Beyond Tokens: Semantic-Aware Speculative Decoding for Efficient Inference by Probing Internal States

Ximing Dong[1], Shaowei Wang[2]*, Dayi Lin[3], Boyuan Chen[1], Ahmed E. Hassan[3]
[1]Centre for Software Excellence, Huawei, Canada
[2] Department of Computer Science, University of Manitoba, Canada
[3]School of Computing, Queen's University, Canada
{ximing.dong, boyuan.chen1}@huawei.com, shaowei.wang@umanitoba.ca,
dayi.lin@queensu.ca, ahmed@cs.queensu.ca

Abstract

Large Language Models (LLMs) achieve strong performance across many tasks but suffer from high inference latency due to autoregressive decoding. The issue is exacerbated in Large Reasoning Models (LRMs), which generate lengthy chains of thought. While speculative decoding accelerates inference by drafting and verifying multiple tokens in parallel, existing methods operate at the token level and ignore semantic equivalence (i.e., different token sequences expressing the same meaning), leading to inefficient rejections. We propose **SemanticSpec**, a semantic-aware speculative decoding framework that verifies entire semantic sequences instead of tokens. SemanticSpec introduces a semantic probability estimation mechanism that probes the model's internal hidden states to assess the likelihood of generating sequences with specific meanings. Experiments on four benchmarks show that **SemanticSpec** achieves up to 2.7× speedup on DeepSeekR1-32B and 2.1× on QwQ-32B, consistently outperforming token-level and sequence-level baselines in both efficiency and effectiveness. Our code is available at https://github.com/Dongximing/semantic.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across various tasks (OpenAI, 2023; Bai et al., 2023; Touvron et al., 2023; Liu et al., 2024; Team et al., 2024). However, their deployment is constrained by high inference latency due to autoregressive decoding, where tokens are generated sequentially, and the latency grows with model size and output length. In particular, Large Reasoning Models (LRMs) such as OpenAI o1 (Jaech et al., 2024) and DeepSeek R1 (Guo et al., 2025) exacerbate this issue by producing lengthy chains of thought (CoT) for detailed intermediate reasoning before yielding the final output.

Speculative Decoding (Spector and Re, 2023; Xia et al., 2024) has emerged as a promising approach to accelerate LLM inference. It drafts multiple future tokens and then verifies them in parallel using the target LLM. It only accepts those that meet its verification criterion to guarantee output quality. However, most existing methods operate at the token level and overlook semantic equivalence, where different token sequences that convey the same meaning (e.g., "The capital of France is Paris." vs. "Paris is the capital of France.") (Kuhn et al., 2023). This oversight leads to unnecessary rejections of semantically valid drafts, and significantly reduces the efficiency of speculative decoding (as shown in Section 5.1). Our pilot study shows that in 56.5% of the studied cases, the draft and target models agree on the reasoning intent, while traditional speculative decoding triggered a false rejection due to minor token-level differences A.1. Although recent work on Large Reasoning Models (LRMs) explores speculating reasoning steps instead of individual tokens, these approaches typically rely on LLM-as-a-judge evaluations, which are often biased and unreliable (Gu et al., 2024; Li et al., 2025; Thakur et al., 2024), and results in low performance.

To address this limitation, we propose semantic-aware speculative decoding (**SemanticSpec**), a novel framework that elevates speculative decoding from the token level to the sequence level and handles semantic-equivalence. **SemanticSpec** operates over entire semantic sequences, capturing higher-level meaning rather than surface token patterns. The key challenge lies in estimating how likely a model is to generate any sequence conveying a particular meaning, which cannot be directly accessed from token probabilities. To overcome this, **SemanticSpec** introduces a novel semantic probability (i.e., the likelihood of generating any sequence that expressing a particular semantic meaning) estimation mechanism by probing the

* Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 2836–2851
October 24-29, 2026 ©2026 Association for Computational Linguistics

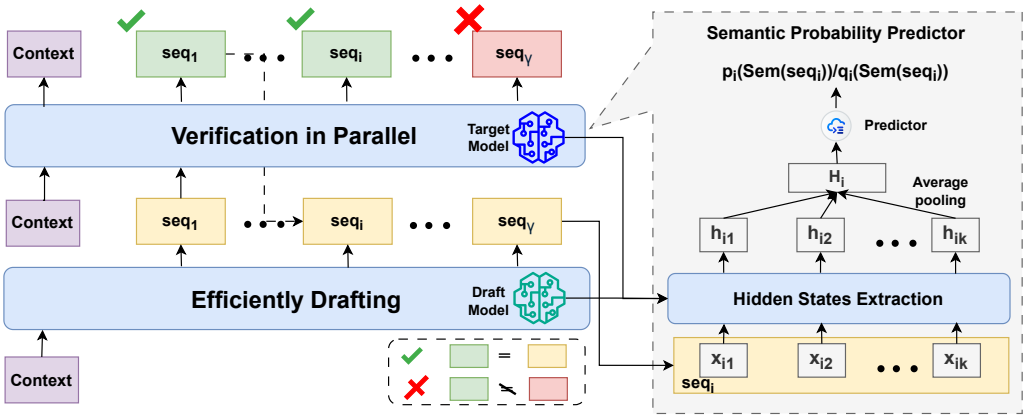

Figure 1: semantic-aware speculative decoding. Instead of drafting and verifying token-by-token, semantic-aware speculative decoding drafts and verifying sequence-by-sequence.

internal hidden states of LLMs during the verification. Specifically, we develop a semantic probability predictor that learns to map the model's internal representations when verifying a sentence, to the likelihood of producing the particular meaning expressed by the sentence. This predictor enables **SemanticSpec** to infer semantic-aware likelihoods beyond surface-level token distributions.

To evaluate the effectiveness of **SemanticSpec**, we compare various token-level and sequence-level baselines (e.g., SpecReason (Pan et al., 2025), Speculative Thinking (Yang et al., 2025), and SpecSampling (Yang et al., 2025)) on four benchmarks (e.g., MATH-500 and GPQA-D) and two pairs of LLMs. Our empirical results show that **SemanticSpec** accelerates the target models, DeepSeekR1-32B and QwQ-32B, by an average of 2.7× and 2.1×, respectively, and consistently outperforms both studied sequence-level and token-level baselines in both effectiveness and efficiency. For instance, **SemanticSpec** further boosts throughput (i.e., Token Per Second) by 1.67X on the DeepSeekR1-32B and 2.66X on the QwQ-32B compared to the standard token-level speculative decoding.

2 Background and Related Work

2.1 Speculative Decoding

Speculative Decoding (SD) is a technique to accelerate LLM inference without changing the model's final output distribution in theory (Leviathan et al., 2023a; Xia et al., 2024). Instead of querying the target LLM M_p for every token, a smaller and faster draft model M_q (e.g., a distilled or reduced-parameter version) speculatively generates a short token sequence; M_p then verifies this draft in one parallel forward pass. If the draft is correct, multiple tokens are accepted for the cost of a single forward pass.

