, in long context generation scenario, the input context contains the key information needed for model generation. Some of the fragments even overlap with the standard generated content. Given this, we can design a conditional verification method to allow the model to accept high-quality draft tokens, rather than just accept that match its original output. This allows the model to adaptively call the input original segments on the decoding side, thus enhancing the generation accuracy and efficiency.

3 Method

In this section, we introduce AASD, an alignment-augmented speculative decoding algorithm based on alignment sampling and conditional verification that improves both generation performance and efficiency. It is a plug-and-play speculative decoding algorithm that does not require additional modules or training. The overall description of AASD is dispalyed in Algorithm 1 in Appendix C.

3.1 Context-Aware Drafting

For long text generation, the input context can act as an important source of draft, because it is highly relevant to the generated content. We construct the input of each request into a draft pool by sliding window sampling, which is an index dictionary. The keys in the draft pool are several consecutive tokens in the input context and the values are the position indexes where these tokens appear. We update the draft pool when new tokens are generated.

In each decoding step, we use the last several tokens of the current sequence as the key to retrieve indexes in the draft pool. Since longer keys give higher draft relevance, following REST (He et al., 2024), we prioritize searching with longer keys and obtain n-grams from the current sequence.

3.2 Alignment Sampling

As is shown in Figure 2, we perform alignment sampling to expand the retrieved n-grams into draft trees. For example, the token "is" is the third most likely word in the probability distribution of the next word of the token "Jackson". Therefore, "was" and "made" are potential high-quality draft tokens, which are more aligned with the output distribution of the target model. We additionally sample these two tokens for the 100th position. In this way, we expand and merge multiple candidate n-grams into a draft tree and calculate the corresponding tree attention mask for parallel verification. The draft tree $\mathbb{T}$ is defined as:

$$\mathbb{T} = (\mathbb{V}, \mathbb{E}), \mathbb{V} = \bigcup_{i=l+1}^{l+r} \bigcup_{j=1}^{n_i} \left\{ \hat{x}_i^j \right\}, \qquad (2)$$

where $\mathbb{V}$ and $\mathbb{E}$ is the set of its nodes and edges. n_i is the number of retrieved tokens in the i_{th}

