# Speculative Decoding with a Speculative Vocabulary

**Miles Williams**[1,2] **Young D. Kwon**[2] **Rui Li**[2]
**Alexandros Kouris**[2] **Stylianos I. Venieris**[2]

[1]University of Sheffield [2]Samsung AI Center, Cambridge, UK
**Correspondence:** mil.williams@samsung.com

## Abstract

Speculative decoding has rapidly emerged as a leading approach for accelerating language model (LM) inference, as it offers substantial speedups while yielding identical outputs. This relies upon a small draft model, tasked with predicting the outputs of the target model. State-of-the-art speculative decoding methods use a draft model comprising a single decoder layer and output embedding matrix, with the latter dominating drafting time for the latest LMs. Recent work has sought to address this output distribution bottleneck by reducing the vocabulary of the draft model. While this can improve throughput, it compromises speculation effectiveness when the target token is out-of-vocabulary. In this paper, we argue for vocabulary speculation as an alternative to a reduced vocabulary. We propose SpecVocab, an efficient and effective method that selects a vocabulary subset per decoding step. Across a variety of tasks, we show that SpecVocab can achieve a higher acceptance length than state-of-the-art speculative decoding method, EAGLE-3. Notably, this yields up to an 8.1% increase in average throughput over EAGLE-3.[1]

## 1 Introduction

Despite the remarkable capabilities of large language models (LMs) (Kamath et al., 2025; Yang et al., 2025; Agarwal et al., 2025), their autoregressive design continues to limit inference efficiency. Speculative decoding has emerged as a prominent approach to accelerate inference while maintaining identical outputs (Xia et al., 2024). Conventionally, speculative decoding combines the desired *target* model with a smaller *draft* model. The draft model rapidly generates a series of candidate tokens that are then verified in parallel via a single forward pass of the target model (Leviathan et al., 2023; Chen et al., 2023).

[1]https://github.com/SamsungLabs/SpecVocab

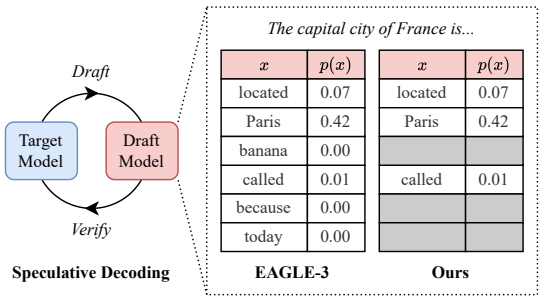



Figure 1: *Vocabulary speculation* accelerates speculative decoding by computing the output distribution for only a contextually relevant subset of the vocabulary.

Contemporary speculative decoding methods leverage lightweight draft models for efficient drafting (Miao et al., 2024; Cheng et al., 2024; Wertheimer et al., 2024; Li et al., 2024b,a, 2025b; Zhang et al., 2025). However, Zhao et al. (2025) recently identified that in the widely adopted EAGLE speculative decoding framework (Li et al., 2024a), the majority of drafting time is spent computing the output distribution over the target vocabulary. This presents a substantial problem, as LMs continue to require expansive vocabularies (Tao et al., 2024; Huang et al., 2025; Takase et al., 2025).

To mitigate the vocabulary bottleneck during drafting, recent work has sought to exploit the Zipfian distribution of natural language (Zipf, 1949). In theory, rare tokens are less likely to be predicted by the target model, so they can be excluded from the draft model vocabulary. FR-Spec (Zhao et al., 2025), EAGLE-3 (Li et al., 2025b), and VocabTrim (Goel et al., 2025) all use a fixed subset of the target model vocabulary to reduce the latency of the vocabulary projection. Nonetheless, when the next token falls outside of this subset, the current and subsequent draft tokens will be rejected, thereby eliminating the speedup from speculative decoding.

In this paper, we argue that speculative decoding should speculate not only on the next tokens, *but also on the output vocabulary* (Figure 1). In



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 40240–40254

contrast to earlier approaches, this allows the cost of computing the output distribution to be reduced, while better preserving the quantity of accepted draft tokens. Our core contributions are as follows:

1. We propose SpecVocab, an efficient method to predict a subset of the vocabulary that is contextually relevant to the next token.

2. Across a variety of tasks, SpecVocab achieves higher acceptance lengths than static vocabulary methods (*i.e.* EAGLE-3, FR-Spec, and VocabTrim). This enables up to 8.1% higher average throughput than EAGLE-3.

3. We implement and benchmark a custom kernel that accelerates logits computation for vocabulary subsets by up to 5×.

4. We empirically demonstrate that the reduced-vocabulary draft model training process introduced in EAGLE-3 (Li et al., 2025b) can unnecessarily harm draft model performance.

## 2 Related Work

**Vocabulary representations.** Recent LMs leverage increasingly large subword vocabularies. Early Transformer-based LMs such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) used vocabularies with 30K and 50K tokens, respectively. In comparison, more recent models such as OLMo 2 (Walsh et al., 2025) and Qwen3 (Yang et al., 2025) adopt larger vocabularies of 100K and 152K, respectively. Recent work has highlighted the performance benefits of pre-training LMs with such substantial vocabularies (Tao et al., 2024; Huang et al., 2025; Takase et al., 2025).

**Speculative decoding.** The core mechanism behind speculative decoding is to propose multiple tokens and continue generation from the longest correct prefix (Stern et al., 2018; Sun et al., 2021). This has been popularized as the *draft-then-verify* pattern, using an efficient draft model, while preserving the output distribution of the target model (Xia et al., 2023; Leviathan et al., 2023; Chen et al., 2023). Recent speculative decoding approaches have explored using auxiliary heads (Cai et al., 2024; Ankner et al., 2024), contextual embeddings (Gritta et al., 2025), intermediate hidden states (Cheng et al., 2024; Li et al., 2025b), and tree-like drafting (Spector and Re, 2023; Li et al., 2025b). In particular, the EAGLE series of speculative decoding methods (Li et al., 2024b,a, 2025b) have been widely adopted, both in popular inference engines (Kwon et al., 2023; Zheng et al., 2024) and at scale (Tang et al., 2025).

**Reduced-vocabulary draft models.** While speculative decoding methods have sought to maximize the efficiency of the draft models, the overhead from computing the output distribution over the model vocabulary has persisted. Zhao et al. (2025) first identified the output distribution bottleneck in speculative decoding draft models. They proposed FR-Spec, which prunes the embeddings for less common tokens, exploiting the long-tailed nature of natural language frequency distributions (Zipf, 1949). Independently, Goel et al. (2025) proposed VocabTrim, also pruning the output embedding matrix based on token frequency. In contrast to FR-Spec, which suggests using large-scale pre-training corpora (Soboleva et al., 2023) for token frequency computation, VocabTrim leverages synthetic data generated by the target model. Finally, EAGLE-3 (Li et al., 2025b) also adopts a reduced vocabulary based on synthetic data from the target model. However, the output embedding matrix is trained from scratch, rather than being pruned after training. Zhao et al. (2025) find that a vocabulary size of 32K provides the optimal throughput, with this vocabulary size also being used by EAGLE-3. However, a fixed vocabulary subset is inherently context-agnostic, leading to suboptimal performance when output tokens fall outside the subset. Our work addresses this through context-aware vocabulary speculation, selecting a relevant subset at each decoding step.