The process for generating one block of tokens in basic speculative decoding (Spector and Re, 2023) is defined as follows:

Step 1: Drafting. Given a context sequence, the draft model (M_q) runs autoregressively for γ steps to propose a draft token sequence $\tilde{x}_{1:\gamma} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_\gamma)$. This is done by sampling from the draft model's distribution at each step: $\tilde{x}_i \sim M_q(\cdot \mid \text{context}, \tilde{x}_{1:i-1})$

Step 2: Parallel Verification. The target model M_p takes [context, $\tilde{x}_{1:\gamma}$] and performs a **single, parallel forward pass**, producing distributions $p_{1:\gamma}$ for all draft positions: $p_{1:\gamma} = M_p(\cdot \mid \text{context}, \tilde{x}_{1:\gamma})$.

Step 3: Acceptance & Correction The draft tokens are now verified sequentially. It accepts draft tokens until the first mismatch is found. The criterion for accepting the i-th draft token $\tilde{x}_i$ is: If $p_i(\tilde{x}_i) \geq q_i(\tilde{x}_i)$, accept $\tilde{x}_i$. Otherwise, reject $\tilde{x}_i$ and resample a corrected token x_i from an adjusted distribution ($norm(max(0, p(x) - q(x))$). It discards all remained tokens and restarts the process from Step 1.

2.2 Related Work and Their Limitations

2.2.1 Token-level Speculative Decoding

The current token-level speculative decoding approaches fall into two families:

Improving Drafting. Draft quality strongly affects SD efficiency. Existing work follows two paradigms: *independent drafting* and *self-drafting*. Independent drafting employs a smaller, separate model to propose candidate tokens for verification

by the target LLM (Xia et al., 2022; Leviathan et al., 2023b; Spector and Re, 2023; Sun et al., 2023). It often uses reduced-size models from the same family to accelerate larger ones without extra training. In contrast, self-drafting reuses the target model to generate drafts via internal sampling or parallel decoding (Stern et al., 2018; Cai et al., 2024; Yang et al., 2023; Zhang et al., 2023; Hooper et al., 2025). For instance, Medusa (Cai et al., 2024) attaches lightweight FFN heads for parallel token generation. **Enhancing Verification.** Each decoding step verifies drafted tokens in parallel to ensure consistency with the target model. Early methods used strict greedy verification that guaranteed lossless outputs (Stern et al., 2019; Xia et al., 2022) but frequently reject good drafts and limit speedup. Later work introduced more flexible criteria, e.g., top-k matching (Xia et al., 2022), rollback strategies (Kim et al., 2023), and token-tree verification (Miao et al., 2023), to improve acceptance and efficiency.

However, existing work either in improving drafting and verification remains token-level, overlooking semantic equivalence between drafts and target outputs, as discussed in Section 1, which still potentially reject good candidates and leads to low acceptance rate.

2.2.2 Sequence-level Speculative Decoding for Large Reasoning Models

With the emergence of large reasoning models (LRMs), several approaches have been developed to speculate reasoning steps (i.e., sequences expressing reasoning steps), which consider the semantic equivalence rather than individual tokens (Pan et al., 2025; Yang et al., 2025; Wang et al., 2025; Liao et al., 2025). For example, Pan et al. proposed SpecReason (Pan et al., 2025), which employs a draft model to perform speculative reasoning (e.g., reasoning steps) and prompts the target model to score the reasoning quality on a 1–10 scale for verification. Similarly, Yang et al. introduced Speculative Thinking (Yang et al., 2025), which leverages the occurrence of reasoning-supportive tokens, such as "wait" or "alternatively", as cues to delegate the remaining reasoning process (e.g., reasoning steps) to a larger mentor model. However, existing approaches mainly rely on LLM-as-a-judge evaluations or hard-coded lexical signals (e.g., "wait," "alternatively") for verification, both of which are prone to inaccuracies, as prior studies have shown that LLM-as-a-judge can be biased and unreliable (Gu et al., 2024; Li et al., 2025; Thakur et al., 2024). In contrast, our method leverages the internal states of the target model to measure the alignment between the draft and target reasoning processes.

3 Methodology

We introduce Semantic-Aware Speculative Decoding (**SemanticSpec**), a new paradigm that reconceptualizes speculative decoding at the level of semantic meaning, rather than individual tokens. While traditional methods sequentially draft and verify token-by-token, **SemanticSpec** drafts and verifies sequences, meanwhile handling semantic-equivalence robustly. Unlike token-level approaches, where token probabilities are directly accessed from LLMs, determining the probability of generating any sequence that conveys a specific semantic meaning (i.e., called ***semantic probability***) is challenging. Therefore, to effectively verify whether the drafted sequences align with the target model's true intent, **SemanticSpec** must estimate the semantic probability. We overcome this by designing a novel semantic probability predictor that infers these probabilities by probing the LLM's internal representations (hidden states). To train the predictor, we prompt LLMs with diverse inputs and generate multiple outputs and their hidden states. We then cluster outputs by semantic meaning and estimate their semantic probability.

3.1 Sematic-Aware Speculative Decoding

Algorithm 1 outlines the process of Semantic-Aware Speculative Decoding implemented by our approach (**SemanticSpec**). The algorithm consists of three main stages: (1) **sequence drafting**, (2) **semantic-aware sequence verification**, and (3) **acceptance decision**.

In sequence drafting, the draft model M_q speculatively generates up to γ candidate sequences $\tilde{s}_{n+1}, \ldots, \tilde{s}_{n+\gamma}$ based on the current context $s_{1:n}$. Along with these drafted sequences, the hidden representations $\{h^q_{n+1}, \ldots, h^q_{n+\gamma}\}$ of the draft model are extracted to capture the internal state associated with each generated sequence (line 4). In this study, we use `\n\n` as split to separate sequences.

