OmniDraft: A cross-vocabulary, online adaptive drafter for on-device speculative decoding

Ramchalam Kinattinkara Ramakrishnan[*, 1] **Zhaocong Yuan**[*, 2],
Shaojie Zhuo[3], Chen Feng[4], Yicheng Lin[5], Chenzheng Su[6], Xiaopeng Zhang[7]

Qualcomm AI Research [†]
{[1]rkinatti, [2]zhaocong, [3]shaojiez, [4]chenf, [5]yichengl, [6]chenzhen, [7]xiaopeng}
@qti.qualcomm.com

Abstract

Speculative decoding generally dictates having a small, efficient draft model that is either pretrained or distilled offline to a particular target model series, for instance, Llama or Qwen models. However, within online deployment settings, there are two major challenges: 1) usage of a target model that is incompatible with the draft model; 2) expectation of latency improvements over usage and time. In this work, we propose OmniDraft, a unified framework that enables a single draft model to operate with any target model and adapt dynamically to user data. We introduce an online n-gram cache with hybrid distillation fine-tuning to address the cross-vocabulary mismatch across draft and target models; and further improve decoding speed by leveraging adaptive drafting techniques. OmniDraft is particularly suitable for on-device LLM applications where model cost, efficiency and user customization are the major points of contention. This further highlights the need to tackle the above challenges and motivates the *"one drafter for all"* paradigm. We showcase the proficiency of the OmniDraft framework by performing online learning on math reasoning, coding and text generation tasks. Notably, OmniDraft enables a single Llama-68M model to pair with various target models including Vicuna-7B, Qwen2-7B and Llama3-8B models for speculative decoding; and additionally provides up to 1.5-2x speedup.

1 Introduction

Unlike traditional auto-regressive generation in LLMs, speculative decoding (SpD) [25, 9] offers a unique advantage to accelerate LLM inference by decoupling the generation phase and verification phase. Speculative decoding generally requires a small but efficient draft model and a large target model. The draft model generates a sequence of proposed tokens to be verified by the target in one shot, amortizing the target model's memory bottleneck in batch inference and attaining better tokens per second throughput. The speedup factor relies not only on the predictability of the generated text like commonly occurring phrases, but also on the alignment between draft and target model. As such, a common practice is to utilize draft and target models from the same model family given their consistency in pretrained data, tokenization and training configurations. Alternatively, one might consider distilling a target model into a smaller model to serve as the drafter [53, 30], which still follows the same principle of better alignment leading to greater speedup.

The tight coupling of draft and target models limits flexibility of model selection and creates additional overhead for draft model distillation and maintenance, especially when deploying LLMs at scale

[*]Contributed equally

[†]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

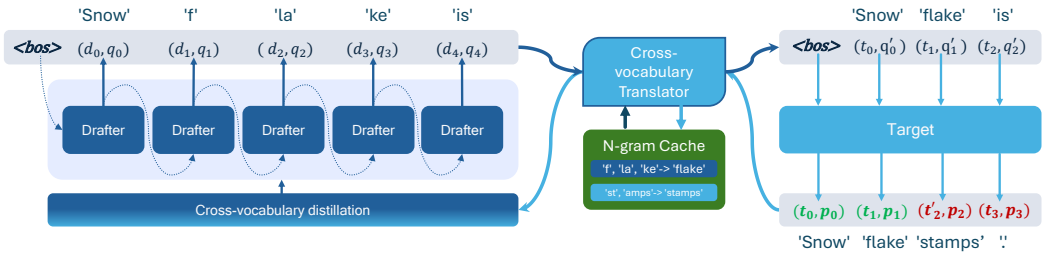

Figure 1: Overview of the OmniDraft framework: during cross-vocabulary speculative decoding, the drafter (Llama-68M) generates multiple tokens d_i with corresponding distributions q_i. Cross-vocabulary translator then converts the drafter tokens into tokens in the vocabulary of the target model (Llama3-8B). In this example, token d_0('Snow') and d_4('is') are directly mapped to target tokens t_0 and t_2, while token d_1('f'), d_2('la') and d_3('ke') are merged into a single target token t_1 ('flake'), since there is a mapping item in the n-gram cache. The translated proposal t_i along with combined probabilities q_i' is verified by the target model, resulting in t_0 and t_1 being accepted while t_2 being rejected and replaced by t_2'. The target outputs tokens and their probabilities p_i are translated into drafter tokens and sent back to drafter for next round of drafting. The n-gram cache is updated by inserting a new unseen item ('st','amps'->'stamps'). Meanwhile, the accepted and corrected tokens from the target model are used to align the drafter through online cross-vocabulary distillation.

across diverse hardware platforms and updating target models overtime. It is compelling to use a universal lightweight model on-device to draft tokens for a broad range of targets models. This universality greatly simplifies deployment, facilitates easy optimization, and allows for rapid model updates. Furthermore, the heterogeneity of on-device and cloud hardware presents an opportunity for hybrid speculative decoding. This enables users to choose between running any target model locally or on the cloud, balancing model performance, inference cost, and privacy concerns.

However, building a universal drafter for speculative decoding presents several unique challenges. Firstly, different family of target models might use tokenizers with different vocabularies. This is natural since target models are typically trained with massive pretrained data and hence a larger vocabulary is needed to include higher-order n-grams or BPE [7] merges. As a result, the vocabulary mismatch breaks the speculative decoding formulation where the draft and target model need to evaluate probabilities over the same set of tokens. A previous work UAG [45] addresses vocabulary mismatch with an intermediate translation layer, but it primarily deals with tokens within the intersection of the drafter and target vocabularies, which might "falsely" reject good tokens from the drafter. Secondly, independent training of the drafter and target models can result in misaligned predicted token distributions, which reduces the acceptance rate during verification. This misalignment diminishes the efficiency benefits of speculative decoding. Although existing research [53, 30, 34] has explored techniques to improve alignment between the drafter and target, the alignment is often done offline with assumption of a fixed target model. In practice, the target model may change due to user personalization or tasks switching, further complicating the alignment issue. Additionally, edge devices usually have limited memory, compute capacity, and power budget. The drafting process must therefore be efficient to maximize the benefits of speculative decoding.

To address the challenges, we propose a scalable speculative decoding framework, named *OmniDraft*, centered on an on-device universal drafter that generates draft tokens for a wide variety of target models. An overview of *OmniDraft* is shown in Figure 1. To tackle the vocabulary mismatch issue and enable cross-vocabulary speculative decoding, we introduce an n-gram cache to store cross-vocabulary mappings from draft tokens to target tokens. By integrating the n-gram cache into the speculative decoding algorithm, we alleviate vocabulary mismatches and achieve a higher acceptance rate for future queries. We further employ online knowledge distillation to improve alignment between the drafter and target. A hybrid distillation loss, combining token-level and distribution level objectives, updates the drafter using the target's accepted and corrected outputs. This enables continuous alignment during speculative decoding, even when the target model changes due to personalization or task switching. To further improve runtime efficiency, we incorporate online adaptive drafting, where the drafter dynamically adjust the number of tokens it proposes

based on predicted confidence. The adaptive drafting balances generation cost and acceptation rate, maximizing throughput under device constraints.

Overall, our OmniDraft framework enables a robust, efficient and flexible speculative decoding system with a universal drafter for on-device applications. Through extensive experiments, we show that a single Llama-68M draft model can be paired with various target models including Vicuna-7B, Qwen2-7B and Llama3-8B models for cross-vocabulary speculative decoding and provides up to 1.5-2x speedup on reasoning, coding and text generation tasks.

Contributions (1) We propose cross-vocabulary speculative decoding via online n-gram cache that translates between drafter and target vocabularies, enabling speculative decoding across models with different tokenizers; (2) We introduce online knowledge distillation with hybrid alignment loss, which updates the drafter using accepted and corrected outputs from the target model to improve alignment and acceptance rate over time; (3) We integrate alignment training with adaptive drafting, allowing the drafter to dynamically adjust draft length based on alignment confidence for improved efficiency and speedup.

2 Related work

Speculative decoding The idea of speculative decoding is proposed and formalized in the pioneer works of [25, 9]. Subsequent works have centered around optimizing different components of the speculative decoding framework. Some have highlighted tree attention to facilitate simultaneous verification of multiple draft sequences such as SpecInfer [32], Medusa [8] and Sequoia [11]. There is also focus on more efficient drafting by using retrieval-based methods as in Lookahead [16], REST [19], NEST [26], RASD [38], or by using dynamic length drafting as in BiLD [22], DISCO [31], AdaEDL [2], SpecDec++ [21], EAGLE2 [28].