## 3 Vocabulary Speculation

### 3.1 Preliminaries

Speculative decoding accelerates autoregressive generation by introducing a lightweight draft model $q$ that generates candidate sequences, which are subsequently verified by the target model $p$. The draft model has an output embedding matrix (*i.e.* LM head) $\mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $|\mathcal{V}|$ is the vocabulary size of the target model and $d$ is the draft model dimensionality. At each decoding step $t$, the draft model produces logits $\mathbf{z}_t = \mathbf{U}\mathbf{h}_t$, where $\mathbf{h}_t$ is the final hidden state of the draft model. These logits are then used to form the next-token probability distribution $q(\mathbf{x}_t \mid \mathbf{x}_{<t}) = \text{softmax}(\mathbf{z}_t) \in \mathbb{R}^{|\mathcal{V}|}$.

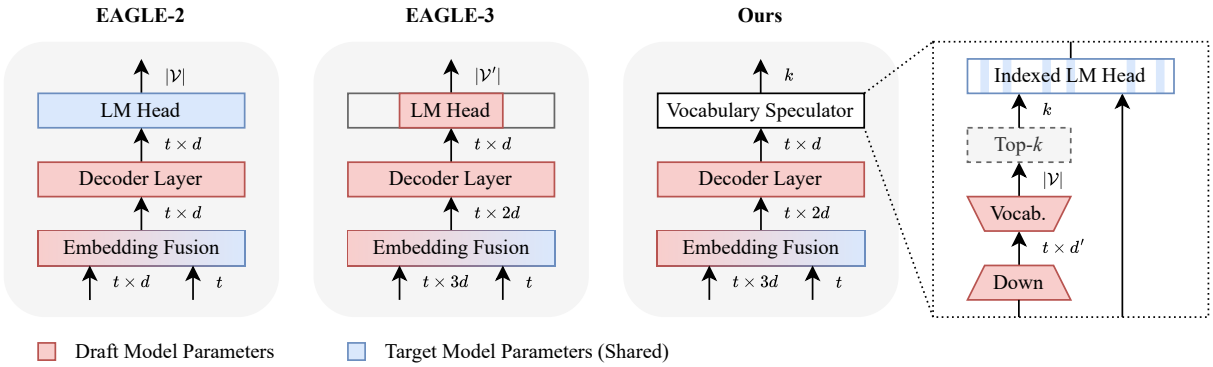



Figure 2: Outline of the draft model architectures for speculative decoding. EAGLE-2 forms predictions over the entire target model vocabulary, whereas EAGLE-3 uses a fixed subset, as in FR-Spec and VocabTrim. In contrast, SpecVocab (ours) speculates on which subset of the target model vocabulary to use at each decoding step.

### 3.2 Problem Definition

Computing the output distribution proves to be an expensive operation, as it scales with both the vocabulary size $|\mathcal{V}|$ and model dimensionality $d$ (Zhao et al., 2025). *Vocabulary speculation* seeks to reduce this cost by efficiently predicting a contextually relevant subset of the target model vocabulary. We formally define the vocabulary speculation problem as follows:

> *Given a context $c$, which subset of the vocabulary $\mathcal{K} \subset \mathcal{V}$ should be evaluated to support accurate decoding?*

The subset $\mathcal{K}$ should be compact, with $|\mathcal{K}| \ll |\mathcal{V}|$, while providing sufficient coverage of the tokens likely to be selected by the sampler for $p(\mathbf{x}_t \mid \mathbf{x}_{<t})$.

### 3.3 SpecVocab

We propose SpecVocab, a method for vocabulary speculation that incorporates an efficient vocabulary ranking module to predict the most relevant vocabulary subset (Figure 2).

**Step 1.** We start by computing an approximate ranking over the target vocabulary to inform candidate selection. We elect to use the final hidden state from the draft model $\mathbf{h}_t$ as our context. First, we obtain a reduced-dimensionality intermediate hidden state $\mathbf{h}'_t$ where $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d' \times d}$ and $d' \ll d$. This is used to efficiently compute approximate logits $\mathbf{s}_t$ over the vocabulary, where $\mathbf{W}_{\text{vocab}} \in \mathbb{R}^{|\mathcal{V}| \times d'}$:

$$\mathbf{h}'_t = \mathbf{W}_{\text{down}} \mathbf{h}_t$$
$$\mathbf{s}_t = \mathbf{W}_{\text{vocab}} \mathbf{h}'_t$$

**Step 2.** We then select the top-$k$ indices from $\mathbf{s}_t$, forming our candidate vocabulary $\mathcal{K}_t$, where $\mathbf{k}_t$ represents the indices of the corresponding tokens. For simplicity, we use a fixed value of $k$, although this value can also be varied per decoding step.

$$\mathbf{k}_t = \text{top-k}(\mathbf{s}_t, k)$$

As SpecVocab is context-aware, it can use substantially smaller subsets than prior work (typically $k = 2048$), compared with the 32K subset conventionally used in static vocabulary methods (Zhao et al., 2025; Goel et al., 2025).

**Step 3.** Finally, we compute the output logits $\mathbf{z}'_t$ for the candidate vocabulary using the original hidden states $\mathbf{h}_t$. Here, $\mathbf{U}'_t$ represents an intermediate matrix containing only the output embeddings corresponding to the candidate vocabulary.

$$\mathbf{U}'_t = \mathbf{U}[\mathbf{k}_t, :]$$
$$\mathbf{z}'_t = \mathbf{U}'_t \mathbf{h}_t$$

Therefore, the final output distribution produced by the draft model over the candidate vocabulary is:

$$q_{\mathcal{K}_t}(\mathbf{x}_t \mid \mathbf{x}_{<t}) = \text{softmax}(\mathbf{z}'_t)$$

### 3.4 Training

To enable accurate vocabulary speculation, we learn the weights for $\mathbf{W}_{\text{down}}$ and $\mathbf{W}_{\text{vocab}}$ through distillation from the target model. During training, we exclude *Steps 2 & 3*, focusing only on approximating the target model output distribution. Specifically, we form the output distribution over the entire target model vocabulary:

$$q_{\text{aux}}(\mathbf{x}_t \mid \mathbf{x}_{<t}) = \text{softmax}(\mathbf{s}_t)$$

The vocabulary speculation module is then trained alongside the draft model. We formulate the final

loss $\mathcal{L}$ as a combination of the draft model and vocabulary speculator losses, where the contribution of the latter is modulated by the weight $\lambda$. The intuition behind this approach is that we wish to encourage the draft model to learn somewhat compressed representations for the final hidden state, without compromising overall acceptance length.