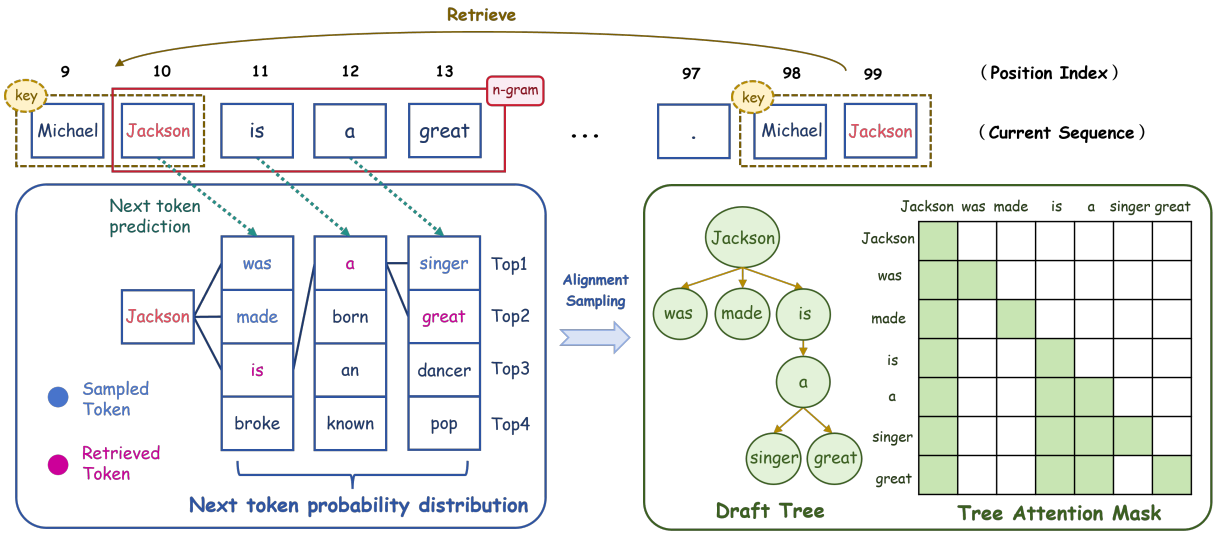

Figure 2: Illustration of alignment sampling. "Jackson is a great" is an n-gram retreived by the key "Michael Jackson". According to the output distribution obtained in the prefilling stage, "is" is the token with the third highest probability. The top-1 token "was" and top-2 token "made" are more likely to be generated after "Michael Jackson". Therefore, we sample two more tokens ("was" and "made" for the 100th position). Token "a" is well aligned with the model output, so we keep it unchanged for the 101st position.

layer of $\mathbb{T}$. r is the depth of $\mathbb{T}$. Then we input $\mathbb{T}$ and the corresponding tree attention mask into the model and get the output probability distribution $P_M(x_i|\mathbb{F}(\hat{x}_i^j))$ for each node, where $\mathbb{F}(\hat{x}_i^j)$ is the set of all parent nodes (including prefix) of $\hat{x}_i^j$.

3.3 Conditional Verification

As discussed in Section 2.2, high-quality drafts retrieved from prompts have the potential to be non-rigorously verified to enhance generation accuracy. The naive idea is to use a fixed probability threshold to filter draft tokens or use top-k verification. We argue that using the probability threshold condition is superior to using the top-k condition alone since the output probability directly reflects the model's confidence in each draft token. Moreover, the verification condition of top-k is not applicable for some extreme cases. For example, suppose the probability of a certain token is close to 1, and the probability of all other tokens is close to 0. In that case, no other token should be accepted, except for the token with the highest probability. If all tokens have the same probability, then they should all be accepted or rejected at the same time. However, the top-k verification condition will only pass individual tokens and reject others. Therefore, we first set a probability threshold δ so that token $\hat{x}_i^j$ will be accepted if the following conditions are met:

$$p_M(\hat{x}_i^j|\mathbb{F}(\hat{x}_i^j)) \geqslant \delta. \quad (3)$$

However, models exhibit varying levels of confidence across different samples or tokens. A static threshold may either over-restrict or under-restrict the verification process, leading to suboptimal performance. Therefore, we formulate an adaptive threshold based on the verification probability distribution. Low-confidence tokens can be processed with stricter thresholds to maintain precision, while high-confidence tokens benefit from relaxed thresholds to ensure recall. Therefore, we adjust the verification threshold of each token based on the information entropy of its verification probability distribution. Draft token $\hat{x}_i^j$ is accepted under the following condition:

$$p_M(\hat{x}_i^j|\mathbb{F}(\hat{x}_i^j)) \geqslant \delta_i^j, \quad (4)$$

where δ_i^j is calculated by:

$$\delta_i^j = min(-\alpha \sum P(\hat{x}_i^j) \log P(\hat{x}_i^j) + \beta, \Delta), \quad (5)$$

where

$$\Delta = \max_{\hat{x}_i^j} P(\hat{x}_i^j), \quad (6)$$

α and β are hyperparameters. α a is the factor affecting the threshold value due to information entropy. β is the benchmark value of the threshold. The function of Δ is to ensure that the token with the highest probability will be accepted during verification. Considering that the overall quality of the alignment sampled tokens is lower than the

tokens retrieved from the prompt. We only perform conditional verification on the draft token retrieved from the input context and selected by alignment sampling. Strict verification is still applied for tokens retrieved from generated context. We accept the longest draft candidate that passes verification.

4 Experiments

4.1 Settings

In most speculative decoding methods with strict verification (e.g., REST (He et al., 2024), PLD (Saxena, 2023) and LLMA (Yang et al., 2023)), the performance on datasets depends solely on the sampling strategy rather than the specific decoding algorithm. Equipped with conditional verification, AASD introduces a novel sampling method, which enhances access to input fragments. Therefore, we compare the performance of **AASD** with several common sampling methods: 1) **Greedy Sampling**: It samples the token with the highest probability; 2) **Top-*k* Sampling** (Fan et al., 2018): It selecting the next token from the top k most probable options ($k = 50$); **Nucleus Sampling** (Holtzman et al., 2020): It is also called top-p sampling ($p = 0.8$), which selects the next token from the smallest set of top tokens whose cumulative probability exceeds a predefined threshold; 3) **Beam Search** (Li et al., 2016): It explores multiple candidate sequences at each step, keeping only the top n most likely options (the beam width $n = 10$), to find an optimal or near-optimal output sequence.