More recent works explore speculative decoding in the context of vocabulary adaptation like UAG [45] and AdaptiVocab [35], long-context tasks like LongSpec [50], MagicDec [41], or more efficient draft models as in EAGLE [27], Speculative Streaming [5], and self speculative decoding as in [51], [14], [48].

Distillaton Model distillation in LLMs is crucial for speculative decoding since quality of the draft model dictates the final speed-up. Early works on sequence model distillation include [3, 23, 46]. There are also recent works that have specific focus on LLM distillation such as MiniLLM [18] and GKD [1]. Most closely aligned to our setting are the works of DistillSpec [53] and OSD [30] that aim towards training a better draft model to a given target. Another related line of works is distillation on different tokenizers as in [6, 33, 34].

Online adaptation Compared to model finetuning on a fixed offline training set over multiple epochs, online adaptation focuses on continual learning or few-shot generalization to new data. Frameworks such as ProtoNet [42] and MAML [15] address the few-shot learning problem. Robotics and reinforcement learning also offer insights with works like DAGGER [40], RLˆ2 [13] and PEARL [39] for continuous adaptation to new tasks. There are also works that emphasize online adaptation for LLMs as in [20, 43, 29]. Lastly in speculative decoding, OSD [30] explicitly optimizes for draft model online adaptation which is closest to ours.

3 Methodology

Notation Assume a small draft model M_q and a large target model M_p, let $p(y_t|x, y_{<t})$ and $q(y_t|x, y_{<t})$ be the distributions of next-token predictions at time step t for M_p and M_q respectively, where x represents the prompt prefix and $x, y_{<t}$ represents the context at time step t. For convenience, we use $p(y_t)$ and $q(y_t)$ as shorthands for $p(y_t|x, y_{<t})$ and $q(y_t|x, y_{<t})$ for the remaining sections. k denotes the number of tokens proposed by the drafter M_q and $p'(y_{t+i}) = \text{norm}(\max(0, p(y_{t+i}) - q(y_{t+i})))$ is the residual distribution for the resampling, as per the original SpD algorithm [25, 9]

[53, 30, 1] shows that better alignment between the draft and target model gives higher acceptance rate and hence higher speedup in speculative decoding. To align them, some common distillation losses include supervised finetuning (FT) or sequence level Knowledge Distillation (KD) $\mathcal{L}_{SFT}(\theta) =$

$\mathbb{E}_{(x,y)\sim(X,Y)}[-\log q_\theta(y|x)]$, and supervised KD $\mathcal{L}_{SD}(\theta) = \mathbb{E}_{(x,y)\sim(X,Y)}[\mathcal{D}(M_p||M_q^\theta)(y|x)]$ with a selected choice of divergence metric $\mathcal{D}$.

3.1 Cross-vocabulary N-gram Cache

Normal speculative decoding is infeasible when the draft vocabulary V_q and target vocabulary V_p are different, since the rejection scheme relies on acceptance ratios evaluated on the same token $\min(1, p(y_{t+i})/q(y_{t+i}))$ and the residual distribution $\text{norm}(\max(0, p(y_{t+i}) - q(y_{t+i})))$ also requires per-token probability differences. This foreshadows two issues: 1) tokens without direct mapping between the drafter vocabulary and target vocabulary cannot be handled i.e. the drafter can propose tokens not recognized by the target or vice-versa. 2) target can "falsely" reject good tokens from the drafter. This occurs when the drafter proposes a sequence of (sub-)tokens that constitute a merged token/n-gram in target, but target rejects the sequence since it prioritizes the merged token over the prefix sub-token. This is a byproduct of the tokenization process that optimizes for the longest token in the vocabulary or sequentially applies merge rules to get the longest possible token ([47, 24, 7]).

UAG [45] addresses the first mismatch with a translation layer between the drafter and target and derives the intersection of draft and target vocabularies, where tokens have direct mappings. During proposal stage, UAG suppresses tokens outside of the intersection and converts drafter token ids to target ids with the mapping. After verification, if a token without direct mapping is sampled, the translation layer will invoke the target and draft tokenzier to map the target token to sub-tokens in draft vocabulary. However, UAG cannot solve the second mismatch meaning it only guarantees feasibility of cross-vocabulary speculative decoding but lacks in optimality.

To overcome this, we propose to build a cache of n-grams $\mathcal{C}$ that tracks the instances of target-draft token translations. Denote target tokens $t_i \in V_p$ and drafter tokens $d_i \in V_q$, then the n-grams cache is $\mathcal{C} = \{(t_i, [d_j^i]_{j=1:n})\}$, where each element represents a mapped n-gram instance given the matching context so far $ctx_q, d_1^i, d_2^i, \cdots, d_n^i = \text{tokenize}_q(\text{detokenize}_p(ctx_p, t_i))$. In inference time, we add a postprocessing (pp) stage at the translation layer, where we scan over the proposed draft tokens $d_t, \cdots, d_{t+k-1}$ and merge sub-tokens that hit the n-gram cache. The resulting new sequence $t_t, \cdots, t_{t+m}$ and their draft probabilities $q'(t_t), \cdots, q'(t_{t+m})$ will be under the target vocabulary space and follow the mapping rule

$$t_i, q'(t_i) = \begin{cases} d_i, q(d_i) & \text{if direct mapping, } t_i \leftrightarrow d_i \\ \text{lookup}([d_j^i]_{j=1:n}, \mathcal{C}), \prod_j q(d_j^i) & \text{otherwise} \end{cases} \tag{1}$$

This cross-vocabulary mapping translates the draft sequence to the target vocabulary space, ensuring speculative decoding still functions well with per-token acceptance ratios $\min(1, p(t_{t+i})/q'(t_{t+i}))$. From the perspective of the final matched text, $p(t_i)$ and $\prod_j q(d_j^i)$ would be the probability of producing that specific chunk of text in their respective tokenization space.

However, in the correction stage we require the full distribution for the residual distribution instead of point-wise evaluation of probabilities, which is infeasible since $p(\cdot), q(\cdot)$ work on different space. Hence we enhance the mapping rule 1 as

$$\forall t \in V_p, \quad q'(t) = \begin{cases} \prod_j q(d_j^i) & \text{if n-gram mapped, } t \leftrightarrow [d_j^i]_{j=1:n} \\ q(d_1^i) - \prod_j q(d_j^i) & \text{prefix sub-token of n-gram, } t = d_1^i \\ q(t) & \text{otherwise} \end{cases} \tag{2}$$

We use the mapped probability for the selected n-gram but adjust the prefix sub-token probability by subtracting the n-gram probability. This can be seen as approximately re-allocating the original probability mass assigned to prefix sub-token d_1^i under the draft distribution $q(\cdot)$, between the "new" n-gram token and the prefix sub-token under the modified draft distribution $q'(\cdot)$. This is also related to the known problem of tokenization bias [37]. Using mapping 2, we at least ensure point-wise, approximate correctness of the n-gram token for the residual distribution $\text{norm}(\max(0, p(t_{t+i}) - q'(t_{t+i})))$. Note that we can apply 2 to the n-grams sampled and matched from the current speculative round. Evaluating all other n-grams require re-running drafter at the step of rejection which is impractical. We summarize the modified speculative decoding with our proposed mappings in Algorithm 1.

