VOCABTRIM: Vocabulary Pruning for Efficient Speculative Decoding in LLMs

Raghavv Goel [1] **Sudhanshu Agrawal** [1] **Mukul Gagrani** [1] **Junyoung Park** [1] **Yifan Zao** [1] **He Zhang** [1] **Tian Liu** [1]
Yiping Yang [1] **Xin Yuan** [1] **Jiuyan Lu** [1] **Chris Lott** [1] **Mingu Lee** [1]

Abstract

In this paper, we introduce a simple training-free technique to improve the performance of drafter-based speculative decoding (SpD) methods that incorporates language modeling head (LM head) during drafting process. A drafter-based speculative decoding leverages one or more smaller language models, a.k.a. *drafters* or *draft models*, to sample a draft sequence or tree consisting of multiple tokens, followed by verification by a base LLM, a *target model*, accepting a subset as its valid generation. As it is usually considered that the speculative decoding requires one-to-one mapping between vocabularies of the target model and the draft model, it has been natural to share the vocabulary between them, or even share the LM head as in EAGLE or Medusa. We first identify that this draft token sampling scheme inherently contains an unnecessary inference overhead in drafting, especially for some target LLMs with very large vocabularies. Then, we propose a simple technique, VOCABTRIM, to mitigate the drafting overhead to improve the generation speed in memory-bound environment. VOCABTRIM reconstructs the drafter LM head to contain only a limited set of tokens, selected by the most frequently sampled from the vocabulary of the target model. While limiting the vocabulary in drafting slightly degrades the acceptance rate, it significantly reduces the drafting latency in memory-bound process which is often the case on edge devices, resulting in higher memory-bound speed up (MBSU). We show that our method can boost the memory-bound speed-up for Llama-3 models on Spec-Bench, specifically by 16% for Llama-3.2-3B-Instruct.

[1]Qualcomm AI Research. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.. Correspondence to: Raghavv Goel, Mingu Lee <raghgoel/mingul@qti.qualcomm.com>.

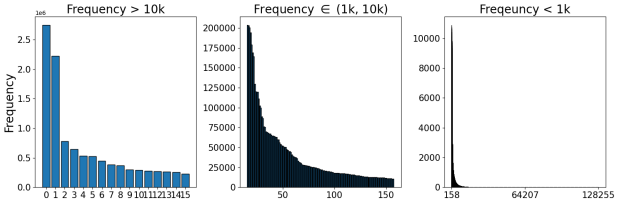

Figure 1: xLAM (function-calling) dataset token freqeuncy based on target model generation. Plots divided into three parts for ease of readability.

1. Introduction

Speculative Decoding (SpD) (Leviathan et al., 2023) is a widely adopted inference optimization technique for large language models (LLMs). In SpD, a lightweight drafter speculates the tokens that a larger target model would generate. The target model then selectively accepts or rejects these speculations based on a policy that often ensures that the output follows the same distribution as the target model. Prior work on SpD has focused primarily on balancing the expressiveness and efficiency of the drafter, either by designing novel model architectures (Li et al., 2024a; Cai et al., 2024; Zimmer et al., 2024) or by developing draft-time search algorithms (Miao et al., 2023; Jeon et al., 2024). However, these approaches typically require training a separate drafter, with the shared tokenizer, when a well-aligned pretrained model is not available (Goel et al., 2024).

While recent advances have proposed increasingly effective drafters, we observe that their predictions tend to focus on "easy-to-predict" tokens such as articles, prepositions, or completions of partially generated words. For example, a drafter may suggest `table` immediately after the target model generates `vege`, thereby completing the word `vegetable` (Gagrani et al., 2024). This behavior suggests that the role of the drafter could be shifted from a general-purpose generator to a more specialized token suggester, particularly by restricting its vocabulary.

We further observe that in many downstream tasks, the target model generation is limited on a small portion of its full vocabulary. As shown in Figure 1, we analyze the output token distribution of Llama-3.2-3B-Instruct on the training split of a function calling dataset (Liu et al., 2024). Only

a small set of tokens occur at very high frequencies. For instance, 15 tokens are sampled more than 10K times and the next 140 tokens appear between 1K and 10K times. In contrast, more than 120,000 tokens are rarely or not sampled at all.