On the other hand, we compare the performance and efficiency of **AASD** with autoregressive decoding and other retrieval-based speculative decoding methods. Baselines are as follows: 1) **Autoregressive decoding (ARD)**, 2) **REST** (He et al., 2024): It retrieves n-grams from an external general database and constructs a tree-structured draft, 3) **PLD** (Saxena, 2023): It retrieves n-grams directly from the input context. Note that **LLMA** (Yang et al., 2023) and **PLD** can be considered as the same method since they are very similar.

We evaluate the proposed method with LLaMA3.1-8B-Instruct (Meta-AI, 2024) and Qwen2.5-32B-Instruct (Qwen et al., 2025) on 3 different tasks: 1) **Question Answering (QA)**: Nature Question (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017), 2WikiMQA (Ho et al., 2020) and HotpotQA (Yang et al., 2018); 2) **Summarization**: Multi-News (Fabbri et al., 2019) and GovReport (Huang et al., 2021); 3) **Code Completion**: RepoBench-P (Liu et al., 2024) and LCC (Guo et al., 2023). For the NQ dataset, we randomly sampled 300 queries from the validation set for evaluation. We use subsets sampled by LongBench (Bai et al., 2023) for other datasets. We report F1 score for QA datasets, ROUGE-L for summarization datasets and Edit Sim for code completion datasets. We also test the decoding efficiency on **SpecBench** (Xia et al., 2024).

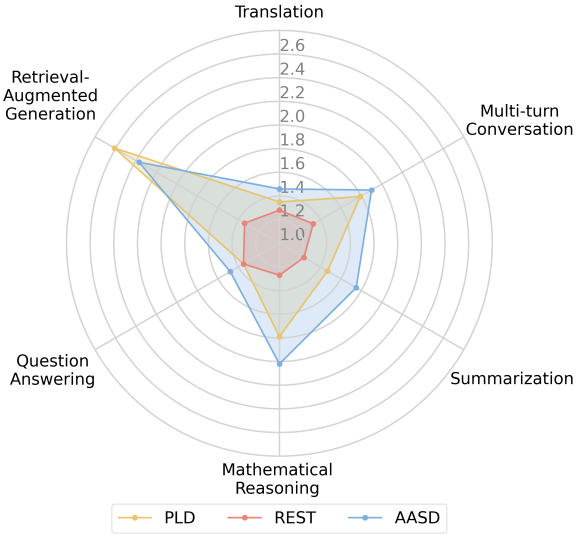

Figure 3: Speed-up ratio on the six different task categories in SpecBench. The center point (1.0×) represents the baseline of autoregressive decoding.

we use the same generation hyperparameters for all baselines. REST involves an additional database as the draft source. We selected the best database from the officially released database for testing (for each model and each datasets). We did not compare with other training-required methods for generating enhancements and speculative algorithms because they require additional training or use additional models. We use 6-grams and set the maximum length of the key for retrieval to 6 for AASD. We use unified hyperparameters for AASD on all datasets. α is set to 0.1 and 0.2 for LLaMA3.1 and Qwen2.5, respectively. β is set to 0.1. Each position can be extended by at most two tokens by alignment sampling during drafting. We use 4 A100-PCIE-40GB GPUs for all experiments.