After drafting, the target model M_p verifies the drafted sequences by processing them in parallel to obtain their corresponding hidden states $\{h^p_{n+1}, \ldots, h^p_{n+\gamma}\}$ (line 5). Later, we use semantic probability predictors—$Predictor_q$

Algorithm 1 Semantic-Aware Speculative Decoding

Require: Target model $\mathcal{M}_p$, draft model $\mathcal{M}_q$, input sequences $s_{1:t}$, number of sequences generated by draft model γ, probability predictor for draft and target model $Predictor_q$ and $Predictor_p$, max new tokens T
Ensure: Generated sequence $s_{1:T}$
#Initialize position counter and Maximum number of draft sequences
1: $n \leftarrow t$
2: **while** $len(s_{1:n}) < T$ **do**
3: $(\tilde{s}_{n+1}, h^q_{n+1}), \ldots, (\tilde{s}_{n+\gamma}, h^q_{n+\gamma}) \leftarrow$ DRAFT$(s_{\leq n}, \mathcal{M}_q)$
#Extract hidden states in parallel from target model
4: $h^p_i \leftarrow \mathcal{M}_p(\tilde{s}_{n+i}|s_{\leq n+i-1})$ for $i = 1, \ldots, \gamma$
#Predict the semantic probability of draft model and target model
5: $q_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1}) \leftarrow Predictor_q(h^q_{n+i})$ for $i = 1, \ldots, \gamma$
6: $p_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1}) \leftarrow Predictor_p(h^p_{n+i})$ for $i = 1, \ldots, \gamma$
#Sequence by sequence verification
7: **for** $i = 1$ to γ **do**
8: **if** With probability $\min\left(1, \frac{p_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})}{q_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})}\right)$ **then**
9: Accept $\tilde{s}_{n+i}$
10: **else**
11: $s_{n+i} \leftarrow \mathcal{M}p(s|s_{1:n+i-1})$
12: **break**
13: **end if**
14: **if** EOS in s_{n+i} **then**
15: Return FirstTokens($s_{1:n+i}$,min(index(EOS),T)
16: **end if**
17: **end for**
18: $n \leftarrow n + i$
19: **end while**
20: Return FirstTokens($s_{1:n}$,T)

for the draft model and $Predictor_p$ for the target model to estimate the likelihood that each model would produce the semantic meaning expressed by the given sequence $\tilde{s}_{n+i}$ under the same context (lines 6-7). The resulting probabilities $q_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})$ and $p_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})$ represent the semantic-aware confidence of each model in producing a particular semantic meaning, which forms the basis for subsequent acceptance decisions.

Lastly, for each drafted sequence, **SemanticSpec** determines whether to accept it based on the agreement between the draft and target models. Specifically, a drafted sequence $\tilde{s}_{n+i}$ is accepted with probability $\min\left(1, \frac{p_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})}{q_i(Sem(\tilde{s}_{n+i})|s_{\leq n+i-1})}\right)$ (lines 9-11). Otherwise, the algorithm reverts to the target model M_p to regenerate the next sequence from the last accepted context, ensuring correctness while maintaining efficiency (lines 12-16). This verification–acceptance loop continues until the max token number or EOS is met.

3.2 Predicting Semantic Probability by Probing Hidden States of LLMs

As motivated in Section 2, one challenge is that an LLM may produce different sequences to express the same meaning (i.e., semantic equivalence) (Kuhn et al., 2023). Directly estimating the likelihood of a sequence of tokens (i.e., lexical representation) is problematic. In almost all applications, we care about meaning rather than lexical representation. Therefore, to overcome this challenge, we convert the problem of estimating the likelihood of generating a specific sequence to estimating the probability of generating a semantic meaning expressed by the sequence, i.e., semantic probability. Formally, given a context C and a target LLM $\mathcal{M}$, our goal is to construct a predictor to estimate the likelihood of generating the semantic meaning expressed in the sequence s generated by $\mathcal{M}$ $p(Sem(s)|[C, s], \mathcal{M})$, where $Sem(s)$ is the semantic meaning expressed by s. Therefore, the semantic probability is an aggregated probability can be calculated:

$$p(Sem(s)|[C, s], \mathcal{M}) = \sum_{s_i \in \text{Sequences}(S)} p(s_i|C, \mathcal{M})$$

,where Sequences(S) is the set of all possible token sequences that express the semantic meaning $Sem(s)$.

Intuitively, the semantic probability for a given context could be empirically estimated through a sampling approach: generating a large number of model outputs, clustering those outputs based on their semantic equivalence, and then using the relative frequency of each cluster as its probability estimate. However, this sampling during inference is cost inefficiency and violates the objective of speculative decoding. Prior studies demonstrate that the internal states of an LLM can indicate the confidence/uncertainty of output generated the LLM (Kuhn et al., 2023; Azaria and Mitchell, 2023; Kossen et al., 2024). To assess whether the internal states of LLMs can effectively reflect semantic probabilities in our setting, we examine relationship between internal states and semantic probability. As shown in Figure 2, our results reveal a pronounced clustering effect, where hidden states are organized into distinct regions according to their semantic probability levels. In other word, hidden states can effectively indicate the probability.

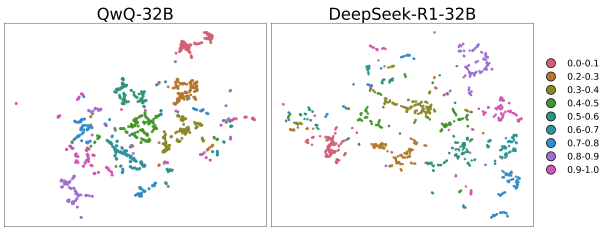

Figure 2: The distribution of hidden states with a various range of semantic probability in two-dimension. For each 0.1 probability interval, we randomly sample hidden state vectors, then project them into two dimensions using UMAP.

Therefore, we develop a semantic probability predictor that, by probing the internal states of the LLM without the need for sampling (see more details in Section 3.2.1). More importantly, the predictor can be trained offline and enables efficient semantic-aware verification during decoding. Below, we elaborate on the predictor and its construction process in detail.

3.2.1 Offline Semantic Probability Predictor Training

To train the semantic probability predictor, we first need to construct training data $\mathcal{D} = \{(H_{ij}, p(Sem(s_i)|[C_j, s_i], \mathcal{M}))\}_{i=1:N, j=1:M}$, which is a set of pair of semantic probability $p(Sem(s_i)|[C_j, s_i], \mathcal{M}))$ and its corresponding hidden states, where $s_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{ik})$ is a sequence that is verified by the LLM $\mathcal{M}$ given the input context C_j. The hidden states H_{ij} is the aggregated representation derived from when the sequence s_i goes through the model. More specifically, we collect the hidden states of the model when each token goes through the model. Then to aggregate the hidden states from each token and obtain a fixed-length representation for a sequence, we apply average pooling over the token-level hidden states for aggregation. In this study, we use the hidden states from all layers of $\mathcal{M}$, which has been demonstrated superior than that from the last layer (see more details in Section 5.2).

To construct the training data, we prompt the LLM $\mathcal{M}$ with various contexts (e.g., original query + multiple reasoning steps). For a specific context input C_j, we perform the following steps to estimate their semantic probability by following prior study (Kuhn et al., 2023):

Step 1. Generation: Generating N output sequences $\{s_1, \ldots, s_N\}$ using the large language model given a context C_j.