This observation suggests an opportunity to simplify the output token space of the drafter. If the drafter only needs to predict a limited set of frequently sampled tokens, computing logits over the full vocabulary may be unnecessary, i.e., saving memory and computation for drafting. The saving becomes more significant when the target model is with a large vocabulary size, which is common in modern LLMs for improved support for multiple language and token efficiency, i.e., compression rate. (Dubey et al., 2024) The language modeling (LM) head, which maps hidden representations to vocabulary logits through a linear projection, is often a major contributor to both model size and inference latency. This issue is particularly pronounced in small drafters with large vocabularies. For example, in a 314M-parameter drafter using the Llama 3 vocabulary (128K tokens), the LM head alone accounts for over 30% of the total parameters. This significantly limits the speedup potential of SpD, given that generation is typically a memory-bound process.

Motivated by these observations, we propose VOCABTRIM, a training-free method for improving the efficiency of speculative decoding by reducing the size of the LM head of the drafter. By restricting the output dimension of the LM head, VOCABTRIM provides significant memory and latency savings without requiring any training or architectural changes. For Llama 3 models, we show that VOCABTRIM can reduce the output dimension of the LM heads by up to 75% with negligible impact to the acceptance rates. When applied to the state-of-the-art SpD method EAGLE-2, our method achieves on average 16% latency improvement on Spec-Bench tasks (Xia et al., 2024). We hope that this work encourages further research into optimizing previously underexplored components of the SpD pipeline.

2. Related Work

Speculative decoding was introduced in (Leviathan et al., 2023), which showed using two language models: a smaller drafter model, and the larger target model, in a memory bound setting can help accelerate LLM token generation. A key contribution of this work was the generations are lossless, i.e., they follow the target model distribution. Several works build on top of this, extending SpD to recursive speculative decoding (Jeon et al., 2024; Miao et al., 2023), instead of training independent small drafters, extra LM-head were augmented (Cai et al., 2024), draft model was attached to last layer of LLM (Li et al., 2024a; Zimmer et al., 2024), etc. Additionally, some work propose training separate drafters for different downstream tasks to achieve further boost in speed (Kim et al.), training drafters is cumbersome and resource intense as many times small models are simply unavailable (Goel et al., 2024). We propose a plug-and-play, training-free method, based on the limited representation capacity of drafter model, and exploit for reducing computation while maintaining block efficiency with boost in memory-bound speed-up.

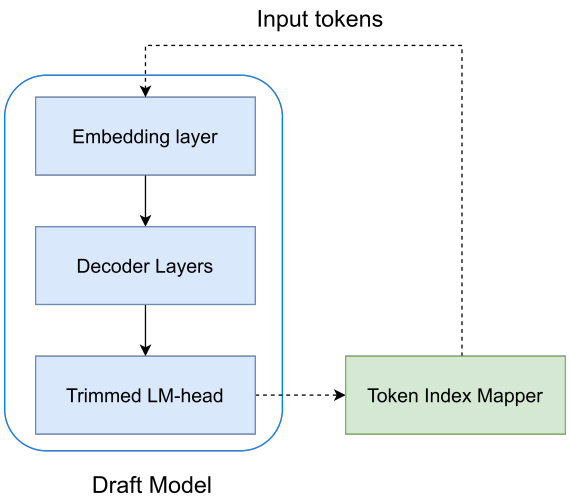

Figure 2: SpD inference with trimmed vocabulary of drafter

Earlier work in (Gagrani et al., 2024) has shown in multimodal setting, that even using only a text-based drafter can achieve good speed-ups and their qualitative analysis shows that drafter primarily predicts easy words such as articles, prepositions, or completing half-generated words.

Recently, AdaptiVocab (Nakash et al., 2025) explored using custom vocabularies for LLM generation, to reduce computation and memory cost for target (or base) model. A concept similar to our approach was independently proposed in (Zhao et al., 2025), although their method is limited to pruning draft LM-head using only raw-text data. Detailed similarities and differences with (Zhao et al., 2025) are discussed in Appendix A.3. It is important to note that the LM head linear layer consumes a significant portion of memory and computation during inference. In contrast to AdaptiVocab, our work focuses specifically on optimizing the LM head of drafter. This design choice ensures that the final performance of the target model remains unaffected, while still leveraging the lossless nature of speculative decoding methods.

3. Method

Based on our observation that only a small subset of tokens occur dominantly in many language modeling tasks (as shown in Figure 1), we propose VOCABTRIM, a training-free and efficient Speculative Decoding (SpD) method that

removes infrequent tokens from the draft model's vocabulary. VOCABTRIM introduces minimal changes to the SpD pipeline and imposes no architectural constraints, facilitating seamless integration into existing SpD techniques, general flow is shown in Figure 2 with algorithm in Appendix C.1.