4.2 Improvement on Accuracy

We compare the performance of different sampling methods in Table 1. We report the average scores of the 8 datasets in the last column. AASD outperforms other sampling methods with LLaMA3.1-8B-Instruct on these datasets except for TQA. It improves the average score from 44.69 to 47.98

Method	NQ	TQA	2WikiMQA	HotpotQA	Multi-News	GovReport	RepoBench-P	LCC	Avg.
	F1	F1	F1	F1	ROUGE-L	ROUGE-L	Edit Sim	Edit Sim	
LLaMA3.1-8B-Instruct									
Greedy	29.13	**92.13**	42.83	49.80	13.51	15.96	48.11	66.05	44.69
Top-*k*	29.08	84.52	38.48	42.01	13.34	15.18	45.36	56.19	40.52
Nucleus	29.21	86.39	39.98	44.79	13.58	15.46	45.59	55.82	41.35
Beam	32.57	83.04	43.29	48.93	14.07	17.75	46.27	67.52	44.16
AASD	**34.54**	91.56	**43.10**	**50.38**	**15.05**	**19.34**	**58.84**	**71.03**	**47.98**
Qwen2.5-32B-Instruct									
Greedy	41.90	81.90	32.35	35.73	**12.77**	14.58	37.72	60.85	39.72
Top-*k*	40.86	80.02	28.17	36.09	11.72	13.19	36.14	50.56	37.09
Nucleus	41.64	79.07	29.64	35.37	12.30	13.56	38.01	49.48	37.43
Beam	36.78	**84.57**	17.49	19.55	9.49	11.07	**42.04**	**63.26**	35.53
AASD	**43.34**	83.55	**34.99**	**39.12**	12.67	**15.25**	37.23	58.65	**40.60**

Table 1: Performance of different sampling methods with LLaMA3.1-8B-Instruct and Qwen2.5-32B-Instruct. For all metrics, higher scores indicate better performance. The best results are in bold.

Method	2WikiMQA			GovReport			LCC		
	MAL	TPS	Speed-up	MAL	TPS	Speed-up	MAL	TPS	Speed-up
LLaMA3.1-8B-Instruct									
ARD	1.00	20.10	1.00	1.00	20.15	1.00	1.00	20.67	1.00
REST	1.08	18.26	0.91	1.27	21.20	1.05	1.26	26.09	1.07
PLD	1.93	32.77	1.63	1.42	21.33	1.06	2.06	36.52	1.77
AASD	**2.37**	**37.18**	**1.85**	**1.97**	**34.00**	**1.69**	**2.39**	**46.20**	**2.23**
Qwen2.5-32B-Instruct									
ARD	1.00	8.08	1.00	1.00	8.14	1.00	1.00	7.35	1.00
REST	1.07	7.33	0.91	1.26	8.16	1.00	1.16	7.53	1.02
PLD	2.13	**14.04**	**1.74**	1.51	10.01	1.23	1.46	8.72	1.19
AASD	**2.14**	13.87	1.72	**2.18**	**14.20**	**1.74**	**1.89**	**11.80**	**1.61**

Table 2: Inference efficiency of AASD with LLaMA3.1-8B-Instruct and Qwen2.5-32B-Instruct. ARD refers to auto-regressive decoding with greedy sampling. MAL represents mean acceptance length and TPS represents tokens per second. The best results are in bold.

compared to greedy sampling. For Qwen2.5-32B-Instruct, AASD also improves the average score. However, it does not show an advantage in code-related tasks, possibly because the model's inherent code capabilities are relatively weak, making it unable to effectively reject the wrong draft tokens.

4.3 Improvement on Efficiency

We evaluate generation efficiency of AASD on TQA, GovReport, and LCC datasets (one dataset for each task). Table 2 displays the results with different retrieval-based speculative decoding approaches. Since the inputs of these tasks are long and have large length variance, input length has a great impact on the prefilling time (computation time of the first step). To avoid the influence of this method-irrelevant factor, we only report the TPS of incremental inference (without the first step of generation for each sample). REST adopts an independent database as the drafting source, thus having a low acceptance rate for out-of-domain tasks. AASD outperforms other approaches both on the perspective of MAL and throughput with LLaMA3.1-8B-Instruct. It achieves a speed-up ratio up to 2.23 compared with auto-regressive decoding. For Qwen2.5-32B-Instruct, AASD has a significant improvement in the generation efficiency of summary and code tasks. However, for 2WikiMQA, as the mean output length is very short (about 8 tokens), its performance is similar to PLD. The total time overhead for retrieving the draft candidates for each step is between 0.02 milliseconds and 2 milliseconds, which is less than 5% of the time cost of a single step for the 8B model. For larger target models, this overhead is negligible.