Algorithm 1 Cross-vocabulary Speculative Decoding

Given draft model $q(\cdot)$, target model $p(\cdot)$, n-gram cache $\mathcal{C}$
Given draft length k, max length T, prompt x
Initialize $t \leftarrow 0, ctx_q, ctx_p \leftarrow x$
while t < T **do**
  for $i = 1 : k$ **do**
    Sample draft auto-regressively $d_{t+i} \sim q(d_{t+i}|ctx_q, d_{<t+i})$
  end for
  Apply translation mapping 12 to get m proposed tokens and draft probabilities in target space
    $t_t, \cdots, t_{t+m-1}, q'(t_t), \cdots, q'(t_{t+m-1})$
  In parallel, compute target probabilities on mapped tokens
    $p(t_t), \cdots, p(t_{t+m})$
  Apply rejection sampling with acceptance ratios $\min(1, p(t_{t+i})/q'(t_{t+i}))$ and correction residual distributions $\text{norm}(\max(0, p(t_{t+i}) - q'(t_{t+i})))$ to get n accepted tokens
    $t_t, \cdots, t_{t+n-1}$ for some accepted length n
  if $n == m$ **then**
    Sample free token $t_{t+m} \sim p(t_{t+m})$ and add to accepted tokens, $n \leftarrow m + 1$
  end if
  Apply reverse translation to get accepted p tokens in draft space
    $ctx_q, d_t, \cdots, d_{t+p-1} = \text{tokenize}_q(\text{detokenize}_p(ctx_p, t_t, \cdots, t_{t+n-1}))$
  Add to n-gram cache $\mathcal{C}$ if there exists unseen n-gram instance
  $t \leftarrow t + n$, update ctx_q, ctx_p
end while
Return results

3.2 Cross-vocabulary Distillation

Using the n-gram cache with the approximate distribution mapping helps to draft and verify n-gram tokens as if operating under the target vocabulary directly. To extend it for online adaptation, we propose a hybrid distillation framework that progressively aligns the draft and target model on both direct mapping tokens and n-gram tokens. Given the online setting, we have limited access to the target model so we distill on the draft model generated data, or simply on-policy data similar to GKD [1]. We employ reverse KL on direct mapping tokens for richer supervision signals, but use maximum log-likelihood (NLL) on n-gram tokens since we only have reliable point-wise evaluation of probabilities on those tokens. Overall, our proposed hybrid distillation loss is

$$\mathcal{L}_{\text{cross\_vocab\_distill}}(\theta) = \mathcal{L}_{\text{DM}}(\theta) + \lambda \mathcal{L}_{\text{N-gram}}(\theta) \tag{3}$$

$$= \mathop{\mathbb{E}}_{\substack{x\sim X, d_i \sim q(\cdot), \\ t_i, q' \leftarrow \text{mapping}(d_i, q)}} \left[\mathcal{D}_{KL}(q'_\theta || p)(t_i|x) \mathcal{I}_{\text{DM}}(d_i) - \lambda \log q_\theta(d_i|x) \mathcal{I}_{\text{N-gram}}(d_i) \right] \tag{4}$$

where $\mathcal{I}_{\text{DM}}, \mathcal{I}_{\text{N-gram}}$ are the indicator functions to identify if current token is part of direct mapping or n-gram; this is implemented as binary masks in practice. Note that the KL loss term corresponds to direct mapping token $t_i \leftrightarrow d_i$, and divergence is computed in the target vocabulary space given $q(\cdot)$ is elevated to $q'(\cdot)$ via the translation mapping. The NLL loss term however operates in drafter vocabulary space to increase likelihoods of drafter tokens which constitute an n-gram accepted by the target during inference.

The parameter λ can either be a hyperparameter to account for ratio imbalance between direct mapping tokens and n-gram tokens, or can be a dynamic weight such as the verified target probability of the n-gram. The latter leads to a loss term of $\lambda \mathcal{L}_{\text{N-gram}}(\theta) = -p(t_i) \log q_\theta(d_i|x)$, which can be treated as the point-wise KL evaluated on the n-gram token.

Moreover, it is possible to extend the NLL or point-wise KL loss to an approximate KL loss using mapping 2 as $\mathcal{L}_{\text{N-gram}} = \mathcal{D}_{KL}(q'_\theta(t_i|x)||p(t_i|x))$. This is equivalent to using KL on the intersection tokens plus the n-gram and it's first sub-token. Due to the additional components on the intersection tokens and the fist sub-token, this approximate KL loss can provide richer learning signal. Empirically we only observed minimal improvement from the approximate KL loss, as shown in section 4.3.4.

3.3 Online Adaptive Drafting

One observation in performing cross-vocabulary speculative decoding is that we are implicitly shortening the proposal draft length, since multiple sub-tokens map to a single n-gram token. We then explicitly incorporate adaptive drafting to gain even better speedup for on-device speculative decoding focusing on the same vocab setting. We adopt the framework in SpecDec++ [21] where a lightweight head network $f_\phi(\cdot)$ predicts the acceptance rate of the current proposed token. The acceptance prediction head takes the embedding of the proposed token e_i as input, and is trained using weighted BCE loss $\mathcal{L}_{\text{adapt}}$ with acceptance ratios $\min(1, p(y_i)/q(y_i))$ as labels. It then controls if to early exist based on the cumulative probability of at least one proposed token getting rejected and a given stopping threshold γ.

$$P(y_i \text{ accepted} | y_{<i} \text{ accepted}) = \text{sigmoid}(f_\phi(e_i)) \tag{5}$$

$$P(\exists 1 \leq i \leq k,\ s.t.\ y_i \text{ rejected}) = 1 - \prod_{i=1}^{k} P(y_i \text{ accepted} | y_{<i} \text{ accepted}) \tag{6}$$

$$P(\exists 1 \leq i \leq k,\ s.t.\ y_i \text{ rejected}) > \gamma \ \ \Rightarrow \ \ \text{exit} \tag{7}$$

However in online adaptation, the labels are subject to change since the draft model is continuously finetuned with distillation loss $\mathcal{L}_{\text{distill}}$ to align with the target, which could cause distribution shift for dynamic drafting.

We propose two variants of online adaptive drafting. The first one performs draft model alignment and acceptance prediction head training jointly at each update step $\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{distill}} + \mathcal{L}_{\text{adapt}}$. The second one interleaves two trainings such that we perform multiple acceptance prediction updates per draft model alignment update, aiming to mimic slowly moving labels to reduce distribution shift. Unlike the first variant that performs both updates jointly using the online data batch, for the second variant we keep a larger buffer for the acceptance prediction updates which includes data from previous batches. This helps to enhance training stability and adaptation speed for the acceptance prediction.

4 Results

Models To show the efficacy of OmniDraft on the setting of a single drafter for multiple targets, we fix the drafter to be Llama-68M [32] and the target model to be Llama3-8B [17], Qwen2-7B [49] for the cross vocabulary results, as well as Vicuna-7B [52] for the same family vocabulary evaluation.

Tasks We perform online distillation across 4 tasks: GSM8K [12], Alpaca [44], XSum [36]and a combined MBPP+HumanEval [4][10] datasets. Each task has a dedicated train and test set or we slice out the a portion of the train set as the test set. For the MBPP+HumanEval, we combine the two datasets to add some more diversity to the data for the coding tasks. The training is conducted for a specific number of steps (<1 epoch) across each of the tasks as per general online adaptation setting. All the tokens will contribute to the loss calculation including the tokens that were accepted by the target as they would provide option for improved alignment between the two models. Moreover, all experiments are performed with temperature 0.01, unless specified otherwise. We also include the setting for online adaptation of the drafter using LoRA across all the tasks. Using dynamic adapter switching we can pair the same drafter with any target across any of the task which is the ideal scenario for on-device online speculative decoding.

Evaluation Metrics

- **Speedup** - Walltime acceleration rate, measures the improvement in tokens-per-second throughput. We follow Medusa's [8] convention.
- **Acceptance Rate** - Ratio of accepted tokens to proposed tokens averaged over speculative decoding steps, measures alignment between draft and target model.

Notation We refer to SpD_{DM} as the baseline speculative decoding which uses direct mapping between vocabularies. N-gram postprocessing (pp) refers to using the N-gram cache as a postprocessing technique without directly training on it as we proposed in Algorithm 1. N-gram hit refers to the average number of successful N-gram cache lookups that were accepted by the target per speculative decoding step.

4.1 Cross-vocabulary Online Distillation

Table 1 shows the results on the test set after training. Across all the tasks, $\mathcal{L}_{\text{DM}} + \mathcal{L}_{\text{N-gram}}$ approach perform better than training only on $\mathcal{L}_{\text{DM}}$. Overall, this indicates that the additional $\mathcal{L}_{\text{N-gram}}$ plays a significant role in improving the acceptance rate. Moreover, $\mathcal{L}_{\text{DM}} + \mathcal{L}_{\text{N-gram}}$ with LoRA finetuning performs reasonably well across all of the tasks when compared to the baseline. The largest speedup is obtained for the GSM8K dataset for both the target models, with XSum being the least improved task. We observe XSum could also be improved by increasing the number of training samples as a scaling effect. Figure 2 portrays the training dynamics across all tasks for both the acceptance rate and speedup metrics. For all the experiments, we can see the metrics improve as training proceeds. The instability seen in some of the LoRA curves could indicate that training hyper-parameters are not fully optimized or that training plateaus quicker for certain tasks, which could also explain why there is still a small gap between the LoRA and the full finetuned model performance.