For a draft model with vocabulary $\mathbb{V}$ and LM head parameter W, VOCABTRIM constructs a trimmed vocabulary $\mathbb{V}^{\text{Trim}}$ and a corresponding trimmed LM head W^{Trim} by running the target model on a calibration dataset $\mathcal{D}$ and selecting the most frequently occurring tokens in $\mathbb{V}$ along with their corresponding rows in W. Formally, vocabulary trimming is defined as follows:

$$\begin{aligned} \mathbb{V}^{\text{Trim}} &= \mathbb{V}[\text{Top-K}(c, k)] \\ W^{\text{Trim}} &= W[\text{Top-K}(c, k), :] \end{aligned} \tag{1}$$

where c is the token frequency counter and k is the desired size of the trimmed vocabulary. The frequency counter c is computed by counting how often each token in $\mathbb{V}$ appears across the calibration dataset $\mathcal{D}$, as shown in Algorithm 1.

Two types of calibration datasets are considered: (a) raw text data, and (b) target model generation. Raw text data is available in ample amounts, even for many evaluation tasks which have train splits. Target model generation may also be readily available as it is used to finetune the drafter (Goel et al., 2024). If not, it may be easily generated by simply letting the target model to generate completions following queries from a dataset. As an ablation, we additionally use drafter generated data (used for finetuning drafter in (Zhou et al., 2023)). We find that the target model generated calibration dataset performs the best (least drop in acceptance rate while maximum increase in memory-bound speed-up) as shown in Table 1.

Note that, in this paper, we restrict VOCABTRIM experiments with Top-K-based trimming of the drafter LM-head. However, our framework is general and also supports choosing based on Top-P, or based on minimum frequency of tokens in calibration datasets. Moreover, other selection criteria for trimmed vocabulary can be: compute resources available, hardware-memory constraints, or maximum allowable drop in accuracy. We leave further exploration including these as a future work.

4. Experiments

To show the efficacy of such trimmed-vocabulary drafter models, we perform experiments using the LLAMA3 models (Grattafiori et al., 2024), note that we perform greedy decoding for target model which implies a token is only accepted if exactly matched with the target model, enabling lossless generations. In the following experiments, we assume a fixed K depending on the evaluation task. We additionally ablate over different sizes of draft LM head (W^{Trim}) to compare the effect on acceptance and speed-up increase. Our experiment setup and other details are as follows:

Algorithm 1 Count Token Frequencies

Require: Calibration dataset $\mathcal{D}$
Initialize counter vector $c \in \mathbb{N}^{|\mathbb{V}|}$ with zeros
for each $x \in \mathcal{D}$ **do**
  $[t_0, t_1, \ldots, t_n] \leftarrow \text{Tokenize}(x)$
  for $i = 0$ to n **do**
    $c[t_i] \leftarrow c[t_i] + 1$
Return c

4.1. Settings

Models: We consider two Llama 3 models of different sizes: Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct. For each target model, we use two kinds of drafter architectures (a) EAGLE-based SpD (Li et al., 2024b), and (b) independent drafter-based SpD (Leviathan et al., 2023). However, in both cases, we use the same draft tokens sampling, draft trees construction, and verification following EAGLE-2.

Drafter Training: For (a), the EAGLE drafter model is trained following EAGLE (Li et al., 2024a), while, for (b), we follow (Goel et al., 2024) to train a 314M standalone drafter (detailed configuration mentioned in Appendix B).

Tasks: We use tasks from Speculative-Decoding benchmark (Spec-Bench) (Xia et al., 2024) which covers several tasks including summarization, coding, and math tasks (from GSM8K (Cobbe et al., 2021)) as well as additional tasks such as function calling (xLAM) (Liu et al., 2024), and open-ended text generation (creative writing subset of Dolly-15k) (Conover et al., 2023).