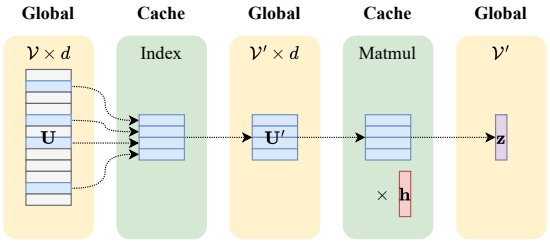
$$\mathcal{L} = \mathcal{L}_{\text{TTT}}(p, q) + \lambda \mathcal{L}_{\text{TTT}}(p, q_{\text{aux}})$$

We use $\mathcal{L}_{\text{TTT}}$ to denote the *training-time test* loss proposed in EAGLE-3 (Li et al., 2025b), which can be conceptualized as a cross-entropy loss across additional timesteps simulated during training. However, we emphasize that our method is not tied to any specific training approach. Rather, we simply mirror the same loss used for the draft model to the vocabulary speculator module.

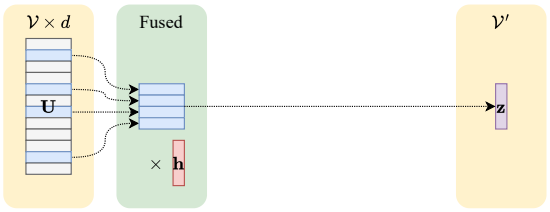
### 3.5 Inference

To demonstrate the efficacy of our approach, we integrate SpecVocab with a state-of-the-art speculative decoding method, EAGLE-3 (Li et al., 2025b). EAGLE-3 leverages tree attention to simultaneously predict multiple tokens at each decoding step of the draft model. Consequently, in practice, SpecVocab operates on batches of hidden states following the tree-like draft. Nonetheless, we emphasize that our approach is not tied to EAGLE-3 and can be applied to other speculative decoding approaches that leverage lightweight draft models.

**Custom kernel.** In *Step 3*, computing the output logits requires taking the dot product between the final hidden state $\mathbf{h}_t$ and the embedding at each index of $\mathbf{k}_t$ in $\mathbf{U}$. Naively, this can be performed by materializing the dense matrix $\mathbf{U}'_t$ followed by a matrix-vector multiplication with $\mathbf{h}_t$. However, this would require reading, writing, and re-reading $k$ vectors from memory, along with any allocation costs. To avoid this unnecessary memory traffic, we instead implement a fused kernel that only reads each embedding from memory once. We visually illustrate these memory operations and the difference between the two approaches in Figure 3.




(a) Indexed LM head operation in PyTorch.




(b) Our custom fused kernel implementation.

Figure 3: The sequence of memory access operations required by the indexed LM head operation, alternating between global and cache memory.

## 4 Experimental Setup

**Baselines.** To evaluate the performance of our approach, we compare against a variety of baselines:

- **Autoregressive Decoding**. To contextualize the improvement in throughput from speculative decoding, we report results for standard autoregressive decoding with the target model.
- **EAGLE-3** (Li et al., 2025b). We focus on the state-of-the-art EAGLE-3 method for speculative decoding. For transparency, we report results for both our own reproduction of draft model training and a third-party model.[2]
- **FR-Spec** (Zhao et al., 2025). We perform post-training vocabulary pruning with FR-Spec. Following the original work, we compute token frequencies with a 1B-token sample of the SlimPajama pre-training corpus (Soboleva et al., 2023).
- **VocabTrim** (Goel et al., 2025). Similarly, we also perform post-training vocabulary pruning with VocabTrim. Following the original work, we compute the token frequencies with target model data. Therefore, this uses the same synthetic dataset used to train all of our draft models.

We emphasize that both FR-Spec and VocabTrim build upon EAGLE-2 (Li et al., 2024a), rather than EAGLE-3. For a fair comparison, we build stronger baselines by re-implementing both methods with EAGLE-3. Namely, we apply each pruning method once to a full-vocabulary EAGLE-3 draft model. Following Zhao et al. (2025) and Li et al. (2025b), we use a draft model vocabulary size of 32K.

**Models.** To examine how each approach generalizes, we experiment with four open-source LMs from two separate model families. We employ

[2] https://huggingface.co/Tengyunw/qwen3_8b_eagle3

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| EAGLE-3 | 4.52 | 3.34 | 3.86 | 4.46 | 5.28 | 4.52 | 4.33 |
| EAGLE-3 (Our Reproduction) | 5.11 | 3.60 | 4.21 | 4.80 | 5.94 | 5.04 | 4.78 |
| EAGLE-3 + FR-Spec | 4.62 | 3.80 | **4.52** | 4.52 | 5.77 | 4.90 | 4.69 |
| EAGLE-3 + VocabTrim | 5.17 | 3.69 | 4.28 | 4.72 | 6.04 | 5.13 | 4.84 |
| EAGLE-3 + SpecVocab (Ours) | **5.30** | **3.82** | 4.49 | **4.92** | **6.28** | **5.26** | **5.01** |

Table 1: Acceptance length (tokens) for each decoding method across the Spec-Bench benchmark with Qwen3 8B. We report the average acceptance length over five seeds. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | $98.5_{0.0}$ | $92.4_{0.1}$ | $96.5_{0.0}$ | $98.9_{0.0}$ | $98.7_{0.0}$ | $97.3_{0.0}$ | $97.1_{0.0}$ (1.00×) |
| EAGLE-3 | $227.7_{1.6}$ | $149.7_{0.4}$ | $187.1_{0.9}$ | $226.3_{0.9}$ | $264.1_{1.4}$ | $221.5_{1.5}$ | $212.7_{0.8}$ (2.19×) |
| EAGLE-3 (Our Reproduction) | $258.9_{0.1}$ | $159.1_{0.2}$ | $203.9_{0.2}$ | $244.1_{0.2}$ | $297.8_{0.2}$ | $246.9_{0.2}$ | $235.1_{0.2}$ (2.42×) |
| EAGLE-3 + FR-Spec | $234.2_{0.2}$ | $166.0_{0.3}$ | $\mathbf{218.2}_{0.2}$ | $230.2_{0.2}$ | $289.3_{0.3}$ | $241.3_{0.2}$ | $229.8_{0.2}$ (2.37×) |
| EAGLE-3 + VocabTrim | $261.7_{0.2}$ | $162.0_{0.2}$ | $207.2_{0.1}$ | $239.7_{0.1}$ | $302.9_{0.2}$ | $252.1_{0.2}$ | $237.6_{0.2}$ (2.45×) |
| EAGLE-3 + SpecVocab (Ours) | $\mathbf{267.6}_{0.2}$ | $\mathbf{166.5}_{0.2}$ | $216.5_{0.1}$ | $\mathbf{249.4}_{0.2}$ | $\mathbf{313.6}_{0.2}$ | $\mathbf{257.2}_{0.2}$ | $\mathbf{245.2}_{0.2}$ (2.53×) |

Table 2: Throughput (tokens per second) for each decoding method across the Spec-Bench tasks with Qwen3 8B. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.

Qwen3 (Yang et al., 2025) in 4B and 8B sizes, and OLMo 2 (Walsh et al., 2025) in 1B and 7B sizes. We select these model families as they represent state-of-the-art performance at the time of writing, relative to their size and respective open-source categories (open-weights and fully-open).