In addition, Figure 3 displays the evaluation results on SpecBench. We report the speed-up ratio compared with autoregressive decoding. AASD

Method	NQ	TQA	2WikiMQA	HotpotQA	Multi-News	GovReport	RepoBench-P	LCC	Avg.
	F1	F1	F1	F1	ROUGE-L	ROUGE-L	Edit Sim	Edit Sim	
AASD	**34.54**	91.56	**43.10**	50.38	15.05	**19.34**	**58.84**	**71.03**	**47.98**
w/ threshold verification	32.05	**91.82**	41.19	**51.22**	**15.16**	18.69	56.51	67.75	46.80
w/ top-k verification	31.21	82.71	22.12	27.34	14.93	18.15	43.17	45.62	35.66

Table 3: Results for ablation study with LLaMA3.1-8B-Instruct. The best results are in bold.

Method	2WikiMQA			GovReport			LCC		
	MAL	TPS	Speed-up	MAL	TPS	Speed-up	MAL	TPS	Speed-up
AASD	**2.37**	**37.18**	**1.85**	**1.97**	**34.00**	**1.69**	**2.39**	**46.20**	**2.23**
w/o alignment sampling	1.80	36.12	1.80	1.87	32.93	1.63	2.25	42.71	2.07
w/o conditional verification	2.36	35.86	1.78	1.86	32.60	1.62	2.34	39.02	1.89

Table 4: Inference efficiency of AASD without alignment sampling and conditional verification.

outperforms the baselines in most tasks, except for the RAG task, which further proves the effectiveness in improving decoding efficiency.

4.4 Ablation Study

We conduct the ablation study with LLaMA3.1-8B-Instruct in this section. We evaluate the performance of AASD with top-k verification and fixed threshold verification. For top-k condition, k is set to 5. For threshold verification, threshold is set to 0.1. Table 3 shows the results. AASD achieves the highest average score. Using the top-k condition decreases the scores for most datasets. For evaluation the generation efficiency, we consider AASD without alignment sampling and AASD without conditional verification. Table 4 displays the results. Both alignment sampling and conditional verification contribute to the inference acceleration.

4.5 Comparison with Large and Small models Collaboration

Another training-free approach is to directly use the same series of small models as the draft model (Leviathan et al., 2023; Kim et al., 2023). We compare AASD with this approach in this section. We test the inference efficiency on NQ dataset with LLaMA2 model series (Touvron et al., 2023). Table 5 displays the result. AASD outperforms SD on generation acceleration for the 7B and 70B model. For SD, the size of the draft model has a significant impact on inference efficiency. A draft model that is too small will result in a low acceptance rate, while a draft model that is too large will lead to excessive draft overhead. For a 70B target model, the 7B draft model has high quality, but its draft overhead accounts for about 80% of the total inference overhead. Therefore, it does not significantly improve throughput. It is difficult to obtain a draft model of appropriate size that can well balance acceptance rate and overhead. In contrast, AASD does not have this concern. It costs little overhead and performs well as the model size scales up.

Method	M	M_d	MAL	TPS	Speed-up
ADR	7B	-	1.00	34.32	1.00
SD	7B	68M	1.16	24.60	0.71
AASD	7B	-	**1.83**	**50.26**	**1.46**
ADR	70B	-	1.00	5.17	1.00
SD	70B	1B	1.14	2.68	0.52
SD	70B	7B	**4.13**	6.61	1.28
AASD	70B	-	1.83	**7.94**	**1.53**

Table 5: Comparison with speculative decoding with big and small model collaboration. SD represents standard speculative decoding. Column M and M_d indicates the target model size and draft model size.