Performance Metrics: (1) **block efficiency** (τ): average number of tokens generated per block (or target model run), for a block of size γ and input x, the maximum value of $\tau(x)$ can be $\gamma + 1$, (2) memory-bound speed-up (**MBSU**): theoretical speed-up achieved by SpD for a given block efficiency $\tau(x)$ and a relative latency c defined as ratio between number of parameters of draft to target model, i.e., $\text{MBSU}(x) := \frac{\tau(x)}{c\gamma+1}$,

Calibration Datasets: For general text generation tasks such as open-ended text generation, summarization, etc., we use samples from Minipile (Kaddour, 2023) as the raw dataset calibration. For target model-generated calibration, we use instruction-based dataset: OIG-small-chip2 [1] and OpenAssistant [2] for Llama3.2-3B-Instruct and Llama3.1-8B-Instruct respectively. OIG is also used to generate drafter based calibration dataset. Lastly, for function-calling eval-

[1] OIG-small-chip2 huggingface link
[2] OpenAssistant huggingface link

uation task (xLAM), samples from train split are used for raw-dataset, target generations, and draft generation based calibration datasets.

Draft Tree: We use the draft tree with the depth of 3 with top-K$= 8$ and maximum tokens $= 32$ for all generation of both EAGLE drafter and independent drafter.

4.2. Results

Llama-3.2-3B-Instruct results with VOCABTRIM on Spec-Bench are shown in Table 1 for both the EAGLE drafter and the independent drafter. We observe that target-generated calibration dataset-based vocabulary trimming leads to the smallest drop in block efficiency (2-5%) and the largest gains in MBSU (14-18%), followed by the cases of using draft model generation and the raw text. With the independent drafter, in the similar vein, VOCABTRIM with target-generated calibration dataset outperforms the raw text and draft-generated dataset. The block-efficiency drop for target model generated calibration dataset is 1-7% with the MBSU gain of 2-12.3%.

To study the performance of VOCABTRIM on other domains, we additionally conduct experiments on open-ended text generation with relatively long generation, and on a function calling task, for both the EAGLE and independent drafter Table 2. Note that, while the same trimmed vocabulary set can be shared across multiple tasks with similar nature of text (e.g. chat, QA, writing), it may degrade performance on some other tasks with little overlaps in expected target model generation (e.g., coding task Appendix A.2). As such, on xLAM function call task, we use a different trimmed vocabulary with the size of $5K$ based on training-split of xLAM dataset, unlike the trimmed vocabulary used for Dolly with the size of $\sim 32K$ (same as the one used for Spec-Bench). Similar to the previous result, using target-generated calibration dataset gives the highest boost in MBSU with the lowest drop in BE. For the EAGLE drafter, BE drops only 1.4% with the MBSU gain of 19% on Dolly, and 2.6% BE drop on xLAM with MBSU gains of 25%. For the independent drafter, using target-generated calibration dataset drops BE by 1.3% and 0.8% with the MBSU gains of $\sim$ 8% and 11.8% on Dolly and xLAM, respectively.

For Llama-3.1-8B-Instruct, we perform experiments on Spec-Bench using the EAGLE drafter with the raw and the target-generated calibration dataset Table 3. Note that as the base model size is larger, the overall MBSU increases as the ratio of drafter size and target model size decreases. We observe that, even in this scenario, reduction in drafter size via smaller LM head improves MBSU ($8 - 12\%$ on Spec-Bench) with minimal BE ($1 - 4\%$ on Spec-Bench) degradation with target-generated calibration dataset outperforming the others.

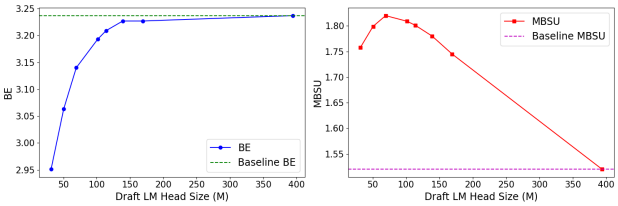

Figure 3: Llama3.2-3B-Instruct performance (BE, MBSU) with different draft LM head sizes.

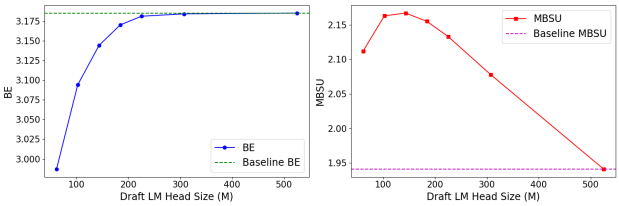

Figure 4: Llama3.1-8B-Instruct performance (BE, MBSU) with different draft LM head sizes.