Step 2. Clustering: Clustering sequences that share the same meaning using the *bi-directional entailment* algorithm (Kuhn et al., 2023). Two sequences, s and s', are considered semantically equivalent if and only if each entails the other (i.e., they logically imply one another). For example, "The capital of France is Paris." entails "Paris is the capital of France." because both express the same proposition. In our implementation, we employ the `DeBERTa-large` model to perform entailment detection. For each pair of sampled sequences s and s', we check whether the concatenation of the context and s can be inferred from the concatenation of the context and s', and vice versa. Specifically, we concatenate each question–answer pair, then combine both sequences with a special separator token. The DeBERTa model classifies this composite input into one of three categories: *entailment*, *neutral*, or *contradiction*. Two sequences are deemed **semantically equivalent** if and only if both directional inferences result in *entailment*. We cluster a pair of two sequences together if they are semantically equivalent.

Step 3. Probability estimation: After clustering, we obtain a set of semantic clusters $\{C_1, C_2, \ldots, C_K\}$, where each cluster contains sequences that convey the same meaning. We then estimate the probability of generating a specific semantic meaning by counting the size of the cluster. More specifically, for any sequence $s_i \in C_k$, its semantic probability can be calculated:

$$p(Sem(s_i)|[C_i, s_i], \mathcal{M}) = \frac{|C_k|}{N}$$

Suppose there are M contexts and for each context, we generate N outputs, then the size of the training data is MN. Once we construct the training data, we train a regression model, which takes the hidden states as the input and output their corresponding semantic probability. We deploy a multi-layer perceptron with three layers as our regression model. Note that we train two predictors, one for the target model and one for the draft model.

3.2.2 Online Semantic Verification

During online inference, the trained *semantic probability predictor* verifies whether a drafted sequence aligns with the target model's intent. For each drafted sequence, we extract its hidden representation and predict the semantic probability using the predictor for draft model. The same sequence is

processed by the target model to obtain corresponding hidden states, and the probability is predicted. The acceptance of a drafted sequence is determined as shown in Algorithm 1, where a higher ratio indicates stronger semantic agreement. This semantic verification captures semantic consistency between models, allowing speculative decoding to remain robust to lexical variations while improving efficiency.

4 Experimental Settings

4.1 Models

To verify both the effectiveness and efficiency of our approach, we conduct experiments with two target models, **DeepSeek-R1-32B** and **QwQ-32B**, while using **DeepSeek-R1-1.5B** as the draft model.

4.2 Datasets and Metrics

We evaluate our approach on four benchmarks: **MATH-500** (Hendrycks et al., 2021), **AIME24**[1], **AMC23**[2], and **GPQA-D** (Rein et al., 2024). These datasets cover diverse domains of mathematical and scientific reasoning—ranging from competition-level math problems requiring multi-step deduction to expert-level science questions assessing factual and analytical understanding.

We use three primary metrics: **Pass@1**, **Latency**, and **Token Per Second (TPS)** by following prior studies (Yang et al., 2025; Pan et al., 2025; Liao et al., 2025; Yan et al., 2024). The Pass@1 metric measures the proportion of correctly generated answers when generating one candidate solution, reflecting the model's reasoning and factual reliability. The latency quantifies the total inference latency, including both drafting and verification stages, to evaluate the overall efficiency of speculative decoding. TPS measures the throughput of LLMs and counts the number of output tokens per second (TPS).

4.3 Baselines

To evaluate the effectiveness of our approach, we compare our proposed approach with various recent approaches. **SpecReason** (Pan et al., 2025) performs speculative reasoning with a lightweight model, using the target model's verification score to validate and refine the generated outputs. **Speculative Thinking** (Yang et al., 2025) leverages the tendency of reasoning-supportive tokens (e.g., "wait") to follow structural delimiters and the stronger reflective control of larger models to reduce backtracking and enhance reasoning quality. **SpecSampling** (Leviathan et al., 2023b) is standard token-level Speculative decoding that accelerates the target LLM's inference without changing its output distribution. Collectively, these baseline methods provide a comprehensive evaluation, covering both segment-level and sentence-level speculative decoding.

4.4 Training and Implementation Details

In our experiments, we use the **SimpleScaling-S1K** dataset (Muennighoff et al., 2025) to train semantic probability predictors, which contains 1,000 high-quality question–answer pairs covering both mathematical and scientific reasoning chains. We use the science-related pairs (i.e., around 400K samples) to train the predictors for GPQA-D, and math-related pairs to train the predictors for math datasets (Math-500, AIME, AMC23). We segment each reasoning chain using the delimiter '`\n\n`' and generate 20 responses per example to enhance dataset diversity. Meanwhile we collect the corresponding internal states of LLMs.We eventually collect 180K and 220K samples from the science and math domains, respectively. This offline collection took approximately 8 hours for the target model and 2 hours for the draft model. For clustering, we employ the **DeBERTa-Xlarge-MNLI** model to perform iterative pairwise clustering of samples. For training, we adopt a simple three-layer feed-forward neural network (**FNN**) as the probe for both the target and draft models. Specifically, the target model predictor uses layer dimensions of 5120–2048–512–1, while the draft model predictor uses 1536–512–256–1. We utilize ReLU for all intermediate layers and a Sigmoid activation for the final output layer. The learning rate is set to $1e-3$, and we use the AdamW optimizer (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We use Mean Squared Error (MSE) as our loss function. All experiments are conducted on a server with two **80 GB GPUs** and are implemented using the **SGLang** framework. Data labeling (clustering via DeBERTa) took roughly 2 hours, while the FNN predictor was trained in just 20 minutes.

[1] https://huggingface.co/datasets/Maxwell-Jia/AIME_2024

[2] https://huggingface.co/datasets/zwhe99/amc23

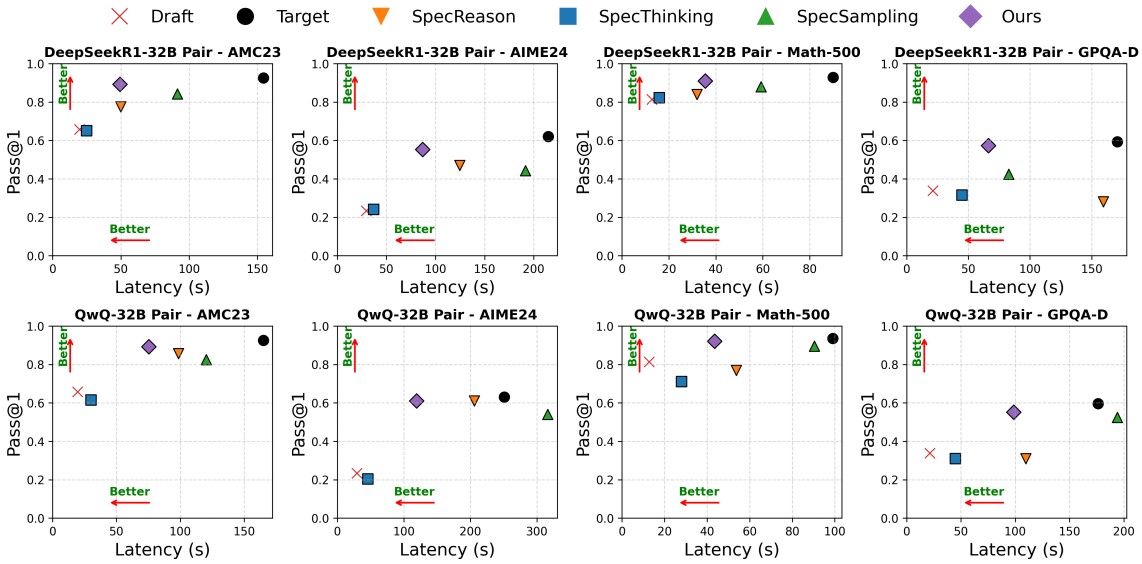

Figure 3: Comparison of pass@1 and latency across different speculative inference schemes and model pairs. Each point represents a decoding approach. The top row shows results for the target model DeepSeekR1-32B, and the bottom row for QwQ-32B.