**Training.** We closely follow the original training protocol from Li et al. (2025b). We adopt the UltraChat (Ding et al., 2023) dataset, with assistant responses generated by the target model, providing approximately 464K training examples. We report the training hyperparameters in Appendix C.

**Evaluation.** We adopt the Spec-Bench (Xia et al., 2024) dataset in its entirety for evaluation and benchmarking. This consists of a varied set of tasks, namely multi-turn conversation (MT-Bench; Zheng et al., 2023), machine translation (WMT14 DE-EN; Bojar et al., 2014), mathematical reasoning (GSM8K; Cobbe et al., 2021), summarization (CNN/Daily Mail; Hermann et al., 2015; See et al., 2017), retrieval-augmented generation, and question answering (Natural Questions; Kwiatkowski et al., 2019). We report full dataset statistics in Appendix D.

**Metrics.** Following convention (Xia et al., 2024), we focus our evaluation on two key metrics: acceptance length and throughput. *Acceptance length* is the average number of tokens proposed by the draft model that are successfully verified by the target model, reflecting the draft model accuracy. However, an increased acceptance length will not translate to faster decoding if the draft model execution is too slow (Zhao et al., 2025). Consequently, we also measure system *throughput*, the rate at which output tokens are generated.

**Implementation details.** To ensure that our experimental results reflect real-world performance, we implement all experiments using the highly-optimized SGLang inference framework (Zheng et al., 2024). Notably, SGLang leverages CUDA graphs to virtually eliminate the CPU overhead when launching GPU operations. We highlight this point as Zhao et al. (2025) find that the performance loss incurred by large vocabularies can be obscured by CPU overhead in naive implementations.

**Computational resources.** We perform all model training using four NVIDIA H100 80GB GPUs, while we perform all inference experiments using only a single GPU. For consistency, we ensure that all evaluations for a given model configuration are performed on the same underlying physical hardware.

## 5 Results & Discussion

**SpecVocab consistently outperforms EAGLE-3.** Table 1 presents the acceptance length across the Spec-Bench tasks for Qwen3 8B. Across all tasks, we observe that SpecVocab demonstrates a higher

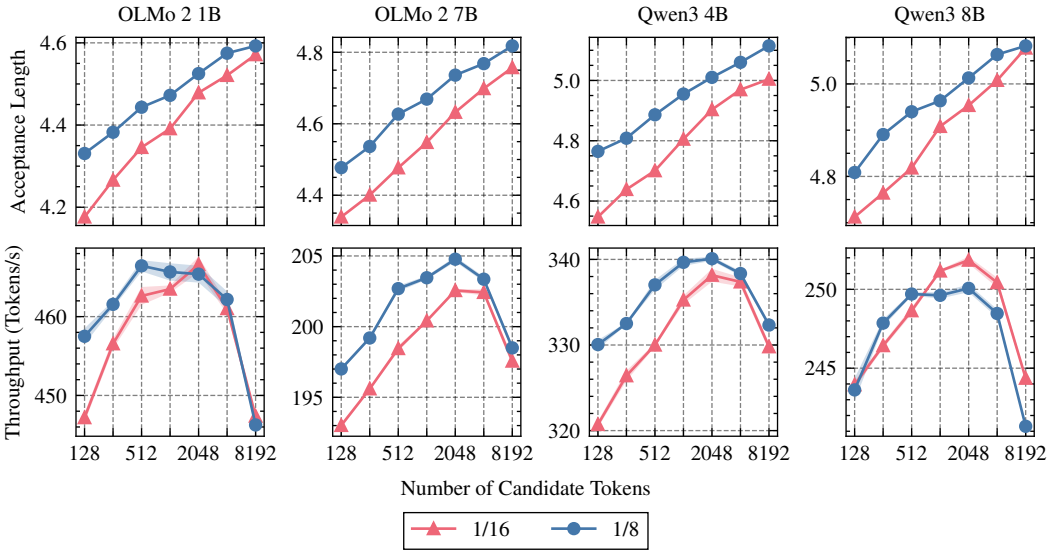



Figure 4: The acceptance length and throughput when varying both the number of candidate tokens ($k$) and intermediate dimensionality of our method relative to the target model ($d'/d$). We present the results for every model across five seeds, with the standard deviation denoted by the shaded area.

acceptance length than EAGLE-3. This ranges from a 2.5% improvement for question answering (+0.12 tokens per drafting step) to a 6.8% improvement for summarization (+0.29 tokens). Considering the 4B size of Qwen3 (Table 10, Appendix A), we observe that acceptance length ranges from a 3.1% improvement for question answering to 10.3% for retrieval-augmented generation. On average, SpecVocab achieves 4.8% higher acceptance length (+0.23 tokens per drafting step) for Qwen3 8B, while Qwen3 4B achieves a slightly higher 6.0% improvement (+0.28 tokens).

**Crucially, improvements in acceptance length translate to greater throughput for SpecVocab.** Considering all four models, we observe that there are substantial improvements in average throughput compared to EAGLE-3. Specifically, we note a 4.3% improvement for Qwen3 8B (Table 2), and a 5.0% increase for Qwen3 4B (Table 13, Appendix A). For OLMo 2 1B, we observe a similar improvement of 5.0%, while OLMo 2 7B achieves an even greater increase of 8.1% (Tables 11 and 12, Appendix A). We believe that these substantial increases in throughput, measured in a real-world setting, underscore the efficacy of our method.

**Post-training vocabulary pruning outperforms the reduced-vocabulary training of EAGLE-3.** We examine the difference in acceptance length between EAGLE-3 and VocabTrim, which both rely on the token frequency statistics from the target-generated training data. However, EAGLE-3 performs vocabulary pruning before training, while VocabTrim is applied afterwards. We observe that VocabTrim outperforms EAGLE-3 in all but one case, with the exception being the question answering task for Qwen3 8B (Table 1). Considering all models, we observe that VocabTrim achieves an average increase in acceptance length ranging from 1.1% for Qwen3 8B (Table 1) to 2.7% for OLMo 2 7B (Table 9, Appendix A). *We highlight the significance of this finding, which suggests that the reduced-vocabulary training regime from EAGLE-3 can unnecessarily harm draft model performance.*

**For a static vocabulary, target model token frequencies perform better than an external corpus.** Since FR-Spec (Zhao et al., 2025) and VocabTrim (Goel et al., 2025) are concurrent studies, neither offers a direct comparison. We therefore examine the performance difference between token frequencies computed from large-scale corpora versus data generated by the target model, respectively. Considering the resulting throughput across all models, we observe that there are clear trends across tasks (Tables 2, 11, 12, and 13). Specifically, VocabTrim performs better for multi-turn conversation, question answering, mathematical reasoning, and retrieval-augmented generation, whereas FR-Spec performs better for machine translation and summarization. As VocabTrim performs better in the majority of tasks (i.e. four of six), this corroborates the finding from Goel et al. (2025) that using target-generated token frequencies benefits performance.