4.3. Ablation Study

4.3.1. PERFORMANCE OF EAGLE-2 WITH VARIOUS DRAFTER LM HEAD SIZES

In Figure 3 and 4, we show the effects of amount of vocab trimming, i.e., draft LM head sizes, to BE and MBSU on Dolly dataset. As expected, BE increases accordingly as the draft LM head size increases. However, due to the trade-off between drafting cost and improved BE, the MBSU has a sweet spot at around 70M sized W^{Trim} corresponding to $k \sim 23K$ out of $128K$ tokens for Llama-3.2-3B-Instruct, boosting MBSU by 19.7% with only a $\sim 3\%$ drop in BE. Similarly, for Llama-3.1-8B-Instruct target, the highest MBSU is achieved by $143.4M$ sized draft LM-head consisting of $35K$ out of $128K$ tokens leading to only 1.2% drop in block-efficiency with 11.6% improvement in MBSU.

4.3.2. PERFORMANCE OF EAGLE-2 WITH VARIOUS DRAFT TREE HYPERPARAMETERS

We ablate over different tree depth using EAGLE-2 decoder to gauge its impact on VOCABTRIM, with total draft tokens $= 60$. We use top-$K = 32$k for trimmed drafter LM head as used in previous results. The original size of LM-head is 394M while the size of trimmed LM-head is 101.3M. The results are shown in Table 4, where we observe that for different tree-depths, VOCABTRIM gives MBSU boost with minimal drop in performance, the absolute of MBSU increase remains consistent.

Table 1: Spec-Dec Benchmark with Eagle and Independent-drafter based SpD for Llama3.2-3B-Instruct

Method	$LM_{Head,Draft}$	Writing		Roleplay		Math/Reasoning		Coding		Extraction		Translation		Summarization		QA		RAG	
	(M)	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU
Eagle	394.0	3.141	1.475	3.092	1.451	3.494	1.64	3.638	1.708	3.693	1.734	3.3	1.549	3.098	1.454	3.214	1.509	3.23	1.516
+Raw-dataset	101.3	2.974	1.685	3.062	**1.735**	3.421	1.938	3.294	1.866	3.405	1.928	3.14	1.778	2.996	**1.697**	3.098	1.755	3.131	1.774
+Target generated	101.3	3.081	**1.745**	3.038	1.721	3.443	**1.950**	3.434	**1.945**	3.588	**2.032**	3.241	**1.836**	2.993	1.695	3.165	**1.793**	3.133	**1.775**
+Draft generated	101.3	3.057	1.732	3.016	1.708	3.43	1.943	3.379	1.914	3.518	1.993	3.229	1.829	2.98	1.688	3.148	1.783	3.073	1.741
Independent	131.3	3.846	2.765	3.864	2.778	4.434	3.187	3.999	2.875	4.780	3.436	4.221	3.035	3.239	2.328	4.160	2.991	3.943	2.835
+Raw-dataset	33.8	3.682	2.900	3.551	2.797	4.224	3.327	3.592	2.829	4.332	3.412	3.975	3.131	3.097	2.439	3.883	3.059	3.737	2.944
+Target generated	33.8	3.941	**3.104**	3.781	**2.978**	4.379	**3.449**	3.707	**2.920**	4.704	**3.705**	4.147	**3.267**	3.086	**2.431**	4.078	**3.212**	3.769	**2.969**
+Draft generated	33.8	3.905	3.076	3.738	2.944	3.948	2.838	3.534	2.783	4.636	3.652	4.117	3.243	3.037	2.392	4.068	3.204	3.685	2.902

Table 2: Eagle based SpD for Llama3.2-3B-Instruct model on open-ended text generation (DOLLY) and function calling (xLAM) task

Method	DOLLY			xLAM		
	$LM_{Head,Draft}$ (M)	BE	MBSU	$LM_{Head,Draft}$ (M)	BE	MBSU
Eagle	394.0	3.237	1.52	394.0	2.937	1.379
+Raw-dataset	101.3	3.091	1.751	15.4	2.397	1.445
+Target generated	101.3	3.193	**1.809**	15.4	2.860	**1.725**
+Draft generated	101.3	3.173	1.797	15.4	2.777	1.674
Independent	131.3	3.675	2.642	131.3	2.364	1.700
+Raw-dataset	33.8	3.483	2.744	5.1	2.075	1.681
+Target generated	33.8	3.629	**2.858**	5.1	2.344	**1.900**
+Draft generated	33.8	3.589	2.827	5.1	2.314	1.875

5. Conclusion

We propose a new direction of improving performance of drafter-based speculative decoding methods by reducing the size of drafter LM head. Our method VOCABTRIM is a training-free off-the-shelf technique that can be easily integrated with various speculative decoding methods that incorporates LM head layers. Experiments with Llama 3 models on various downstream task shows that drafters can operate on smaller vocabulary space giving significant boost in performance in memory bound token generation process of LLMs.