Table 1: Performance on Cross-vocabulary Distillation with Llama-68M and two different targets

Target	Method	GSM8K		MBPP+HumanEval		Alpaca		XSum	
		Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup
Llama3-8B	SpD_{DM}	0.10	0.94x	0.09	1.03x	0.09	0.96x	0.11	0.91x
	$\mathcal{L}_{\text{DM}}$	0.32	1.58x	0.22	1.26x	0.16	1.25x	0.20	1.20x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$	**0.42**	**1.70x**	**0.27**	**1.33x**	**0.20**	**1.30x**	**0.24**	**1.24x**
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ + LoRA	0.37	1.59x	0.19	1.28x	0.17	1.21x	0.23	1.21x
Qwen2-7B	SpD_{DM}	0.14	1.04x	0.09	0.91x	0.13	1.01x	0.12	0.96x
	$\mathcal{L}_{\text{DM}}$	0.33	1.50x	0.22	1.29x	0.17	1.25x	0.19	1.16x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$	**0.37**	**1.61x**	**0.26**	**1.36x**	**0.20**	**1.30x**	**0.22**	**1.22x**
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ + LoRA	0.31	1.41x	0.21	1.21x	0.18	1.25x	0.22	1.21x

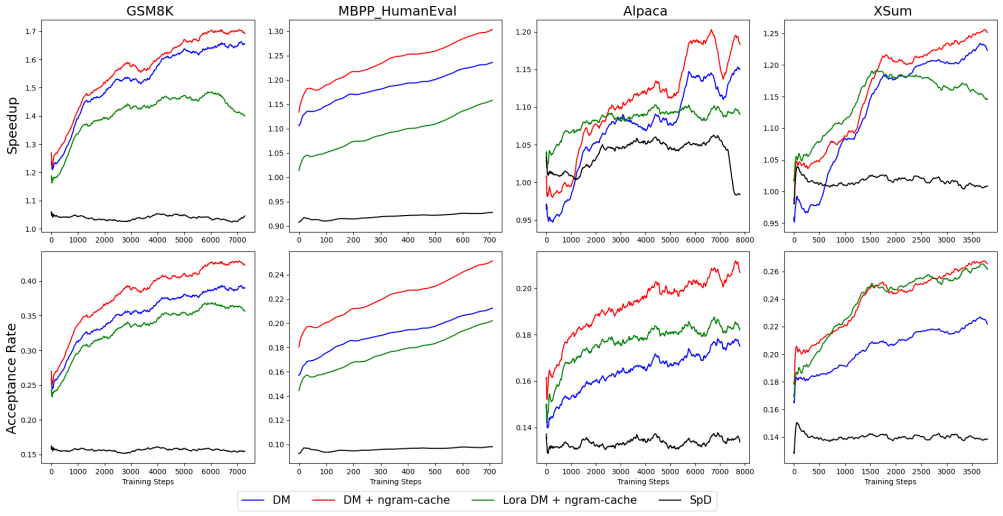

Figure 2: Cross-vocabulary SpD online distillation on Llama-68M with Qwen2-7B as target

4.2 Online Adaptive Drafting

Table 2 shows the test results of online adaptive drafting with ablations to the different training variants. Across all tasks we observe a consistent increase in acceptance rate. This is the combined effect of better model alignment due to distillation, and also the acceptance prediction head learning to exit early when there is a higher chance for proposal rejection

In terms of speedup, we can also see improvement in most tasks except on GSM8K where the distill-only baseline outperforms both adaptive drafting variants. This could be due to the latency reduction over the number of the proposed tokens are not sufficient in comparison to the increase in successful accepted tokens with longer draft length. We hypothesize it could be due the difficulty in training the acceptance prediction head in an online setting. This involves optimizing with respect to

changing labels and training only on incoming data without a stable corrective feedback compared to offline training. By end of the one epoch training, the acceptance prediction head might not have converged well to align with the current distilled drafter, leading to sub-optimal performance.

Finally, we also observe the interleaved variant has better speedup than the joint variant on average (similar in Alpaca and Xsum, larger increment in GSM8K and MBPP+HumanEval). But the joint variant has higher acceptance rates over all tasks, indicating it might be underestimating the acceptance probability, leading to wrongful early exit. This could be a direct consequence of the previously mentioned training challenges, which is alleviated with the interleaved variant with more stable training.

Table 2: Performance on Online Adaptive Drafting with Llama-68M and Vicuna-7B.

Target	Method	GSM8K		MBPP+HumanEval		Alpaca		XSum	
		Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup
Vicuna-7B	SpD	0.21	1.44x	0.14	1.22x	0.20	1.44x	0.20	1.42x
	Adapt Only	0.38	1.50x	0.28	1.34x	0.38	1.52x	0.38	1.51x
	Distill Only	0.42	**2.20x**	0.35	1.92x	0.25	1.57x	0.23	1.53x
	Joint Distill + Adapt	**0.61**	2.08x	**0.51**	1.91x	**0.44**	**1.61x**	**0.42**	**1.59x**
	Interleaved Distill + Adapt	0.52	2.15x	0.48	**1.94x**	0.41	1.60x	0.38	1.58x

4.3 Ablations Studies

4.3.1 Scalability to Larger Target and Draft Models

We perform an ablation on scaling the target and draft model and compare the various metrics and whether it follows a similar trend. As shown in Table 3, we used a new family of LLMs, the Qwen2.5 series of models of larger sizes including 7B, 14B and 32B parameters. Datasets used are GSM8k and MBPP+HumanEval. We also performed analysis on a larger drafter of size 160M parameters. The performance on these larger models are consistent with our previous results, with larger target models resulting in 2x improvement on the GSM8K dataset. The Omnidraft framework demonstrates strong scalability as the size of the target LLM increases, as shown in the table. While the gap between the drafter and the target model grows with model size, the drafter remains the limiting factor. During training, the drafter progressively aligns with the target; however, once it reaches its alignment capacity, further increase in target size continues to yield speedup improvements, while the acceptance rate plateaus. Regarding the larger drafter, the Llama-160M parameter model was chosen after taking into consideration the increased latency of around 3x of the drafter model compared to Llama-68M. Overall, the larger drafter model does introduce a reduction in the overall speedup across all target models, however, due to its enhanced potential capability, the acceptance rate does improve across all the target models.

4.3.2 N-gram Cache Memory Footprint

We also analyzed the various cache size at the end of training for each of the reported tasks as shown in Table 4. Overall n-gram cache size across tasks remain relatively small compared to the size of the draft or target model. This indicates potential feasibility for on-device setting given the small overhead.

4.3.3 Effectiveness of N-gram Cache

In this section, we focus on the impact of the N-gram cache to different training variants as per Table 5. We perform ablation on a subset of GSM8K dataset (4k samples) across multiple techniques. It can be seen that the baseline SpD_{DM} performs poorly which indicates the mismatch between the drafter and the target model alignment for the pretrained models. We also capture all the possible N-gram matches during the SpD_{DM} baseline for the train set, and then use the cache as a lookup for an improved baseline in the SpD_{DM} + N-gram_{pp}. As such there is a cache hit of 0.87, which indicates that the baseline model without any pretraining can still benefit with the N-gram cache. We also train the model to improve the alignment using $\mathcal{L}_{\textbf{DM}}$ and although without the N-gram, we can still see a large improvement over the baseline. Moreover. when we train with N-gram_{pp}, there is a further improvement on the overall speedup. Finally, we train using both $\mathcal{L}_{\textbf{DM}} + \lambda\mathcal{L}_{\textbf{N-gram}}$, which provides the best results across all techniques on different draft length k.

Table 3: Comparison of Cross-vocabulary Distillation Performance with Llama-68M and Llama-160M across three target models

Target	Method	GSM8K		MBPP+HumanEval	
		Acc Rate	Speedup	Acc Rate	Speedup
Qwen2.5 7B	SpD_{DM} (68M)	0.15	1.02x	0.10	0.94x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (68M)	**0.401**	**1.66x**	**0.27**	**1.33x**
	SpD_{DM} (160M)	0.18	0.70x	0.13	0.60x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (160M)	**0.47**	**1.12x**	**0.32**	**0.90x**
Qwen2.5 14B	SpD_{DM} (68M)	0.15	1.17x	0.10	1.14x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (68M)	**0.407**	**1.92x**	**0.272**	**1.57x**
	SpD_{DM} (160M)	0.178	0.89x	0.13	0.84x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (160M)	**0.472**	**1.40x**	**0.33**	**1.19x**
Qwen2.5 32B	SpD_{DM} (68M)	0.153	1.30x	0.10	1.23x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (68M)	**0.42**	**2.05x**	**0.274**	**1.71x**
	SpD_{DM} (160M)	0.187	1.03x	0.133	0.97x
	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ (160M)	**0.49**	**1.62x**	**0.335**	**1.40x**

Table 4: Summary of final N-gram cache size across tasks with Llama-68M drafter and Qwen2-7B target after online inference and distillation. Cache memory (MB) is derived from pympler.asizeof.asizeof().