Method	AMC23				AIME24				Math-500				GPQA-D			
	Pass@1	TPS	Latency	Ratio	Pass@1	TPS	Latency	Ratio	Pass@1	TPS	Latency	Ratio	Pass@1	TPS	Latency	Ratio
Draft: DeepSeekR1-1.5B Target: DeepSeekR1-32B																
Random	0.775	115.90	49.98	0.27	0.470	80.94	124.61	0.28	0.840	109.20	31.92	0.20	0.308	65.68	88.91	0.31
Last-hidden	0.865	**134.31**	53.69	0.17	0.400	**129.65**	**82.03**	0.14	0.874	**145.56**	**32.57**	0.23	0.557	84.61	69.60	**0.21**
Ours	**0.892**	120.55	**49.29**	**0.15**	**0.553**	116.27	**86.75**	**0.12**	**0.910**	117.33	**35.47**	**0.14**	**0.573**	**85.28**	**66.28**	0.22
Draft: DeepSeekR1-1.5B Target: QwQ-32B																
Random	0.758	54.12	133.36	0.31	0.460	53.63	202.87	0.29	0.814	58.11	80.90	0.28	0.286	52.04	91.49	0.21
Last-hidden	0.825	**146.09**	**51.87**	0.31	0.466	**139.84**	**81.24**	0.29	0.892	**144.79**	**36.32**	0.32	0.525	49.68	156.00	0.33
Ours	**0.892**	85.03	75.42	**0.23**	**0.610**	86.88	119.06	**0.21**	**0.922**	97.00	43.53	**0.19**	**0.552**	**73.36**	**98.68**	**0.21**

Table 1: **Performance comparison** of predictors across four reasoning benchmarks (**AMC23**, **AIME24**, **Math-500**, **GPQA-D**) in terms of the studied metrics. Each cell reports accuracy (**Pass@1**), average generation speed (**TPS = Output Tokens / Latency**), and inference time (**Latency**) and we also record the proportion of tokens that were generated by the target model (**Ratio**).

5 Results

5.1 Comparison with Baselines

SemanticSpec (ours) consistently achieves higher pass@1 compared to baselines and competitive efficiency across all datasets and model settings as shown in Figure 3. Although SepcThink achieves similar latency as a small model, it also suffers from the lowest pass@1 among all baselines. The reason is that it always accepts the drafted sequences from draft models, while overlooking the refinement from the target model. Compared with SpecReason and SpecSampling, **SemanticSpec** consistently achieves higher pass@1 and lower latency across all datasets on both QwQ-32 and DeepSeekR1-32B. Compared with token-level speculative decoding (i.e., SpecSampling), all other sequence-level speculative decoding approaches often achieve lower latency, particularly **SemanticSpec**. For instance, **SemanticSpec** outperforms SpecSampling by 17.3% and 7.3% in terms of pass@1, meanwhile reducing substantial inference time by 40.2% and 45.3% on average on DeepSeekR1-32B and QwQ-32B, respectively. However, **SemanticSpec** further boosts throughput by 1.67X on the DeepSeekR1-32B and 2.66X on the QwQ-32B compared to SpecSampling. More details can be found in Table 5 in the Appendix.

5.2 Comparison with Original Target and Draft Models.

Compared with the target model, SemanticSpec achieves an average speedup of 2.7× and 2.1× on DeepSeekR1-32B and QwQ-32B, respectively, while maintaining comparable pass@1 performance (an average of 4.9% and 3.9% pass@1 drops). Figure 3 illustrates the effectiveness and efficiency of **SemanticSpec** compared with base-

	Science ⇒ Math									Math ⇒ Science		
	AMC23			AIME24			Math-500			GPQA-D		
Method	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**
Within-domain	**0.892**	120.55	**49.29**	**0.553**	116.27	**86.75**	0.910	**117.33**	35.47	**0.573**	**85.28**	**66.28**
Cross-domain	0.875	**125.57**	49.90	0.533	**122.64**	82.38	**0.911**	139.23	**31.72**	0.552	70.09	82.14

Table 2: Results of using cross-domain semantic probability predictor. conducted with the **DeepSeek-R1-1.5B** → **DeepSeek-R1-32B**.

line methods across four datasets and two target models. **SemanticSpec** consistently accelerates inference, achieving speedups ranging from 2.48× to 3.13× for DeepSeekR1-32B and 1.79× to 2.28× for QwQ-32B. The higher speedup on DeepSeekR1-32B is attributed to using DeepSeekR1-1.5B as the draft model, which belongs to the same model series as the target. Furthermore, **SemanticSpec** achieves performance comparable to the target models, with an average performance drop (i.e., pass@1) of 4.9% and 3.9% on DeepSeekR1-32B and QwQ-32B, respectively, compared to the target models. However, **SemanticSpec** achieves a range of 2.83X and 2.72X higher TPS than targets models on DeepSeekR1-32B and QwQ-32B, respectively.

Compared with the smaller draft model, our approach markedly improves accuracy while shortening reasoning length. Smaller drafts tend to over-generate repetitive reasoning patterns when they fail to reach a valid intermediate state, producing long, semantically redundant traces that seldom yield the correct answer. **SemanticSpec** guides the draft to produce meaningfully distinct steps aligned with the target model's reasoning distribution, thereby reducing repetition and enabling the verifier to accept more tokens per step. This yields several-fold accuracy gains over the draft-only baseline (e.g., $+161.8\%$ on AIME24) together with shorter reasoning paths.

5.3 Ablation Analysis.

To evaluate the effectiveness of our semantic probability predictor, we conduct an ablation study by comparing it with two variants: (1) **Random**, where the semantic probabilities for both the target and draft models are assigned random values uniformly sampled from the range [0, 1], while keeping all other components of **SemanticSpec** unchanged; and (2) **Last-hidden**, where the predictor utilizes only the hidden states from the final layer rather than aggregating representations from all layers.