**SpecVocab generally outperforms all methods.** We first consider the throughput for each of the Spec-Bench tasks individually, across all models. We observe that SpecVocab outperforms all other methods in all but one case, i.e. 23 of 24 instances. The only exception is the summarization task for Qwen3 8B (Table 2), where SpecVocab is narrowly slower than FR-Spec by 0.8% (1.7 fewer tokens per second). Considering the average throughput across tasks, we observe that SpecVocab consistently outperforms all other methods across every model. Next, we consider the performance of our approach over VocabTrim, the best-performing static vocabulary method. We observe that the improvements in average throughput for SpecVocab range from 3.1% for Qwen3 4B (Table 13, Appendix A) to 5.4% for OLMo 2 7B (Table 12, Appendix A). For three of the four models, this translates to at least an additional ten tokens per second.

**Performance improvements vary between tasks.** Across all models, we observe that the improvements in throughput differ between tasks. For example, we first consider the range of improvements in throughput demonstrated by our method versus EAGLE-3, across all models. The greatest improvement is 11.0% on mathematical reasoning with OLMo 2 7B (Table 12, Appendix A). In contrast, the lowest improvement is 2.2% on question answering with Qwen3 8B (Table 2). This pattern is not consistent across models. For example, the greatest improvement in throughput for OLMo 2 1B is 8.0% on the summarization task (Table 11, Appendix A), while Qwen3 4B sees a maximum improvement of 9.5% for the retrieval-augmented generation task (Table 13, Appendix A).

**SpecVocab performance scales with model size.** To examine how model size impacts the performance of SpecVocab, we compare the throughput speedup between models of different sizes from the same model family. Following the trend set by EAGLE-3, we observe that our method can achieve a greater speedup for larger models. For example, SpecVocab achieves a 2.18× speedup over autoregressive decoding for Qwen3 4B (Table 13, Appendix A), yet achieves a larger speedup of 2.53× for Qwen3 8B (Table 2). In the case of OLMo 2 1B, we observe a 1.38× speedup for SpecVocab over autoregressive decoding (Table 11, Appendix A), whereas we see a much larger speedup of 2.20× for OLMo 2 7B (Table 12, Appendix A).

| Dimensionality ($d'/d$) | $\lambda$ | Acceptance Length |
|---|---|---|
| 1/16 | 0.01 | 4.57 |
| | 0.1 | **4.61** |
| | 0.2 | **4.61** |
| | 0.5 | 4.60 |
| 1/8 | 0.01 | 4.65 |
| | 0.1 | **4.74** |
| | 0.2 | 4.71 |
| | 0.5 | 4.71 |

Table 3: The impact of the training loss weighting ($\lambda$) for SpecVocab upon acceptance length. We list results for both intermediate dimensionalities with Qwen3 8B.

## 6 Analysis

**SpecVocab requires no more than 2% of the exact logits to be computed at each decoding step.** Figure 4 presents the throughput when varying the number of candidate tokens ($k$). Across all models, we observe that computing the exact logits for only 2048 candidate tokens is sufficient for maximum throughput. This represents 1.4% and 2.0% of the target model vocabulary for Qwen3 and OLMo 2, respectively. In comparison to the static vocabulary methods (EAGLE-3, FR-Spec, and VocabTrim), this requires over 15× fewer tokens to be evaluated. Interestingly, for the smallest model (OLMo 2 1B), maximum throughput can be achieved by using only 512 tokens. This is equivalent to 0.5% of the model vocabulary, over 60× fewer than the 32K subset used by the static vocabulary methods.

**The intermediate dimensionality for SpecVocab can be as small as just 1/16 of the target model.** Figure 4 also shows the acceptance length when varying the intermediate embedding dimensionality ($d'$). We observe that increasing the intermediate dimensionality consistently leads to an increase in acceptance length across all models and quantities of candidate tokens. However, this also increases the computational complexity of forming the candidate vocabulary. Therefore, increasing the dimensionality may not lead to an increase in throughput. As an example, we consider two models with identical intermediate dimensionality, OLMo 2 7B and Qwen3 8B. The difference in acceptance length between an intermediate dimensionality of 1/16 and 1/8 for OLMo 2 7B is 2.2%, assuming 2048 candidate tokens. For Qwen3 8B, the difference is nearly halved, at 1.2%. Consequently, OLMo 2 7B achieves greater throughput at 1/8, while Qwen3 can achieve maximum throughput at 1/16.

| Approach | Complexity |
|---|---|
| Full Vocabulary (e.g. EAGLE-2) | $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d)$ |
| Reduced Vocabulary (e.g. FR-Spec) | $\mathcal{O}(\lvert\mathcal{V}'\rvert \cdot d)$ |
| SpecVocab (Ours) | $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d' + k \cdot d)$ |

Table 4: Computational complexity of calculating the output logits for each approach.

**SpecVocab benefits from the joint training loss.** Table 3 presents the impact of the loss weight hyperparameter ($\lambda$) upon the draft model acceptance length during early experimentation. We observe that increasing the contribution of the loss from the vocabulary speculator module leads to a greater acceptance length. For example, raising $\lambda$ from 0.01 to 0.1 leads to a 0.09 increase in acceptance length for Qwen3 8B when using 1/8 of the target model dimensionality. We hypothesize that allowing the vocabulary speculator module loss to affect the entire draft model has a regularizing effect, encouraging the draft model to learn more compressible representations. In turn, this may improve vocabulary speculation at reduced dimensionalities.

**SpecVocab is relatively robust to the loss weight.** We trial values of $\lambda \in \{0.01, 0.1, 0.2, 0.5\}$ for two different intermediate embedding dimensionalities. We observe that training with $\lambda = 0.1$ achieves the highest acceptance length for both dimensionalities. For 1/16 dimensionality, an increased $\lambda$ of 0.2 also achieves the highest acceptance length. In theory, using a high loss weight should harm performance, as it is a lower fidelity approximation of the true loss. Interestingly, we observe that even using a loss weight of 0.5 (i.e. contributing a third of the overall loss) is less harmful than using a very low loss weight of 0.01. For example, compared to the best-performing loss weight (0.1) for 1/8 dimensionality, $\lambda = 0.5$ leads to a 0.03 decrease in acceptance length, while $\lambda = 0.01$ leads to a $3\times$ greater decrease of 0.09.

**Theoretical analysis.** Table 4 presents the computational complexity for each type of approach, specifically full vocabulary (EAGLE-2), reduced vocabulary (EAGLE-3, FR-Spec, VocabTrim), and our own method. This formulation highlights the two-phase nature of our method, consisting of a reduced-dimensionality approximation ($\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d')$) and an exact computation over a likely subset ($\mathcal{O}(k \cdot d)$). SpecVocab is asymptotically more efficient than the other approaches when $\lvert\mathcal{V}\rvert \cdot \frac{d'}{d} + k < \lvert\mathcal{V}'\rvert$. In other words, our approach necessitates both (1) informative contextual representations when $d' \ll d$, and (2) sufficiently high recall that enables a small candidate set size $k$. In contrast to reduced-vocabulary methods, which require a large $\lvert\mathcal{V}'\rvert$ to maintain performance, SpecVocab can achieve coverage of the full vocabulary at a lower cost.