	GSM8K	MBPP+HumanEval	Alpaca	XSUM
Training Samples	7473	910	8000	4000
Cache Size (#n-grams)	5569	2238	20339	17013
Cache Memory (MB)	1.372	0.501	4.569	3.924

4.3.4 Distillation Loss Comparisons

The different loss variants also provide different levels of performance on the test set as shown in Table 6. When trained only on the $\mathcal{L}_{\text{N-gram}}$, the training is very unstable. This could either be due to the number of n-grams being substantially smaller than the direct mapping tokens within most datasets, or the training requires additional constraints to direct towards a minima. Consequently, training on the combined loss provides better performance metrics across temperatures as well. The final $\mathcal{L}_{\text{DM}}$ KL + $\lambda\mathcal{L}_{\text{N-gram}}$ KL, 2, provides slightly better results on temperature $= 1$ indicating the impact of the additional KL over the intersection vocabulary which is beneficial for the sampling process. However, we noticed tuning the scaling factor λ becomes critical and hence we use Equation 1 for all of the experiments since it was much more stable across all tasks. We fix a $\lambda = 0.2$ for all the tasks, across all experiments.

4.3.5 Adaptive Drafter Initialization and Thresholds

Two additional aspects in training and using adaptive drafting are the acceptance prediction head initialization and the stopping threshold for early exit. Before the joint training of model distillation and adaptive drafting, we can pretrain the acceptance prediction head on an offline dataset, while keeping the drafter fixed. This should ideally capture some priors of the draft-target alignment and provide a better initialization for the online joint training. To verify this, we pretrain the acceptance prediction head on the Alpaca dataset for 1 and 3 epochs respectively, and use the two checkpoints as initialization for online training. We observe mixed results where the pretrained initializations harm performance on GSM8K and XSum, and only improve on MBPP+HumanEval by small margins. We suspect MBPP+HumanEval has a closer affinity in data distribution to Alpaca while the other two do not, which leads to negative transfer with pretraining.

The stopping threshold γ in adaptive drafting also plays a key role for speedup performance. In Alpaca and XSum we observe a conservative threshold of $\gamma = 0.3$ is sufficient to attain better speedup than baselines, while in GSM8K and MBPP+HumanEval we need a more relaxed threshold

Table 5: Effect of n-grams cache with Llama-68M and Llama3-8B on GSM8K (subset)

Metrics	SpD$_{DM}$	SpD$_{DM}$ + N-gram$_{pp}$	$\mathcal{L}_{\text{DM}}$	$\mathcal{L}_{\text{DM}}$ + N-gram$_{pp}$	$\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$
		$k = 3$			
Acc Rate	0.16	0.20	0.40	0.42	**0.46**
Speedup	1.04x	1.16x	1.59x	1.61x	**1.66x**
Avg n-gram hit	0	0.87	0	0.48	**2.40**
		$k = 4$			
Acc Rate	0.12	0.16	0.32	0.35	**0.41**
Speedup	1.01x	1.11x	1.51x	1.54x	**1.61x**
Avg n-gram hit	0	1.49	0	0.75	**3.66**

Table 6: Training Loss comparisons with Llama-68M and Llama3-8B on GSM8K (subset)

Metrics	$\mathcal{L}_{\text{N-gram}}$ NLL	$\mathcal{L}_{\text{DM}}$ NLL + $\mathcal{L}_{\text{N-gram}}$ NLL	$\mathcal{L}_{\text{DM}}$ KL +$\mathcal{L}_{\text{N-gram}}$ NLL	$\mathcal{L}_{\text{DM}}$ KL + $\mathcal{L}_{\text{N-gram}}$ KL
		Temperature $= 0.01$		
Acc Rate	0.090	0.368	0.375	**0.376**
Speedup	0.86x	1.51x	**1.57x**	**1.57x**
		Temperature $= 1$		
Acc Rate	0.070	0.271	0.273	**0.275**
Speedup	0.79x	1.23x	1.27x	**1.31x**

of $\gamma = 0.7$ to achieve comparable speedup. We also observe applying a relaxed threshold on model trained on a stricter threshold improves speedup performance, indicating adaptive drafting could be prone to underestimating the maximum acceptance draft length. It also suggests the actual stopping threshold to be used for inference should be chosen based on the task and then adjusted based on online performance.

4.4 Limitations

While most of our results indicate the potential of our methodology and the OmniDraft framework, some potential limitations still require additional research. 1) Although we are training on incoming data for online adaptation of the drafter, since it is limited to a single iteration of the data stream, there is still potential for instability on new unseen data. 2) We currently use the full n-gram cache per task per target model, however, should memory become a bottleneck, it would require optimized cache eviction policy to cater to edge devices. 3) Special tokens that do not have direct mapping currently would require some additional effort to handle. Consequently, this would make it less seamless to integrate multi-modal tasks. As part of our future plan of action, although cross-vocabulary speculative decoding already implicitly includes the adaptive proposal length due to n-gram merge, we are working towards incorporating an explicit adaptive head in the cross-vocabulary setting.

5 Conclusion

In this work we propose the OmniDraft framework that leverages n-gram cache and hybrid distillation loss to enable cross-vocabulary speculative decoding. We show how the draft model can be aligned to different target models with different vocabulary space via online adaptation, and we further showcase online adaptive drafting to get additional speedup. Our empirical results also show good performance across all metrics. Overall, OmniDraft shows great potential and could pave way to new on-device LLM applications.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Sudhanshu Agrawal, Wonseok Jeon, and Mingu Lee. Adaedl: Early draft stopping for speculative decoding of large language models via an entropy-based lower bound on token acceptance probability. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pages 355–369. PMLR, 2024.

[3] Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710, 2022.

[4] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[5] Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pages 395–413. PMLR, 2024.

[6] Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: the universal logit distillation loss for llms. *arXiv preprint arXiv:2402.12030*, 2024.

[7] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*, 2020.

[8] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, pages 5209–5235. PMLR, 2024.

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

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[11] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *CoRR*, 2024.

[12] 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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rlˆ 2: Fast reinforcement learning via slow reinforcement learning. 2016.

[14] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, 2024.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[16] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *International Conference on Machine Learning*, pages 14060–14079. PMLR, 2024.

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

[18] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

[19] Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 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, 2024.

[20] Nathan Hu, Eric Mitchell, Christopher D Manning, and Chelsea Finn. Meta-learning online adaptation of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4418–4432, 2023.

[21] Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.

[22] Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 39236–39256, 2023.

[23] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1317–1327, 2016.

[24] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

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

[26] Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. Nearest neighbor speculative decoding for llm generation and attribution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[27] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 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, 2024.

[28] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, pages 28935–28948, 2024.

[29] Juhao Liang, Ziwei Wang, Zhuoheng Ma, Jianquan Li, Zhiyi Zhang, Xiangbo Wu, and Benyou Wang. Online training of large language models: Learn while chatting. *arXiv preprint arXiv:2403.04790*, 2024.

[30] Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *Forty-first International Conference on Machine Learning*.

[31] Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic speculation lookahead accelerates speculative decoding of large language models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pages 456–467. PMLR, 2024.

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

[33] Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. *arXiv preprint arXiv:2405.07883*, 2024.

[34] Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Cross-tokenizer distillation via approximate likelihood matching. *arXiv preprint arXiv:2503.20083*, 2025.

[35] Itay Nakash, Nitay Calderon, Eyal Ben David, Elad Hoffer, and Roi Reichart. Adaptivocab: Enhancing llm efficiency in focused domains through lightweight vocabulary adaptation. *arXiv preprint arXiv:2503.19693*, 2025.

[36] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

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

[38] Guofeng Quan, Wenfeng Feng, Chuzhan Hao, Guochao Jiang, Yuewei Zhang, and Hao Wang. Rasd: Retrieval-augmented speculative decoding. *arXiv preprint arXiv:2503.03434*, 2025.

[39] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[40] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[41] Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*, 2024.