4.6 Case Study

Section 4.2 shows that AASD has the potential to improve generation accuracy with a near relatively strict threshold. In this section, we explore how the verification threshold affects the generation results through a case study. Table 6 displays the predictions under different thresholds for a sample in 2WikiMQA. Vanilla output fails to give the correct answer. However, the prediction of AASD (δ=1e-3 and δ=1e-5) contains the ground truth. With an appropriate threshold, AASD improves the accuracy of generation through context-aware drafting from the relevant context. When the threshold is set to 1e-5, the logic of generated content begins to lose. The prediction is completely messy when the

Method	Label and Predictions
Ground Truth	**Mahesh Bhatt**
Greedy	Pooja Bhatt (✗)
AASD (δ=1e-3)	Pooja Bhatt's father is **Mahesh Bhatt**. (✓)
AASD (δ=1e-5)	Pooja Bhatt's television film Daddy was directed by her father **Mahesh Bhatt**. (✓)
AASD (δ=1e-7)	Pooja Bhatt's television film Daddy was directed by Pooja Bhatt, starring! the director's father, played by actor Anupam Kher (✗)

Table 6: Comparison of generated results with AASD under different thresholds. The question is "Who is the father of the director of film Kajraare?"

threshold comes to 1e-7. Therefore, the threshold should be large enough to keep the model output logical and fluent. Besides, too relaxed verfication may affect the model's ability to defend against attack inputs. If the input context contain harmful information, using AASD may lead the model to incorporate it into its responses. Therefore, in practical applications, AASD requires additional safety alignment to ensure the security of the language model. See Appendix A for a more comprehensive study for the impact of verification strictness.

5 Related Work

Context Utilization Using context to improve generation performance is typical in retrieval-augmented generation (RAG) scenarios. RAG methods (Zheng et al., 2023; Dai et al., 2023; Gao et al., 2023; Fan et al., 2024; Zhao et al., 2024) retrieves relevant documents based on the given input for model reference, thus improving the quality of generation. The naive approach (Ma et al., 2023) applies the search engine as the retriever and directly combines the retrieved documents with the user query as the input for frozen LLMs. Most RAG methods (Yoran et al., 2023; Luo et al., 2023; Asai et al., 2023; Melz, 2023; Yan et al., 2024) leveraging context on the input side to enhance generation. Self-RAG (Asai et al., 2023) introduces generating reflection tokens to enable customizing models' behaviors for different tasks. Speculative RAG (Wang et al., 2024b) adopts instruct-tuned draft models to drafting according to different retrieved documents and uses the target model to pick out the best draft as the final response. Apart from them, CoG (Lan et al., 2023) proposes a encoder-based model architecture to seek suitable text spans from the context during generation. Cao et al. (2024) improves CoG through linguistic heuristics initialization and iterative self-reinforcement.

Speculative decoding Stern et al. (2018); Leviathan et al. (2023); Xia et al. (2023, 2024) adopt a "drafting-verification" pattern to lossless accelerates autoregressive decoding. At each decoding step, a small model is used to draft the following few tokens. Then, the target model verifies the draft in parallel and accepts tokens that are consistent with the original output. In this way, multiple tokens can be generated in a single step. Some works (Leviathan et al., 2023; Cai et al., 2024; Li et al., 2024) employ independent small models or additional trained modules as the draft models, while others (Saxena, 2023; Fu et al., 2024; He et al., 2024) retrieve drafts from a draft pool. The draft structure has evolved from n-grams (Leviathan et al., 2023; Fu et al., 2024) to draft trees (Li et al., 2024; Wang et al., 2024a). Among them, PLD (Saxena, 2023) and LLMA (Yang et al., 2023) retrieve n-grams from the prompt to construct the draft. REST (He et al., 2024) uses a common public data source to build a draft pool and retrieves the tree structure draft according to the last several tokens of the current sequence at each decoding step. These retrieval-based methods avoid the hassle of additional training, but their acceleration performance is relatively lower.

6 Conclusion

In this paper, we proposed AASD, an alignment-augmented speculative decoding algorithm with alignment sampling and conditional verification. Both techniques improve the input-model alignment in prompt-based speculative decoding, thus increasing the mean acceptance length and improving decoding efficiency. Conditional verification based on a heuristic probability threshold shows the potential to enhance the generation accuracy at the token level. AASD does not introduce additional parameters or training, making it conveniently applicable to pre-trained LMs. It outperforms the baseline with two different models on the average score of 8 datasets in terms of generation performance and achieves a speed-up ratio of up to 2.23.