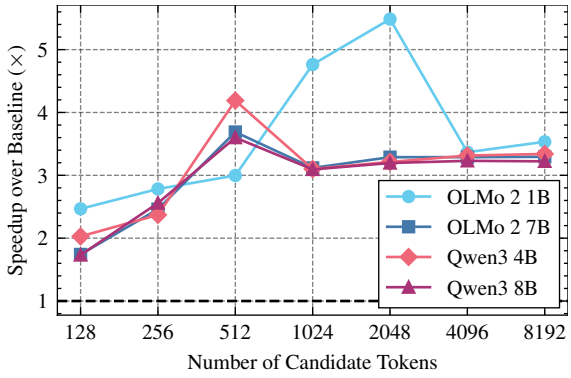



Figure 5: Microbenchmark results for our custom fused kernel versus a PyTorch baseline, for each model.

**Our kernel achieves a 3× to 5× speedup in comparison to PyTorch.** Figure 5 presents the speedup of our kernel relative to a PyTorch baseline. To create a realistic microbenchmark, we construct input activation and weight matrices corresponding to the dimensions of each model. To eliminate the impact of CPU overhead from launching GPU operations, we perform all benchmarking using CUDA graphs. We observe that our kernel substantially outperforms the PyTorch baseline across all models and quantities of candidate tokens. For example, considering the optimal number of candidate tokens of $k = 2048$, we observe that all models see a performance increase of at least $3.2\times$. OLMo 2 1B appears to be an outlier, achieving a speedup of over $5\times$, seemingly due to its smaller size.

## 7 Conclusion

In this paper, we argued that speculative decoding should additionally speculate on the output vocabulary for maximum performance. To this end, we proposed SpecVocab as a concrete implementation of a dynamic vocabulary strategy. We empirically validated our method across a variety of tasks and models, demonstrating substantial gains in acceptance length over strong speculative decoding baselines. Crucially, these gains translated to substantial improvements in real-world throughput within a widely-used inference framework. We hope that this study will inspire further work that considers the role of vocabulary within speculative decoding.

## Limitations

**Linguistic diversity.** Our evaluation relies on Spec-Bench (Xia et al., 2024), which primarily targets English-language tasks. While one of the six Spec-Bench tasks involves a non-English language (German), the benchmark remains limited to West Germanic languages. As future work, we are interested in investigating how both static and dynamic vocabulary speculative decoding approaches generalize beyond the Indo-European family, particularly to languages with substantially different morphologies.

## Acknowledgments

We would like to thank the SGLang team for their optimized implementation of speculative decoding (Zheng et al., 2024) and the SpecForge (Li et al., 2025a) draft model training framework. We are also grateful to the anonymous reviewers for their invaluable feedback. MW is supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation grant EP/S023062/1.

## References

Amine Abdaoui, Camille Pradel, and Grégoire Sigel. 2020. Load what you need: Smaller versions of mutililingual BERT. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 119–123, Online. Association for Computational Linguistics.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, and 106 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. In *First Conference on Language Modeling*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 5209–5235. PMLR.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *Preprint*, arXiv:2302.01318.

Yunfei Cheng, Aonan Zhang, Xuanyu Zhang, Chong Wang, and Yi Wang. 2024. Recurrent drafter for fast speculative decoding in large language models. *Preprint*, arXiv:2403.09919.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

Raghavv Goel, Sudhanshu Agrawal, Mukul Gagrani, Junyoung Park, Yifan Zao, He Zhang, Tian Liu, Yiping Yang, Xin Yuan, Jiuyuan Lu, Christopher Lott, and Mingu Lee. 2025. VocabTrim: Vocabulary pruning for efficient speculative decoding in LLMs. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Milan Gritta, Huiyin Xue, and Gerasimos Lampouras. 2025. DReSD: Dense retrieval for speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19822–19832, Vienna, Austria. Association for Computational Linguistics.

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.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and Zhou Xun. 2025. Over-tokenized transformer: Vocabulary is generally worth scaling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 26261–26282. PMLR.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, and 196 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Shenggui Li, Yikai Zhu, Chao Wang, Fan Yin, Shuai Shi, Yubo Wang, Yi Zhang, Yingyi Huang, Haoshuai Zheng, and Yineng Zhang. 2025a. SpecForge: Train speculative decoding models effortlessly.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28935–28948. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025b. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Michele Marzollo, Jiawei Zhuang, Niklas Roemer, Niklas Zwingenberger, Lorenz K. Müller, and Lukas Cavigelli. 2024. SSSD: Simply-scalable speculative decoding. *Preprint*, arXiv:2411.05894.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949, New York, NY, USA. Association for Computing Machinery.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Benjamin Frederick Spector and Christopher Re. 2023. Accelerating LLM inference with staged speculative decoding. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

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

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1:*

*Long Papers)*, pages 5937–5947, Online. Association for Computational Linguistics.

Sho Takase, Ryokan Ri, Shun Kiyono, and Takuya Kato. 2025. Large vocabulary size improves large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1015–1026, Vienna, Austria. Association for Computational Linguistics.

Bangsheng Tang, Carl Chengyan Fu, Fei Kou, Grigory Sizov, Haoci Zhang, Jason Park, Jiawen Liu, Jie You, Qirui Yang, Sachin Mehta, Shengyong Cai, Xiaodong Wang, Xingyu Liu, Yunlu Li, Yanjun Zhou, Wei Wei, Zhiwei Zhao, Zixi Qi, Adolfo Victoria, and 19 others. 2025. Efficient speculative decoding for llama at scale: Challenges and solutions. *Preprint*, arXiv:2508.08192.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *Advances in Neural Information Processing Systems*, volume 37, pages 114147–114179. Curran Associates, Inc.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, and 23 others. 2025. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*.

Davis Wertheimer, Joshua Rosenkranz, Thomas Parnell, Sahil Suneja, Pavithra Ranganathan, Raghu Ganti, and Mudhakar Srivatsa. 2024. Accelerating production LLMs with combined token/embedding speculators. *Preprint*, arXiv:2404.19124.

Miles Williams and Nikolaos Aletras. 2025. Vocabulary-level memory efficiency for language model fine-tuning. In *Proceedings of the 10th Workshop on Representation Learning for NLP (RepL4NLP-2025)*, pages 185–196, Albuquerque, NM. Association for Computational Linguistics.

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, Singapore. Association for Computational Linguistics.

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. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand. Association for Computational Linguistics.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025. Learning harmonized representations for speculative sampling. In *The Thirteenth International Conference on Learning Representations*.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jie Zhou, Hao Zhou, Jianyong Wang, Zhiyuan Liu, and Maosong Sun. 2025. FR-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3909–3921, Vienna, Austria. Association for Computational Linguistics.

Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu. 2024. Lookahead: An inference acceleration framework for large language model with lossless generation accuracy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6344–6355, New York, NY, USA. Association for Computing Machinery.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pages 62557–62583. Curran Associates, Inc.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Cambridge, MA.