6. Acknowledgment

We thank Weilin Zhao and Xu Han who kindly pointed out the resemblance of the idea in our paper to their prior work (Zhao et al., 2025). We discuss their similarities and differences in Section 2 and Appendix A.3.

References

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gagrani, M., Goel, R., Jeon, W., Park, J., Lee, M., and Lott, C. On speculative decoding for multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8285–8289, 2024.

Table 3: Spec-Dec Benchmark with Eagle based SpD for Llama3.1-8B-Instruct

Method	$LM_{Head,Draft}$	Writing		Roleplay		Math/Reasoning		Coding		Extraction		Translation		Summarization		QA		RAG	
	(M)	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU	BE	MBSU
Eagle	525.3	3.134	1.910	2.864	1.745	3.300	2.011	3.628	2.211	3.517	2.144	3.072	1.872	2.970	1.810	2.928	1.785	2.962	1.805
+Raw-dataset	135.0	2.942	2.034	2.728	1.886	3.237	2.238	3.264	2.257	3.301	2.282	2.984	2.063	2.857	1.975	2.847	1.968	2.833	1.959
+Target generated	135.0	3.096	**2.140**	2.737	**1.892**	3.323	**2.297**	3.484	**2.408**	3.494	**2.416**	3.040	**2.102**	2.877	**1.989**	2.908	**2.010**	2.868	**1.983**

Table 4: Eagle based SpD for Llama3.2-3B-Instruct model on open-ended text generation (DOLLY) for different tree depth with and without draft model vocabulary trimming

Depth	Baseline		Baseline+VOCABTRIM	
	BE	MBSU	BE	MBSU
5	3.551	1.318	3.504	1.631
6	3.578	1.202	3.510	1.500
7	3.567	1.094	3.510	1.387

Goel, R., Gagrani, M., Jeon, W., Park, J., Lee, M., and Lott, C. Direct alignment of draft model for speculative decoding with chat-fine-tuned llms. *arXiv preprint arXiv:2403.00858*, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jeon, W., Gagrani, M., Goel, R., Park, J., Lee, M., and Lott, C. Recursive speculative decoding: Accelerating llm inference via sampling without replacement. *arXiv preprint arXiv:2402.14160*, 2024.

Kaddour, J. The minipile challenge for data-efficient language models. *arXiv preprint arXiv:2304.08442*, 2023.

Kim, T., Jung, H., and Yun, S.-Y. A unified framework for speculative decoding with multiple drafters as a bandit.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024b.

Liu, Z., Hoang, T., Zhang, J., Zhu, M., Lan, T., Kokane, S., Tan, J., Yao, W., Liu, Z., Feng, Y., et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*, 2024.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Wong, R. Y. Y., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 1(2):4, 2023.

Nakash, I., Calderon, N., David, E. B., Hoffer, E., and Reichart, R. Adaptivocab: Enhancing llm efficiency in focused domains through lightweight vocabulary adaptation. *arXiv preprint arXiv:2503.19693*, 2025.

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 7655–7671, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.456. URL https://aclanthology.org/2024.findings-acl.456.

Zhao, W., Pan, T., Han, X., Zhang, Y., Sun, A., Huang, Y., Zhang, K., Zhao, W., Li, Y., Wang, J., et al. Fr-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. *arXiv preprint arXiv:2502.14856*, 2025.

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

Zimmer, M., Gritta, M., Lampouras, G., Ammar, H. B., and Wang, J. Mixture of attentions for speculative decoding. *arXiv preprint arXiv:2410.03804*, 2024.

A. Discussion

A.1. Future Work

In current work, we rely on limited number of datasets for different tasks, specifically, Minipile (Kaddour, 2023) for computing occurrence of tokens in raw-dataset, and OIG-small-chip2, OpenAssistant datasets for generating target (and draft) model dataset, and xLAM (Liu et al., 2024) train-split for function-calling task. In following works, we will experiment specific downstream tasks using more task-specific domain dataset, for example, using train-split from GSM8k (Cobbe et al., 2021) for math tasks, and using train split from HumanEval (Chen et al., 2021) for coding tasks, we believe this will further reduce the acceptance-rate gap while boosting MBSU. The idea of having separate draft model LM-head for separate downstream task can be considered analogous to low-rank adaptation (LORA) for LLMs (Hu et al., 2022), where for each downstream task, a set of LORA parameters are trained.