Table 1 presents the comparison results. Compared to Random, our predictor achieves substantial improvements across all evaluation metrics on all models, demonstrating its effectiveness in accurately routing inference between the target and draft models. Compared with Last-hidden, our predictor consistently yields higher Pass@1 scores across all datasets and models—for example, improving Pass@1 from 0.400 to 0.553 and from 0.466 to 0.610 on AIME24 for DeepSeekR1-32B and QwQ-32B, respectively. However, Last-hidden has the highest throughput (although it use the target model to generate more tokens), as it only accesses the final layer's hidden states, whereas our predictor incurs additional computational overhead by extracting features from all layers. This observation highlights a **trade-off between effectiveness and efficiency: incorporating multi-layer information enhances predictive accuracy but reduces throughput.** Practitioners can therefore choose between the two designs based on their performance requirements and latency constraints in practical deployment scenarios.

5.4 Generalization of Semantic Probability Predictor

To evaluate the generalization ability of the proposed semantic probability predictor, we conduct cross-domain prediction experiments between the math and science domains. Specifically, we use the predictor trained on science data (i.e., the science subset of SimpleScaling-S1K) to predict the probabilities for math tasks (AMC23, AIME24, and Math-500), and the predictor trained on math data to predict the probabilities for science tasks (GPQA-D).

Table 2 reports the results for both within-domain (predictor and tasks from the same domain) and cross-domain settings on DeepSeek-R1-32B. Compared to within-domain prediction, cross-domain performance shows a slight decrease on AMC23 (from 0.892 to 0.875), AIME24 (from 0.553 to 0.533), and GPQA-D (from 0.573 to 0.552), while remaining comparable on Math-500

(from 0.910 to 0.911). **These results suggest that the semantic probability predictor based on internal states of models generalizes well across domains**.

6 Discussion

6.1 Overhead

Across all tasks and model pairs, the semantic probability prediction adds only 0.12 - 0.22 seconds to the decoding process, less than 0.4% of the inference time. In terms of memory usage, the target model and draft model occupy 73.7 GB and 26.3 GB, respectively, while the two semantic probability predictors together require only 0.58 GB. Therefore, the predictors contribute less than 1% additional memory overhead relative to the models themselves.

6.2 Robustness to semantic clustering noise

To evaluate the system's robustness against potential errors in semantic clustering (e.g., DeBERTa-based biases), a noise injection stress test was conducted on AMC23 using DeepSeekR1-1.5B (Draft) and DeepSeekR1-32B (Target). Cluster assignments were manually corrupted by randomly reassigning $x\%$ of samples to incorrect clusters. The predictor remains highly effective under 10% noise, showing only a marginal decrease in Pass@1 (from 0.892 to 0.866). Performance significantly degrades at 30% noise, falling below the "Random Assignment" baseline (0.775). This indicates that while the model is resilient to minor semantic inaccuracies, the predictor relies on a minimum threshold of cluster purity to outperform a naive random baseline. Interestingly, TPS remains high even as accuracy drops, suggesting the predictor still attempts to generate drafts, though their quality (Pass@1) suffers.

Table 3: Noise Injection Stress Test on AMC23

Noise	Pass@1	Latency (s)	TPS
0%	0.892	49.30	120.60
10%	0.866	56.86	145.39
30%	0.717	58.07	155.50
50%	0.700	61.61	148.25

7 Conclusion

This work introduces a semantic-aware speculative decoding that advances beyond token-level by capturing meaning rather than surface token patterns. Through a novel semantic probability estimation mechanism based on internal model representations, **SemanticSpec** effectively identifies semantically valid sequences during verification. Our extensive experiments demonstrate that it substantially accelerates inference while preserving semantic fidelity, and achieves up to 2.7× speedup over standard decoding. These results highlight the potential of semantic-aware speculative decoding as a new direction for efficient LLM inference.

8 Limitations

In this paper, we evaluated our semantic-level speculative decoding on two pairs of open source models, and four widely used benchmarks. Our findings might be not generalized to other models, we encourage future research to measure the effective of our approach on more models.

References

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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. *arXiv preprint arXiv:2401.10774*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Yakun Sophia Shao. 2025. Speed: Speculative

pipelined execution for efficient decoding. In *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*, pages 19–32. Springer.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36:39236–39256.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *Preprint*, arXiv:2406.15927.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

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

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

Qingquan Li, Shaoyu Dou, Kailai Shao, Chao Chen, and Haixiang Hu. 2025. Evaluating scoring bias in llm-as-a-judge. *arXiv preprint arXiv:2506.22316*.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5):1.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Benjamin Spector and Chris Re. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

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

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. 2024. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. *arXiv preprint arXiv:2406.12624*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jikai Wang, Juntao Li, Jianye Hou, Bowen Yan, Lijun Wu, and Min Zhang. 2025. Efficient reasoning for llms through speculative chain-of-thought. *arXiv preprint arXiv:2504.19095*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2022. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint arXiv:2203.16487*.

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. *arXiv preprint arXiv:2401.07851*.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. 2024. Decoding speculative decoding. *arXiv preprint arXiv:2402.01528*.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. 2023. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*.

A Appendix

A.1 Empirical evidence of semantically equivalent but lexically different sequence rejection

To quantify the frequency of false rejections, where sequences are semantically equivalent but lexically divergent, we conducted a controlled analysis using 100 samples from SimpleScaling-S1K dataset, where we collected training data.

We segmented reasoning traces into discrete steps using as the delimiter. For each prefix (original query + multiple reasoning steps), we generated the subsequent reasoning step under two conditions:

- **Strict Lexical Verification:** Traditional speculative decoding (rejection on any token mismatch).
- **Unconstrained Draft Generation:** The draft model generates the sequence without any target model intervention or rejection

We collected 20,496 pairs of generated outputs for comparison. We then evaluated the semantic equivalence (using the bi-directional entailment algorithm described in Section 3) of these pairs to identify "false rejections".

45.8% (9,396/20,496) of the output pairs have the same semantic meaning, and among them, 56.5% (5,315/9,396) pairs have the same semantic meaning but lexical mismatches, where at least one token was rejected in Strict Lexical Verification. In other words, in 56.5% of cases, the draft and target models agree on the reasoning intent, while traditional speculative decoding triggered a false rejection due to minor token-level differences. Crucially, in these false rejection instances, we observed an average of 3.25 tokens rejected per sequence. **This high frequency of false rejections directly translates to a latency bottleneck, as the target model is forced to re-generate sequences that were already semantically correct. This validates the necessity of SemanticSpec in bypassing strict lexical constraints to unlock higher effective throughput.**

A.2 Impact of drafted sequence length γ

To further investigate the impact of drafted sequence length (i.e., impact of hyper-parameter γ), we evaluate the performance of our semantic-level speculative decoding under different values

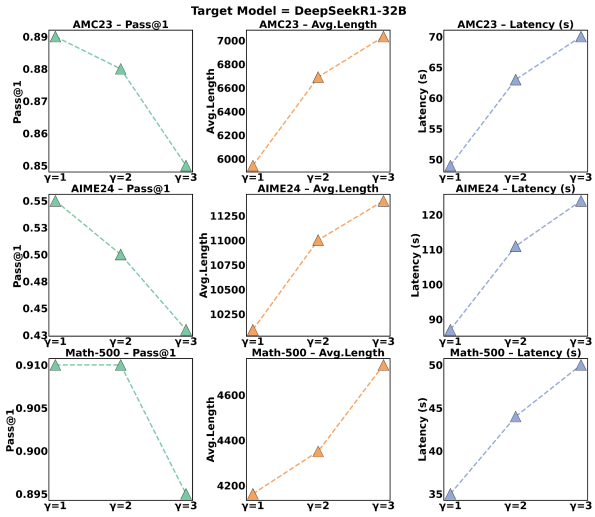

Figure 4: Performance comparison under different size of drafted sequence ($\gamma = 1, 2, 3$).