## A Complete Results

In addition to the results presented for Qwen3 8B in Section 5 (Tables 1 and 2), we report complete results for the remaining models. Tables 8, 9, and 10 present the acceptance length with each decoding method across the Spec-Bench benchmark for OLMo 2 1B, OLMo 2 7B, and Qwen3 4B, respectively. Tables 11, 12, and 13 report the corresponding throughput values. We provide a discussion of these results in Section 5.

## B Additional Experiments

### B.1 EAGLE-3 with Full Vocabulary

To establish an upper bound on acceptance length, we construct an additional EAGLE-3 baseline that adopts the complete vocabulary of the target model. Specifically, we follow the implementation from EAGLE-2 (Li et al., 2024a) in sharing the target model LM head with the draft model (Figure 2). Tables 14 and 15 report the acceptance length and throughput for this additional baseline with Qwen3 8B. We observe that EAGLE-3 reaches 91.0% of the full vocabulary acceptance length, while SpecVocab achieves 95.3%. Despite the gains in acceptance length from a full vocabulary, the computational overhead leads to lower throughput, corroborating the findings from Zhao et al. (2025).

### B.2 Kernel Ablation

To evaluate the impact of our kernel on throughput, we implement a naive indexed LM head operation in PyTorch (Figure 3). Table 16 presents the throughput for Qwen3 8B both with and without our kernel. We observe that using this approach, instead of our kernel, leads to a 2.7% *decrease* in average throughput compared to EAGLE-3.

## C Hyperparameters

We follow the hyperparameters used by EAGLE-3 (Li et al., 2025b) as closely as possible, deriving values from the paper and software implementation. We report all training hyperparameters in Table 5 and all inference hyperparameters in Table 6.

## D Datasets

We report dataset statistics for our evaluation dataset, Spec-Bench (Xia et al., 2024), in Table 7. Specifically, we report the number of examples both per task and in aggregate.

| Hyperparameter | Value |
|---|---|
| Batch Size | 8 |
| Training Steps | 800,000 |
| Warmup Steps | 1.5% |
| Learning Rate | $5\times10^{-5}$ |
| Adam $\beta_1$ | 0.9 |
| Adam $\beta_2$ | 0.95 |
| Adam $\epsilon$ | $1\times10^{-8}$ |
| Weight Decay | 0 |
| TTT Length | 7 |

Table 5: Hyperparameters used for EAGLE-3 draft model training.

| Hyperparameter | Value |
|---|---|
| Decoding Steps | 8 |
| Top-$k$ | 10 |
| Draft Tokens | 60 |

Table 6: Hyperparameters used for inference with EAGLE-3 draft models.

| Category | Examples |
|---|---|
| Multi-turn Conversation | 80 |
| Machine Translation | 80 |
| Summarization | 80 |
| Question Answering | 80 |
| Mathematical Reasoning | 80 |
| Retrieval-augmented Generation | 80 |
| Total | 480 |

Table 7: The number of examples in each category of Spec-Bench (Xia et al., 2024).

## E Additional Related Work

Our paper focuses on vocabulary use in neural language models with finite subword vocabularies (Abdaoui et al., 2020; Williams and Aletras, 2025). Many speculative decoding approaches rely upon neural language models, typically either using an external draft model (Xia et al., 2023; Leviathan et al., 2023; Chen et al., 2023; Li et al., 2024b) or adapting the target model (Cai et al., 2024; Zhang et al., 2024; Ankner et al., 2024). However, some speculative decoding techniques avoid using neural language models for drafting (He et al., 2024; Zhao et al., 2024; Marzollo et al., 2024), potentially mitigating the need for vocabulary speculation.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| EAGLE-3 (Our Reproduction) | 4.80 | 3.22 | 4.55 | 4.00 | 4.88 | 4.39 | 4.31 |
| EAGLE-3 + FR-Spec | 4.56 | 3.41 | 4.77 | 4.04 | 4.87 | 4.32 | 4.33 |
| EAGLE-3 + VocabTrim | 4.83 | 3.24 | 4.64 | 4.12 | 5.08 | 4.39 | 4.38 |
| EAGLE-3 + SpecVocab (Ours) | **4.98** | **3.43** | **4.92** | **4.13** | **5.14** | **4.55** | **4.53** |

Table 8: Acceptance length (tokens) for each decoding method across the Spec-Bench benchmark with OLMo 2 1B. We report the average acceptance length over five seeds. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| EAGLE-3 (Our Reproduction) | 4.76 | 3.58 | 4.39 | 4.19 | 5.09 | 4.44 | 4.41 |
| EAGLE-3 + FR-Spec | 4.72 | **3.87** | 4.54 | 4.19 | 4.93 | 4.52 | 4.46 |
| EAGLE-3 + VocabTrim | 5.03 | 3.74 | 4.44 | 4.24 | 5.21 | 4.51 | 4.53 |
| EAGLE-3 + SpecVocab (Ours) | **5.12** | 3.86 | **4.75** | **4.34** | **5.60** | **4.75** | **4.74** |

Table 9: Acceptance length (tokens) for each decoding method across the Spec-Bench benchmark with OLMo 2 7B. We report the average acceptance length over five seeds. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| EAGLE-3 (Our Reproduction) | 5.02 | 3.37 | 4.31 | 4.94 | 5.79 | 4.94 | 4.73 |
| EAGLE-3 + FR-Spec | 4.57 | 3.52 | 4.59 | 4.66 | 5.61 | 4.79 | 4.62 |
| EAGLE-3 + VocabTrim | 5.20 | 3.43 | 4.35 | 5.05 | 5.86 | 5.04 | 4.82 |
| EAGLE-3 + SpecVocab (Ours) | **5.26** | **3.59** | **4.63** | **5.09** | **6.05** | **5.45** | **5.01** |

Table 10: Acceptance length (tokens) for each decoding method across the Spec-Bench benchmark with Qwen3 4B. We report the average acceptance length over five seeds. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 338.3 $_{3.0}$ | 309.0 $_{0.2}$ | 330.3 $_{1.1}$ | 340.6 $_{0.1}$ | 339.3 $_{0.4}$ | 332.2 $_{0.4}$ | 331.6 $_{0.7}$ (1.00×) |
| EAGLE-3 (Our Reproduction) | 499.5 $_{1.3}$ | 301.5 $_{0.7}$ | 457.7 $_{1.0}$ | 416.0 $_{0.8}$ | 501.2 $_{1.1}$ | 445.7 $_{1.1}$ | 436.9 $_{1.0}$ (1.32×) |
| EAGLE-3 + FR-Spec | 474.0 $_{0.5}$ | 315.4 $_{0.7}$ | 477.2 $_{0.8}$ | 419.1 $_{0.8}$ | 500.8 $_{0.5}$ | 438.2 $_{0.4}$ | 437.4 $_{0.5}$ (1.32×) |
| EAGLE-3 + VocabTrim | 502.0 $_{0.9}$ | 302.2 $_{0.7}$ | 463.7 $_{1.8}$ | 427.6 $_{0.9}$ | 519.7 $_{0.9}$ | 444.7 $_{1.1}$ | 443.3 $_{0.8}$ (1.34×) |
| EAGLE-3 + SpecVocab (Ours) | **519.7** $_{0.6}$ | **319.0** $_{0.5}$ | **494.0** $_{1.1}$ | **430.7** $_{0.7}$ | **528.2** $_{0.9}$ | **461.3** $_{2.1}$ | **458.8** $_{0.8}$ (1.38×) |