[42] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[43] Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. *arXiv preprint arXiv:2403.04317*, 2024.

[44] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[45] Nadav Timor, Jonathan Mamou, Daniel Korat, Moshe Berchansky, Oren Pereg, Gaurav Jain, Roy Schwartz, Moshe Wasserblat, and David Harel. Accelerating llm inference with lossless speculative decoding algorithms for heterogeneous vocabularies. 2025.

[46] Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-divergence minimization for sequence-level knowledge distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10817–10834, 2023.

[47] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[48] Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. Swift: On-the-fly self-speculative decoding for llm inference acceleration. *arXiv preprint arXiv:2410.06916*, 2024.

[49] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[50] Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. Longspec: Long-context speculative decoding with efficient drafting and verification. *arXiv preprint arXiv:2502.17421*, 2025.

[51] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 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, 2024.

[52] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

[53] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Section 4.4 discusses the limitations of our methodology.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper focuses on enabling cross vocabulary online speculative decoding and providing analysis and insights on the performances. There is no associated theoretical proof.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This paper provides all the details of the experiments and the necessary information for reporducibility. The details are provided in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper uses open-source datasets, which are all available on huggingface. Links to all datasets are provided. We definitely want to provide access to code. However, it takes time for corporate legal team to review and approve. If reviewers feel necessary, we will try our best to accelerate the process of releasing code

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer:[Yes]

   Justification: All experimental details and results are provided

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We have a multitude of experiments done across different models and across different tasks. However for the main tables in the graph, we have done the evaluation by averaging 3 runs each with a different seed and reported the number in the main table. We have also done some ablations on some experiments for further analysis.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All our experiements are done on a single NVIDIA-A100 40GB RAM GPU, for both training and inference.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper provides research results and analysis on impact of cross vocab online speculative decoding to speed up LLM inference. We do not see any negative societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: All the experiments conducted in the paper are based on open source datasets

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All references on data source are provided

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Implementation Details

This appendix provides detailed information about the implementation and training setup used in our experiments.

A.1 Model and Hardware Details

Throughout our work, we use the environment setup with NVIDIA A100 GPU (40/80GB), PyTorch 2.1.0 framework, CUDA version 12.1, and Ubuntu 22.04 LTS. Our models used include Llama-68M [32], Llama3-8B [17], Qwen2-7B [49] and Vicuna-7B [52]. We train the Llama-68M model in our experiments. Their model details are listed in the following.

Table 7: Model hyperparameters

Hyperparameter	Value
model	Llama-68M
layers	2
hidden size	768
attention heads	12
activation function	SiLU

Table 8: Model latency and vocabulary statistics

Model	Wall-time per Step (s)	Vocabulary Size	Intersection with Llama-68M
Llama-68M	0.00203	32000	32000
Qwen2-7B	0.02667	152064	22275
Llama3-8B	0.02846	128256	22416
Vicuna-7B	0.02683	32000	32000

A.2 Training Details

Our experiments use GSM8K [12], Alpaca [44], XSum [36] and a combined MBPP+HumanEval [4][10] datasets. We also show the major hyperparameters used across the experiments. For evaluation, we perform three runs with different seeds and report the average in the main results sections.

Table 9: Dataset details

Dataset	GSM8K	MBPP+HumanEval	Alpaca	XSum
train	8K	1K	8K	4K/8K
test	200	228	100	100

Table 10: Training hyperparameters

Hyperparameter	Value
batch size	8
learning rate (LR)	1e-4/2e-5
LR scheduler	constant
optimizer	AdamW
β_1	0.9
β_2	0.999
weight decay	0.01/0
epochs	1 (online)
mixed precision	FP16
LoRA rank	32
temperature	0.01

B Additional Experiment Details

B.1 Speedup Metric

We follow the convention in Medusa [8] to define the speedup metric. Given **acceleration rate** as the average number of tokens decoded per decoding step and **overhead** as the average per step latency of the proposed model divided by that of the vanilla model, **speedup** refers to the wall-time acceleration rate and can be computed with **speedup** $=$ **acceleration rate**$/$**overhead**.

B.2 Adaptive Drafting Training

We show the training curves for adaptive drafting across all tasks. It can be seen that our proposed variants of joint or interleaved align + adapt training have the best performances in both average acceptance rate ans speedup.

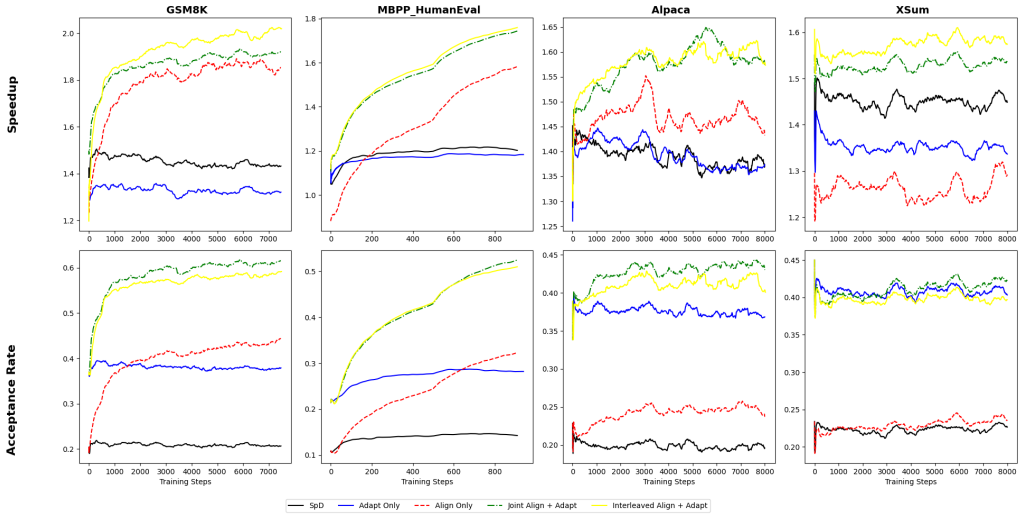

Figure 3: Online Adaptive Drafting training plot of Llama-68M vs Vicuna-7B

B.3 Adaptive Drafting LoRA Training

To showcase the idea of *"one drafter for all"*, we further show results of online adaptive drafting with LoRA used in distillation in Table 11. We observe slightly lower performances across tasks compared to distillation with full fine-tuning as expected. But each of the distillation or distillation with adaptive drafting baseline with LoRA still outperforms normal speculative decoding. And distillation with adaptive drafting outperforms the distillation only baseline, except on GSM8K where the gap is minimal.

By demonstrating the compatibility of LoRA with our proposed online distillation and adaptive drafting training, we empower a single draft model to serve as the backbone and selectively fine-tune LoRA modules and acceptance prediction heads for any potential target model. This would strike a good balance to keep minimal memory overhead while retaining the benefits of greater speedup and flexibility, which are ideal for on-device applications.

Table 11: Performance on Online Adaptive Drafting using LoRA with rank 32 with Llama-68M and Vicuna-7B.

Target	Method	GSM8K		MBPP+HumanEval		Alpaca		XSum	
		Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup	Acc Rate	Speedup
Vicuna-7B	SpD	0.21	1.44x	0.14	1.22x	0.20	1.44x	0.20	1.42x
	LoRA Distill Only	0.37	**1.95x**	0.24	1.52x	0.25	1.54x	0.23	1.49x
	Joint LoRA Distill + Adapt	**0.49**	1.87x	**0.39**	1.59x	0.39	**1.58x**	**0.41**	1.54x
	Interleaved LoRA Distill + Adapt	**0.49**	1.94x	0.38	**1.61x**	**0.42**	**1.58x**	0.39	**1.55x**

B.4 Cross-vocabulary SpD online distillation on Llama-68M with Llama3-8B as target

Figure 4 showcases the training dynamics of Llama-68M with Llama3-8B as the target model. Similar to the previous Figure 2, the pattern is very much matched. Overall, across all the tasks, our methods are able to improve upon the SpD baseline significantly as training progresses.