($\gamma \in 1, 2, 3$) across all datasets. As observed in Figure 4, as the length of the drafted sequence increases, both the effectiveness and efficiency of our semantic-level speculative decoding consistently decreases across all datasets, which suggests to use small γ in practice. This is reasonable, since we use '\n\n' to split sequences, a large γ usually results in a long sequence which probably expresses various semantic meanings, which reduce the accepted rate of the drafted sequences.

A.3 Impact of sequence splitting strategy

In default setting, we use **'\n\n'** as the split to separate sequence, which probably lead to relatively coarse-granularity. To investigate the impact of smaller granularity, we also investigate to use **sentence-ending punctuation** (such as **.?!**) as the split. As shown in Figure5, increasing the granularity consistently improves pass@1 across all datasets slightly, while TPS decreases significantly. Finer-grained split triggers more frequent validations, resulting in a more constrained generation process.

A.4 Generalization to other platforms

To demonstrate the generalization ability of our method, we also applied it within the Hugging Face framework. Table 4 shows that our method reduces output length while maintaining performance competitive with the target model, indicating that it can be applied across a variety of frameworks as long as the hidden states are accessible.

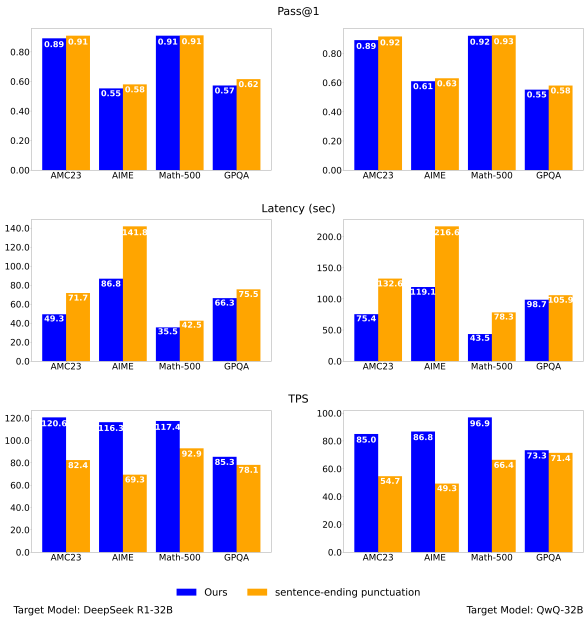

Figure 5: The impact of different splitting strategies ours vs. sentence-ending punctuation.

	AIME24			Math-500		
Method	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**
Draft Model	0.211	21.99	519.35	0.738	40.49	127.73
Target Model	0.550	11.86	735.75	0.945	11.09	333.01
Ours	0.520	14.80	653.05	0.935	20.51	243.92

Table 4: The performance across two reasoning benchmarks (**AIME24**, **Math-500**). Each cell reports accuracy (**Pass@1**), updated throughput (**TPS / Latency** on Hugging Face.), and inference time (**Latency**).

A.5 Detailed results of the studied speculative decoding methods

Table 5 presents the detailed performance of the studied speculative decoding approaches in terms of Pass@1, TPS, and Latency.

	AMC23			AIME24			Math-500			GPQA-D		
Method	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**	**Pass@1**	**TPS**	**Latency**
Draft: DeepseekR1-1.5B Target: DeepseekR1-32B												
Draft Model	0.658	378.30	19.59	0.233	380.67	29.45	0.8142	380.51	12.70	0.338	371.93	21.30
Target Model	0.925	34.78	154.34	0.621	41.03	214.76	0.928	39.21	89.91	0.593	40.65	170.87
SpecReason	0.775	125.92	49.98	0.470	80.95	124.61	0.840	149.19	31.92	0.281	52.79	159.33
SpecThinking	0.650	**284.59**	**24.99**	0.241	**272.79**	**37.09**	0.823	**284.34**	**15.94**	0.315	**250.31**	**44.88**
SpecSampling	0.842	65.73	91.40	0.443	53.29	191.45	0.880	68.58	59.32	0.424	73.00	82.78
Ours	**0.892**	120.58	49.29	**0.553**	116.32	86.75	**0.910**	117.36	35.47	**0.5725**	85.30	66.28
Draft: DeepseekR1-1.5B Target: QwQ-32B												
Draft Model	0.658	378.30	19.59	0.233	380.67	29.45	0.8142	380.51	12.70	0.338	371.93	21.30
Target Model	0.925	40.62	165.02	0.63	40.62	250.75	0.935	40.71	99.04	0.5959	39.73	176.25
SpecReason	0.857	64.52	98.59	0.610	45.96	205.70	0.770	76.02	53.66	0.310	68.83	109.64
SpecThinking	0.614	309.55	30.20	0.204	285.20	45.87	0.710	312.77	27.99	0.310	250.31	44.88
SpecSampling	0.825	58.53	120.39	0.540	34.25	316.19	0.895	52.61	90.33	0.5250	37.59	194.01
Ours	**0.892**	**85.00**	**75.42**	**0.610**	**86.79**	**119.06**	**0.9217**	**96.95**	**43.53**	**0.552**	**73.32**	**98.68**

Table 5: Performance comparison of different methods across **Math-500**, **AMC23**, **AIME24**, and **GPQA-D**. The the cells achieving the best result among ours and baselines are marked as bold.

Appendix: Example from AMC23 Dataset

Question

Flora the frog starts at 0 on the number line and makes a sequence of jumps to the right. In any one jump, independent of previous jumps, Flora leaps a positive integer distance m with probability $\frac{1}{2^m}$. What is the probability that Flora will eventually land at 10? Write the answer as a simplified fraction $\frac{m}{n}$, find $m+n$.

Speculative Thinking(wrong!)