Limitations

Due to resource constraints, we only consider applying alignment sampling on prompt-based speculative decoding in this paper. However, alignment sampling is applicable to most existing retrieval-based speculative decoding (including REST) and most generated scenarios. Taking REST as an example, if we input the draft pool into the model in advance to obtain the output distribution of top-k tokens (or directly use the content of the model's historical output to build the draft pool), then when using REST, we can use alignment sampling to improve the alignment of the draft pool with the target model, thereby improving the draft token acceptance rate and algorithm performance. We leave this for future work.

If the input context contains harmful information, using AASD may lead the model to incorporate it into its responses. Therefore, in practical applications, AASD requires additional safety alignment to ensure the security of the language model.

References

Anthropic. 2024. Claude-3-5-sonnet model card.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. *Preprint*, arXiv:2501.19309.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*.

Bowen Cao, Deng Cai, Leyang Cui, Xuxin Cheng, Wei Bi, Yuexian Zou, and Shuming Shi. 2024. Retrieval is accurate generation. In *The Twelfth International Conference on Learning Representations*.

LMDeploy Contributors. 2023. Lmdeploy: A toolkit for compressing, deploying, and serving llm. `https://github.com/InternLM/lmdeploy`.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pages 12098–12107. PMLR.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. 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 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *Preprint*, arXiv:2104.02112.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is all you need. In *The Eleventh International Conference on Learning Representations*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*.

Tianyang Liu, Canwen Xu, and Julian McAuley. 2024. Repobench: Benchmarking repository-level code auto-completion systems. In *The Twelfth International Conference on Learning Representations*.

Hongyin Luo, Yung-Sung Chuang, Yuan Gong, Tianhua Zhang, Yoon Kim, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023. Sail: Search-augmented instruction learning. *arXiv preprint arXiv:2305.15225*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Eric Melz. 2023. Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation. *arXiv preprint arXiv:2311.04177*.

Meta-AI. 2024. Introducing llama 3.1: Our most capable models to date.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 2023. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*.

OpenAI. 2024. Learning to reason with llms.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Apoorv Saxena. 2023. Prompt lookup decoding.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024a. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, and 1 others. 2024b. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

A Trade-off between Accuracy and Efficiency

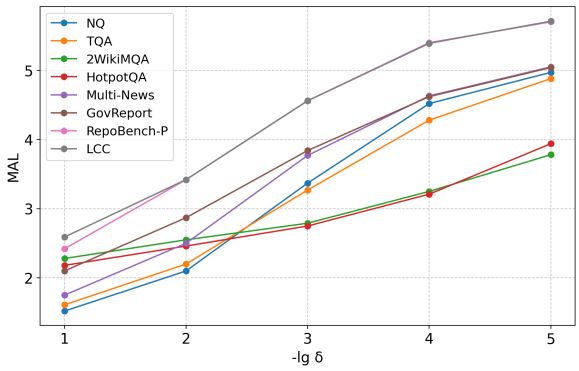

Figure 4: Mean acceptance length of different tasks under threshold verification.

For non-strict verification, more relaxed verification conditions lead to faster reasoning, but may also cause performance degradation. To study the trade-off between acuracy and efficiency, we conduct an experiment on LLaMA3.1-8B-Instruct (Meta-AI, 2024) with threshold verification.

Figure 5 demonstrates the performance on each dataset as the threshold changes. Note that alignment sampling is not applied. F1 score increases with decreasing threshold for NQ while decreases for other 3 QA datasets. The ground truth is implicit in the context for each sample in NQ. A more relaxed verification condition makes the model tend to use the original statement in the context, which may be more accurate than the model's output in some cases, thus leading to better results. Therefore, for high-quality context, the threshold can be lowered to increase confidence in the context for better performance and efficiency. For both summary datasets, the overall trend of the scores decreases as the threshold decreases in the search range. The performance drops rapidly for more complex tasks such as code completion.

Figure 4 displays the mean acceptance length under different thresholds. For all datasets, a lower threshold leading to a higher acceptance rate, thus improving the inference efficiency.