Table 11: Throughput (tokens per second) for each decoding method across the Spec-Bench tasks with OLMo 2 1B. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 94.3 $_{0.0}$ | 88.5 $_{1.2}$ | 92.1 $_{0.2}$ | 94.9 $_{0.4}$ | 95.1 $_{0.1}$ | 92.6 $_{0.0}$ | 92.9 $_{0.2}$ (1.00×) |
| EAGLE-3 (Our Reproduction) | 209.7 $_{0.1}$ | 140.4 $_{0.2}$ | 185.9 $_{0.1}$ | 185.6 $_{0.1}$ | 224.1 $_{0.2}$ | 190.3 $_{0.0}$ | 189.3 $_{0.1}$ (2.04×) |
| EAGLE-3 + FR-Spec | 207.9 $_{0.1}$ | 150.1 $_{0.1}$ | 191.9 $_{0.4}$ | 185.0 $_{0.9}$ | 217.1 $_{0.3}$ | 193.2 $_{0.2}$ | 190.9 $_{0.1}$ (2.05×) |
| EAGLE-3 + VocabTrim | 221.1 $_{0.1}$ | 146.0 $_{0.3}$ | 188.3 $_{0.2}$ | 187.5 $_{0.3}$ | 229.3 $_{0.2}$ | 193.2 $_{0.1}$ | 194.2 $_{0.2}$ (2.09×) |
| EAGLE-3 + SpecVocab (Ours) | **227.4** $_{0.1}$ | **151.0** $_{0.2}$ | **202.4** $_{0.0}$ | **194.0** $_{0.1}$ | **248.7** $_{0.1}$ | **204.7** $_{0.1}$ | **204.7** $_{0.0}$ (2.20×) |

Table 12: Throughput (tokens per second) for each decoding method across the Spec-Bench tasks with OLMo 2 7B. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| Autoregressive Decoding | 160.5 $_{0.0}$ | 143.4 $_{0.1}$ | 157.5 $_{0.0}$ | 161.4 $_{0.0}$ | 160.6 $_{0.0}$ | 159.1 $_{0.0}$ | 157.1 $_{0.0}$ (1.00×) |
| EAGLE-3 (Our Reproduction) | 357.0 $_{0.4}$ | 200.2 $_{0.4}$ | 295.9 $_{0.2}$ | 353.9 $_{0.4}$ | 405.8 $_{0.4}$ | 344.7 $_{0.8}$ | 326.2 $_{0.4}$ (2.08×) |
| EAGLE-3 + FR-Spec | 326.1 $_{0.6}$ | 207.5 $_{0.5}$ | 314.3 $_{0.6}$ | 334.1 $_{0.6}$ | 393.9 $_{1.2}$ | 335.3 $_{0.6}$ | 318.5 $_{0.7}$ (2.03×) |
| EAGLE-3 + VocabTrim | 369.4 $_{0.7}$ | 202.2 $_{1.2}$ | 298.5 $_{0.6}$ | 362.0 $_{0.8}$ | 410.0 $_{1.1}$ | 351.6 $_{0.9}$ | 332.3 $_{0.7}$ (2.12×) |
| EAGLE-3 + SpecVocab (Ours) | **371.5** $_{0.9}$ | **207.9** $_{3.4}$ | **314.8** $_{0.7}$ | **362.3** $_{0.9}$ | **421.2** $_{1.2}$ | **377.4** $_{1.1}$ | **342.5** $_{0.9}$ (2.18×) |

Table 13: Throughput (tokens per second) for each decoding method across the Spec-Bench tasks with Qwen3 4B. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| EAGLE-3 (Our Reproduction) | 5.11 | 3.60 | 4.21 | 4.80 | 5.94 | 5.04 | 4.78 |
| EAGLE-3 (Full Vocabulary) | **5.45** | **4.01** | **4.79** | **5.11** | **6.41** | **5.79** | **5.26** |
| EAGLE-3 + SpecVocab (Ours) | 5.30 | 3.82 | 4.49 | 4.92 | 6.28 | 5.26 | 5.01 |

Table 14: Acceptance length (tokens) across the Spec-Bench benchmark with Qwen3 8B, including a full-vocabulary draft model. We report the average acceptance length over five seeds. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| EAGLE-3 (Our Reproduction) | $258.9_{0.1}$ | $159.1_{0.2}$ | $203.9_{0.2}$ | $244.1_{0.2}$ | $297.8_{0.2}$ | $246.9_{0.2}$ | $235.1_{0.2}$ |
| EAGLE-3 (Full Vocabulary) | $219.9_{0.2}$ | $143.0_{0.3}$ | $186.0_{0.2}$ | $206.8_{0.2}$ | $256.4_{0.3}$ | $227.5_{0.2}$ | $206.6_{0.2}$ |
| EAGLE-3 + SpecVocab (Ours) | $\mathbf{267.6}_{0.2}$ | $\mathbf{166.5}_{0.2}$ | $\mathbf{216.5}_{0.1}$ | $\mathbf{249.4}_{0.2}$ | $\mathbf{313.6}_{0.2}$ | $\mathbf{257.2}_{0.2}$ | $\mathbf{245.2}_{0.2}$ |

Table 15: Throughput (tokens per second) across the Spec-Bench tasks with Qwen3 8B, including a full-vocabulary draft model. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.

| Method | Conv. | MT | Summ. | QA | Math | RAG | Mean |
|---|---|---|---|---|---|---|---|
| EAGLE-3 (Our Reproduction) | $258.4_{0.3}$ | $158.5_{0.5}$ | $203.7_{0.3}$ | $243.6_{0.3}$ | $297.2_{0.4}$ | $246.7_{0.3}$ | $234.7_{0.4}$ |
| EAGLE-3 + SpecVocab (Without Kernel) | $248.0_{0.2}$ | $154.5_{0.4}$ | $202.0_{0.3}$ | $232.1_{0.2}$ | $287.9_{1.0}$ | $245.2_{0.2}$ | $228.3_{0.3}$ |
| EAGLE-3 + SpecVocab | $\mathbf{266.9}_{0.2}$ | $\mathbf{165.2}_{0.2}$ | $\mathbf{215.8}_{0.2}$ | $\mathbf{248.6}_{0.1}$ | $\mathbf{312.1}_{0.8}$ | $\mathbf{256.6}_{0.1}$ | $\mathbf{244.2}_{0.2}$ |

Table 16: Throughput (tokens per second) across the Spec-Bench tasks with Qwen3 8B, including an ablation of our kernel. We report the average throughput over five seeds, with the standard deviation as subscripts. The best result in each category is highlighted in **bold**.