Alright, so I need to find the probability that Flora the frog will eventually land on 10 starting from 0. She can jump any positive integer distance m with probability $\frac{1}{2^m}$ each time, and each jump is independent.
First, note that once she jumps past 10, she can never return, since all jumps are to the right. Thus, the only way Flora can ever land on 10 is if the sum of her jumps equals exactly 10 at some point.
Let $P(n)$ be the probability that starting from position n, the frog will eventually reach 10. We have:

$$P(n) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(n+m),$$

with $P(10) = 1$ and $P(n) = 0$ for $n > 10$.
We can recursively compute backward:

$$P(9) = \tfrac{1}{2},\ P(8) = \tfrac{1}{2},\ \ldots,\ P(1) = \tfrac{1}{2}.$$

Then,

$$P(0) = \sum_{m=1}^{\infty} \frac{1}{2^m} P(m) = \frac{1}{2} \sum_{m=1}^{9} \frac{1}{2^m}.$$

The finite geometric series gives:

$$\sum_{m=1}^{9} \frac{1}{2^m} = \frac{1/2(1-(1/2)^9)}{1-1/2} = \frac{511}{512}.$$

Hence,

$$P(0) = \frac{1}{2} \times \frac{511}{512} = \frac{511}{1024}.$$

Therefore, the probability that Flora will eventually land on 10 is $\frac{511}{1024}$, so $m = 511$, $n = 1024$, and $m + n = 1535$.
</think>

Final Answer:

$$\boxed{\frac{511}{1024}}, \quad \boxed{m+n=1535}.$$

Speculative Sampling(9162 tokens, 102s)

Okay, so I have this problem where Flora the frog starts at 0 on the number line and jumps to the right. Each jump she makes is a positive integer distance m, and the probability of jumping $\frac{1}{2^m}$. I need to find the probability that she will eventually land on 10. The answer should be written as a simplified fraction $\frac{m}{n}$, and then I need to find $m+n$.
Hmm(Okay), let me think about how to approach (solve) this. It seems like a probability problem

involving (about) jumps on a number line. Since each jump is independent(uncorrelated), the probability of reaching 10 depends(relies) on the probabilities of all possible sequences of jumps that sum (total) up to 10. So(Thus), maybe(I) I can model (take)this as a recursive probability problem.
Let me denote $P(k)$ as the probability that Flora will eventually (land) land on position k. So, (Thus) I need (should) to find $P(10)$.
Since (Because) Flora can (could) jump any positive integer distance m with probability $\frac{1}{2^m}$, the probability of landing on position k can (could) be thought of as the sum of the probabilities of all(whole) possible ways to reach (find) k in one jump from some (any) position j, where j is less than k.
Wait(Hold), actually(that's), that's(likely) more like the probability of reaching k for the first time. But (However) the problem(question) says "eventually land at 10," which might (include) include(have) landing on 10 multiple times. But (Wait) I think (can) in this context, it's just about whether(if) she ever lands on 10, not (but) necessarily the first time. So, perhaps(maybe) it's the probability that (she) she reaches 10 in some finite number of jumps.
[...] Therefore, starting from 0, $P(0) = \frac{1}{2}$. Therefore, the probability is $\frac{1}{2}$, so $m = 1$, $n = 2$, and $m + n = 3$.
I think this is correct. Each position before 10 has a 50
Final Answer The probability is $\boxed{3}$.

SemanticSpec (6092 tokens, 64s)

Okay, so I have this problem about Flora the frog who starts at 0 on the number line and makes jumps only to the right. Each jump she makes is a positive integer distance m with probability $1/(2^m)$. I need to find the probability that she will eventually land exactly on 10. Hmm, that sounds like it involves some probability theory and maybe recursion or generating functions.
[...] But notice that the inner sum is $\sum_{k=0}^{\infty} P(k)\, 2^k\, x^{n-k}$. Wait, because for each k, it's multiplied by x^n and 2^k,so perhaps we can factor x^n as $x^{n-k} \cdot x^k$. Hmm, maybe not.
Wait, perhaps we can swap the order of summation on the right-hand side. Let's try: [...] Wait, but let me think again. If the frog can jump any positive integer distance, then the probability of eventually reaching 10 is 1/2? That seems high, but given that the frog can potentially jump any distance, the probability might be higher. Wait, let's think about the recurrence relation. For $n \geq 1$, $P(n) = \sum_{m=1}^{n} P(n-m) \cdot \frac{1}{2^m}$. So, if we assume that $P(k) = \frac{1}{2}$ for all $k \geq 1$, then: [...] In such a case, the probability of reaching state 10 is the same for all states n >=1, due to the symmetry in the jump probabilities. Wait, is that true?" In such a chain, since all jumps are to the right, and the frog can potentially jump over 10 in a single jump, the question is whether the frog will ever land exactly on 10. [...] Wait, but that seems a bit counterintuitive because, in reality, the frog has to make a finite number of jumps to reach 10, but the probability of making such jumps is decreasing exponentially. Alternatively, another way to think about it is that for each position $n \geq 1$, the probability of ever reaching n is $1/2$. That seems to be the case here because of the generating function.
So, perhaps the answer is 1/2 for P(10). So, $m = 1$, $n = 2$, $m + n = 3$.

$$\boxed{3}$$

</think>
Flora the frog starts at 0 on the number line and makes jumps to the right. Each jump distance m has a probability of $\frac{1}{2^m}$. We need to find the probability that Flora will eventually land on 10.
We denote $P(n)$ as the probability that Flora will eventually land on position n. We need to find $P(10)$.
The recurrence relation for $P(n)$ is:

$$P(n) = \sum_{m=1}^{n} \frac{P(n-m)}{2^m}$$

with the base case $P(0) = 1$.
We derived the generating function $G(x)$ for the sequence $P(n)$ and found that:

$$G(x) = \frac{2 - x}{2(1 - x)}$$

Expanding $G(x)$ as a power series, we found that:

$$G(x) = 1 + \frac{1}{2}x + \frac{1}{2}x^2 + \frac{1}{2}x^3 + \cdots$$

This expansion shows that for all $n \geq 1$, the coefficient of x^n is $\frac{1}{2}$. Therefore, the probability $P(n)$ for any $n \geq 1$ is $\frac{1}{2}$.
Thus, the probability that Flora will eventually land on 10 is $\frac{1}{2}$. The fraction $\frac{1}{2}$ is already in its simplest form, so $m = 1$ and $n = 2$. Therefore, $m + n = 1 + 2 = 3$.
The final answer is:

$$\boxed{3}$$

An example from the AMC23 dataset showing how **SemanticSpec**, **speculative sampling**, and **speculative thinking** produce different responses to the same input. In **speculative sampling**, the red word denotes the token replaced by its left neighbor. In contrast, **SemanticSpec** replaces entire sentences (red part) with the target model's output (green part) when the model judges them to be untruthful. From the example, we can see that **speculative sampling** is very restricted. However, **SemanticSpec** implicitly identifies abnormal sequences and replaces them more appropriately.