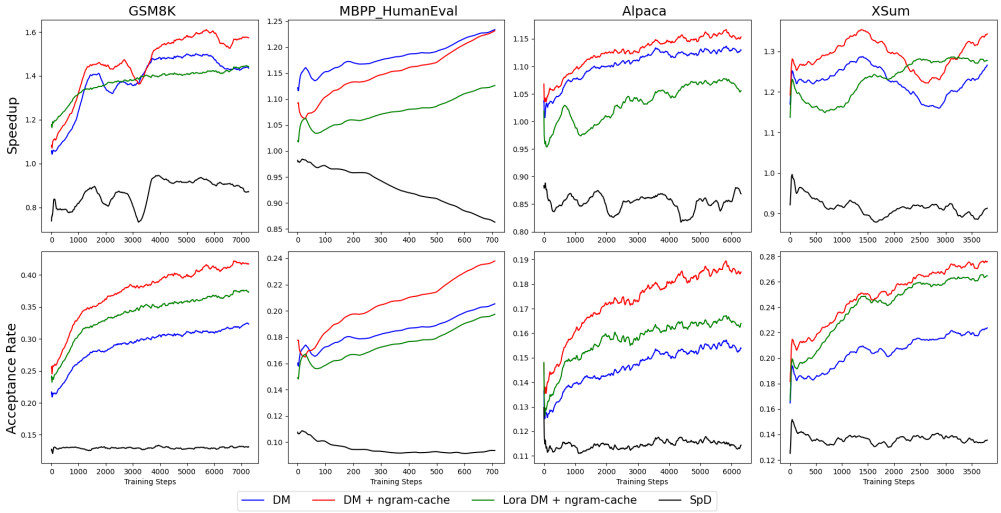

Figure 4: Cross-vocabulary SpD online distillation on Llama-68M with Llama3-8B as target

B.5 Rank ablation for Cross-vocabulary online distillation of LoRA

We also ablate on the rank for the LoRA experiments, specifically focussed on the $\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ + LoRA method. As shown in the Table 12, it is clear that as the rank increases, performance also improves, but beyond rank 32, there is a diminishing return in terms of overall improvement. As such we choose a conservative rank of 32 across all our experiments, tasks and methodologies. Moreover, for the *"one drafter for all"* setting, folding of the LoRA weights is infeasible and as such having a lower rank would benefit the overall memory and compute required for the drafter model on-device.

Table 12: Effect of rank for $\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ + LoRA for Llama-68M and Llama3-8B on GSM8K

Metrics \ LoRA Rank	8	16	32	64	128
Acc Rate	0.30	0.34	0.37	0.38	**0.38**
Speedup	1.478x	1.543x	1.595x	1.625x	**1.632x**

B.6 Distribution Shift

B.6.1 Heterogeneous Dataset Shift

We also ablated on dataset distribution shift and data heterogeneity. We analyzed our training framework by starting the training on the GSM8K dataset and after around 1k steps, shifting to a new data distribution of MBPP+HumanEval Mix. As shown in Figure 5, initially, both speedup and acceptance rate show a steady upward trend, indicating that the model is learning effectively and becoming more efficient. At step 1000, we introduce a new dataset with a different distribution which causes a sharp drop in both acceptance rate and speedup, highlighting the impact of this shift. However, following this disruption, both metrics begin to gradually recover, demonstrating the model's ability to adapt to the new data distribution over time. Additionally, when using our n-gram cache, the recovery seems to be much better and larger when compared to no cache. Finally, when comparing to the model that was solely trained on MBPP+HumanEval mix dataset, the model is able to slowly reach similar performance on both metrics. Overall, using our n-gram cache and despite the distribution shift, the model is showing resilience and adaptability to the new dataset.

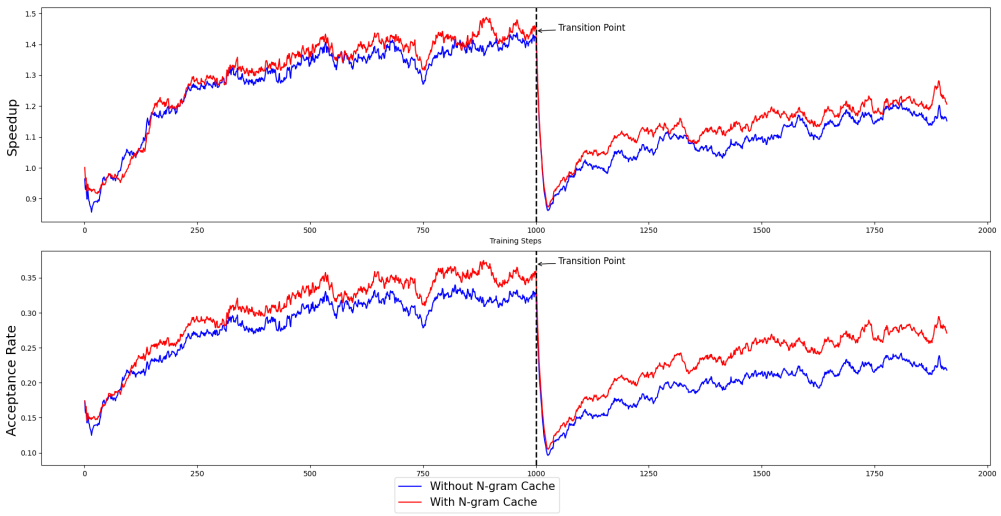

Figure 5: Dataset Drift tracking during training by switching the dataset from GSM8K to MBPP+HumanEval at Step 1000 for Llama-60M vs Qwen-2.5-7B model

B.6.2 Target Model Shift

To further understand the impact of target model drift during training, We conducted empirical analysis on target switching between Qwen2.5 7B and 14B at arbitrary step intervals (e.g., 14B $\rightarrow$ 7B $\rightarrow$ 14B) As shown in Figure 6, our observations indicate that the drafter model is able to re-align seamlessly after each switch. Notably, switching to the 7B model yields reduced speedup due to its lower drafter latency, while maintaining alignment quality. This behavior is largely attributed to the fact that the target models belong to the same model family and as such the drafter is able to continually align during training. We also evaluated the impact of enabling the n-gram cache during target switching. Results show that the drafter recovers alignment quickly when the cache is enabled. However, in scenarios involving cross-family model swaps, a new n-gram cache is preferred due to differences in tokenization. In such cases, a simple distribution shift is insufficient to maintain alignment.

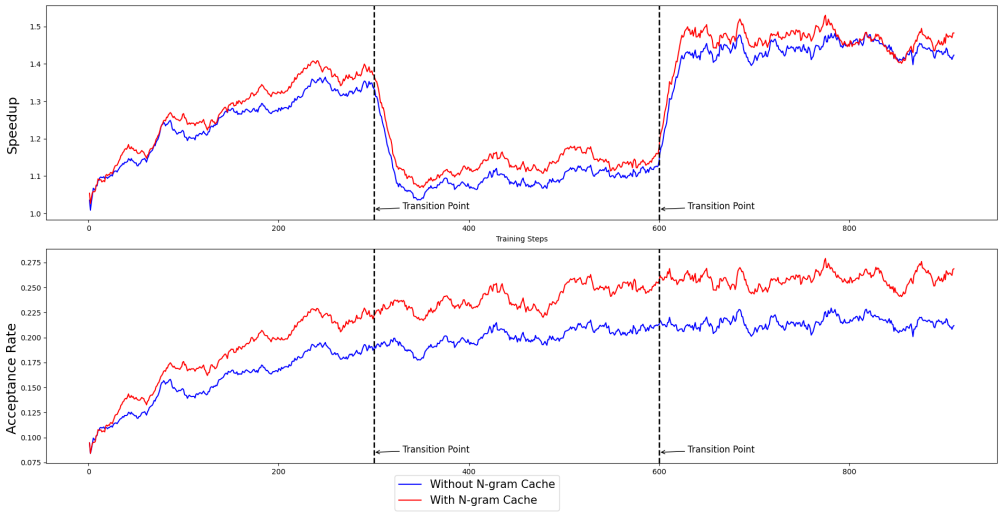

Figure 6: Model Drift tracking during training by switching the target model from 14B to 7B and back to 14B at step 300 and step 600, with Llama-68M as drafter

C Additional Algorithm Details

We show the detailed algorithm for cross-vocabulary distillation in Algorithm 2. The online distillation training leverages our proposed cross-vocabulary speculative decoding in inference or data collection, and uses the hybrid distillation loss to update the draft model.