A.2. Limitations

We observe that the coding task in Table 1 leads to slightly higher BE drop (5.6% for Eagle-drafter using target based calibration dataset) than other tasks with lower MBSU gains (13.9%). We believe this is due to mismatch in the token distribution used in code-generation compared to the token distribution used in our calibration datasets (general english). Using a code-dataset for calibrating the VocabTrim would be beneficial in this scenario, to further boost the performance.

A.3. Comparison of VOCABTRIM and FR-Spec: Similarities and Differences

It has come to our attention that VOCABTRIM and FR-Spec (Zhao et al., 2025) share the idea of pruning LM head for speculative decoding. In this section, we highlight some key differences between them. First, they commonly identify the LM head as a bottleneck in draft model as motivation for the proposed works. While the induced algorithms resemble to each other, we emphasize that our focus is on the memory bandwidth of drafting process associated with the size of the drafter, regardless of its computational complexity. This is crucial in memory-bound environment, which is often the case in edge devices but not obvious in larger scale devices such as Nvidia GPUs in which cases the methods may lead to only marginal improvements. In this context, we report MBSU (memory-bound speed up) as the primary performance metric and make the performance measurement more consistent with on-edge-device performance.

Second, we propose the use of model-generated datasets, i.e., distillation datasets, to construct the token distribution, under the motivation of distribution matching and leading to the best performance on many cases, while comparing different types of calibration datasets: (a) target-generated, (b) draft-generated, and (c) raw dataset, similarly in (Zhao et al., 2025).

Lastly, our approach is a generic method for speculative decoding method with drafter LM head in the loop, not limited to EAGLE. Thus, we compare both independent drafter model and EAGLE-style drafter model.

B. Independent Draft Model

B.1. Model configurations

The following configurations are used for 314M draft models following (Biderman et al., 2023):

Table 5: Draft model configurations

	Llama 3-Instruct-Drafter (314M, draft)
Layers	4
Attention heads	8
KV heads	8
Intermediate dim	2816
Hidden dim	1024
Activation	SiLU

B.2. Training hyper-parameters

For draft model pretraining, we used deepspeed stage1 on 32 A100 GPUs. Additionally, during pre-training a large batch-size can be used as compared to distillation which requires the target model weight and output consuming a lot of memory. The optimizer used is *AdamW* with *WarmUpDecayLR* learning-rate scheduler, maximum learning rate was 0.0001 while minimum was $1e-6$.

For draft model fine-tuning, we used deepspeed stage 3 with batch-size=40 on 8 A-100 GPUs with maximum learning-rate=0.0003 with same optimizer and scheduler with 2000 warm-up steps.

B.3. Data processing

We preprocess the text data with the tokenizer of the target language model while appending End-Of-Sentence (EOS) token at the end of each input sequence. Furthermore, as a postporcessing step, all the sequences are concatenated into chunks of 4096 length, to maximize training throughput without adding pad tokens.

C. Trimmed draft LM-head

C.1. Algorithm Flow

The VocabTrim follows same inference code as standard SpD systems, with minimal changes related to using a lightweight draft LM-head, and mapping draft model generations back to original vocabulary space as shown in Algorithm 2.

Algorithm 2 SpD with VOCABTRIM

Input: Prompt x, Draft model with trimmed vocab M_{d_trim}, Target model M_t, Draft length k, token index mapper $T_{d \to t}$
Initialize output $y \leftarrow x$
while not end of sequence **do**
  for $j = 1$ to k **do**
    $\hat{z}_j \leftarrow M_{d_\text{trim}}(y, \hat{y}_{<j})$
    $\hat{y}_j \leftarrow T_{d \to t}(\hat{z}_j)$
  Compute probabilities $P_t \leftarrow M_t(\hat{y}_{1:k})$
  for $i = 1$ to k **do**
    if $\hat{y}_i$ is consistent with P_t **then**
      Append $\hat{y}_i$ to y
    else
      Generate $y_i \leftarrow M_t(y)$
      Append y_i to y
      break
Return y