B Target Length and Source-Target Overlapping Rate of Each Datasets

To explore the effectiveness of AASD in different generation scenarios, we report the target length and overlapping rate between the input and output context of each dataset used in the main experiment. We adopts longest substring match ratio (R) to evaluate the source-target overlapping rate, which is calculated as:

$$R = \frac{LCS(X, Y)}{|Y|}, \tag{7}$$

where LCS is length of the longest common subsequence of input context X and target context Y. The average source-target overlapping rate and the average target length of each dataset are shown in Table 7. Even for QA tasks (e.g., TQA, 2WikiQA and HotpotQA) where the average target length is very short, AASD demonstrates effectiveness according to the results in Table 1. On the other hand, although AASD exhibits greater advantages in tasks with high overlap, it remains effective even in low-overlap scenarios, highlighting its versatility across diverse generation tasks.

C The Overall Description of AASD

Algorithm 1 AASD

Input: Prompt q, LM M, α, β, length of n-grams n, Maximum length of the key l.
Output: Prediction A_q

$pool \leftarrow \{\}$
for $k = 1$ to l **do**
  for $i = 0$ to $len(q) - k$ **do**
    $key \leftarrow q\,[i : i + k]$
    $pool\,[key]\,.append(i + k)$
  end for
end for ▷ Initialize the draft pool.
while $\langle eos \rangle$ not in q **do**
  $\mathbb{T} \leftarrow \{q[-1]\}$
  for $k = l$ to 1 **do**
    if $q\,[-k :] \in pool$ **then**
      $\mathbb{T} \leftarrow \mathbb{T} + Sample(q\,[pool\,[q\,[-k :]]])$
      ▷ Conduct alignment sampling.
      $break$
    end if
  end for
  $P \leftarrow M(\mathbb{T})$
  $y \leftarrow Verify(P, \mathbb{T}, \alpha, \beta)$
  $q \leftarrow q + y$
  for $k = 1$ to l **do**
    for $i = -k - len(y)$ to $-k$ **do**
      $key \leftarrow q\,[i : i + k]$
      $pool\,[key]\,.append(i + k)$
    end for
  end for ▷ Update the draft pool.
end while
$A_q \leftarrow q$

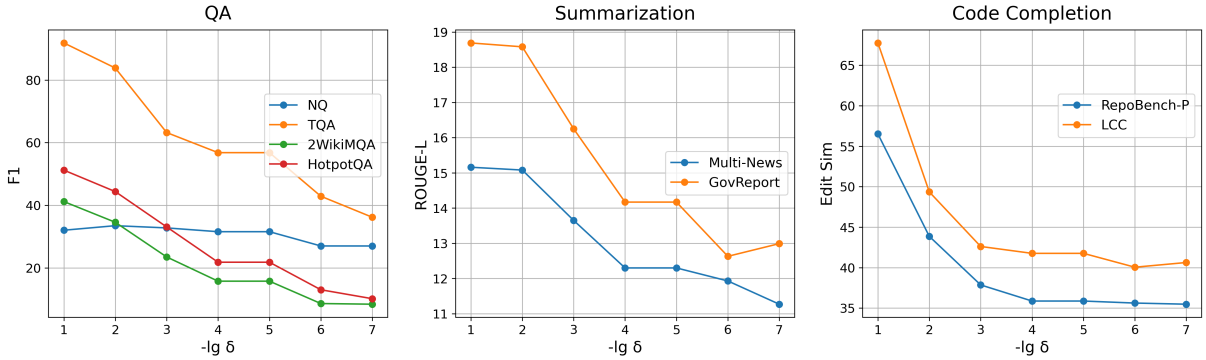

Figure 5: Performance of different tasks under threshold verification. The dotted lines in each figure are the baselines of the dataset with the corresponding colors.

	TQA	**2WikiQA**	**HotpotQA**	**NQ**	**MultiNews**	**GovReport**	**LCC**	**Repobench-P**
Input length	9681	8849	9458	3612	7882	8161	13516	15300
Target length	5	5	5	145	335	702	13	14
R	0.13	0.77	0.82	1.00	0.43	0.54	0.71	0.42

Table 7: The target length and overlapping rate (R) between the input and output context of each dataset used in the main experiment.