Algorithm 2 Cross-vocabulary Distillation

Given target $p(\cdot)$, drafter $q_\theta(\cdot)$, online data stream $\mathcal{S}$, data buffer $\mathcal{Q}$, update interval I.
$i, \mathcal{Q} \leftarrow 0, \{\}$
while True **do**
  $x \sim \mathcal{S}, i \leftarrow i + 1$
  Get a response y with cross-vocabulary speculative decoding as in Algorithm 1
  Append (x, y) to $\mathcal{Q}$
  if $i \mod I == 0$ **then**
    Update q_θ on $\mathcal{Q}$ with hybrid distillation loss $\mathcal{L}_{\text{cross\_vocab\_distill}}(\theta)$
    $\mathcal{Q} \leftarrow \{\}$
  end if
end while

We show the detailed algorithms for the two variants of online adaptive drafting training in Algorithm 3 and 4. The distillation loss $\mathcal{L}_{\text{distill}}(\theta)$ uses KL divergence on generated response tokens. The adaptive drafting loss $\mathcal{L}_{\text{adapt}}(\phi)$ is a weighted BCE loss between the acceptance prediction head outputs and acceptance ratio labels constructed from the target model p and current draft model q_θ.

Algorithm 3 Online Adaptive Drafting — Joint Training

Given target $p(\cdot)$, drafter $q_\theta(\cdot)$, acceptance prediction head $f_\phi(\cdot)$, online data stream $\mathcal{S}$, data buffer $\mathcal{Q}$, update interval I.
$i, \mathcal{Q} \leftarrow 0, \{\}$
while True **do**
  $x \sim \mathcal{S}, i \leftarrow i + 1$
  Get a response y with speculative decoding using adaptive drafting
  Push (x, y) to $\mathcal{Q}$
  if $i \mod I == 0$ **then**
    Update q_θ on $\mathcal{Q}$ with distillation loss $\mathcal{L}_{\text{distill}}(\theta)$
    Compute labels $l = \min(1, p(y)/q_\theta(y))$ on $(x, y) \in \mathcal{Q}$
    Update f_ϕ on $\{(x, y, l)\}_{|\mathcal{Q}|}$ with adaptive drafting loss $\mathcal{L}_{\text{adapt}}(\phi)$
    $\mathcal{Q} \leftarrow \{\}$
  end if
end while

D Cross-vocabulary N-gram Cache Ablations

D.1 N-gram distributions

To showcase the captured n-grams during our cross-vocabulary speculative decoding, we collect the n-gram cache across tasks after online training with the hybrid distillation losses. We plot the distribution of the n-gram counts with respect to the frequencies they are encountered in Figure 7. We include n-grams with maximum frequency of 3000 and use a log-scale of the n-gram counts for cleaner visualization. N-grams with higher frequencies over 3000 typically represents a group of delimiters such as spaces and newline characters that are used for formatting. Besides these, we can observe a clear long-tail distribution for more frequent n-grams. Although the majority of n-grams have low frequencies which are not guaranteed to re-appear in future data stream, the long-tail frequency n-grams still have non-trivial contribution and can be utilized to speed up the cross-vocabulary speculative decoding process. We also notice the frequent n-grams constitute a higher percentage in the MBPP+HumanEval domain, despite having smaller dataset size. It indicates

Algorithm 4 Online Adaptive Drafting — Interleaved Training

Given target $p(\cdot)$, drafter $q_\theta(\cdot)$, acceptance prediction head $f_\phi(\cdot)$, online data stream $\mathcal{S}$, distillation data buffer $\mathcal{Q}$, adaptive drafting data buffer $\mathcal{R}$ with max size N, batch size B, update interval I.
$i, \mathcal{Q}, \mathcal{R} \leftarrow 0, \{\}, \{\}$
while True **do**
  $x \sim \mathcal{S}, i \leftarrow i + 1$
  Get a response y with speculative decoding using adaptive drafting
  Push (x, y) to $\mathcal{Q}$
  if $i \mod I == 0$ **then**
    Update q_θ on $\mathcal{Q}$ with distillation loss $\mathcal{L}_{\text{distill}}(\theta)$
    Push $\mathcal{Q}$ to $\mathcal{R}$
    if $|\mathcal{R}| > N$ **then**
      Evict early data from $\mathcal{R}$
    end if
    $\mathcal{Q} \leftarrow \{\}$
  else
    Sample a batch $\mathcal{R}_B \in \mathcal{R}$
    Compute labels $l = \min(1, p(y)/q_\theta(y))$ on $(x, y) \in \mathcal{R}_B$
    Update f_ϕ on $\{(x, y, l)\}_B$ with adaptive drafting loss $\mathcal{L}_{\text{adapt}}(\phi)$
  end if
end while

the n-gram cache technique for cross-vocabulary speculative decoding can be more effective in well-structured domain like coding.

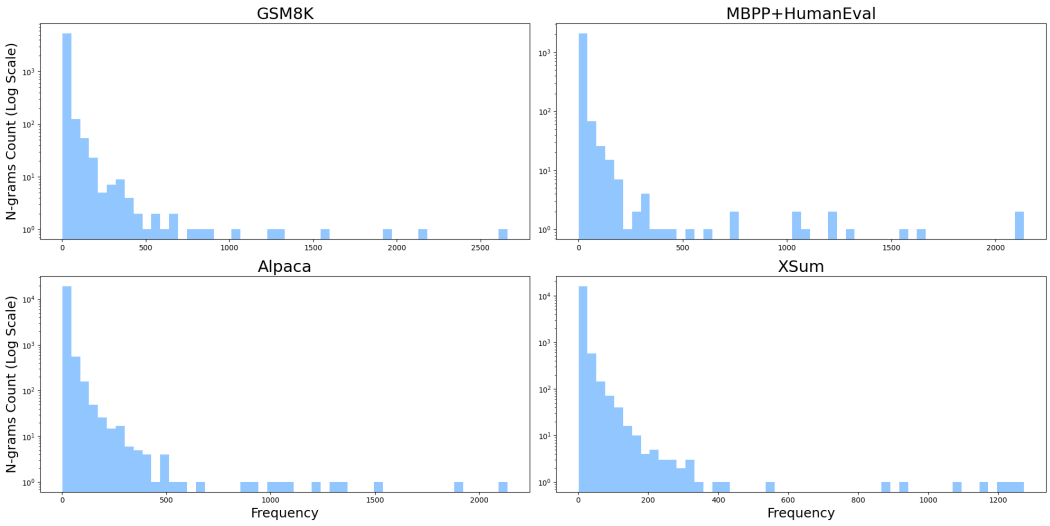

Figure 7: Cross-vocabulary N-gram Distributions of Llama-68M and Qwen2-7B

D.2 N-gram inference examples

Next in Table 13, we show examples of the n-grams being used from the cache during inference in different tasks. The n-gram examples are extracted from actual samples in the datasets with Llama-68M and Qwen2-7B, given a draft length of 4 each proposal or speculative decoding step. Pink tokens represent tokens with direct mapping between the drafter vocabulary and target vocabulary. Yellow tokens represent sub-tokens in the drafter space and the corresponding n-gram tokens in the target space, which are mapped via cache lookup. Lime tokens represent those that are accepted by the target model after verification. Note that even with the mapping, the proposed n-gram tokens are not guaranteed to be accepted and require cross-vocabulary distillation to further align their distributions with the target.

Proposed Tokens (Drafter Vocab)	Mapped Tokens (Target Vocab)	Accepted Tokens
GSM8K		
There , fore , , , there	Therefore , , , there	-
appoint , ments , . , Since	appointments , . , Since	appointments
of , qu , izz , es	of , quizzes	-
distance , tra , ve , led	distance , traveled	distance , traveled
MBPP+HumanEval		
_ , to , _ , sw	_to , _sw	-
asc , ending , order , .	ascending , order , .	ascending , order
= , C , ounter , (	= , Counter , (	= , Counter , (
test , _ , sol , ution	test , _solution	-
Alpaca		
Future , of , A , I	Future , of , AI	-
of , Int , ellig , ent	of , Intelligent	-
most , common , n , oun	most , common , noun	most , common , noun
Imp , act , : , **	Impact , :**	Impact
XSum		
an , off , ender , .	an , offender , .	an , offender , .
is , on , going , .	is , ongoing , .	is , ongoing , .
Ins , pect , or , David	Inspector , David	Inspector
and , thank , ed , the	and , thanked , the	-

Table 13: Examples of N-gram Cache Across Tasks. Each row represents one cross-vocabulary speculative decoding step/proposal extracted from the inference results on the data test set (no sequential order between rows). Drafter proposes 4 tokens each time, which are mapped to the target space either as direct mapping tokens or as merged n-gram tokens. Target model then verifies the mapped tokens to get the final accepted tokens, plus any correction token from the residual distribution or a sampled free token if all mapped tokens are accepted. We also denote no token being accepted as "-".

D.3 N-gram learning

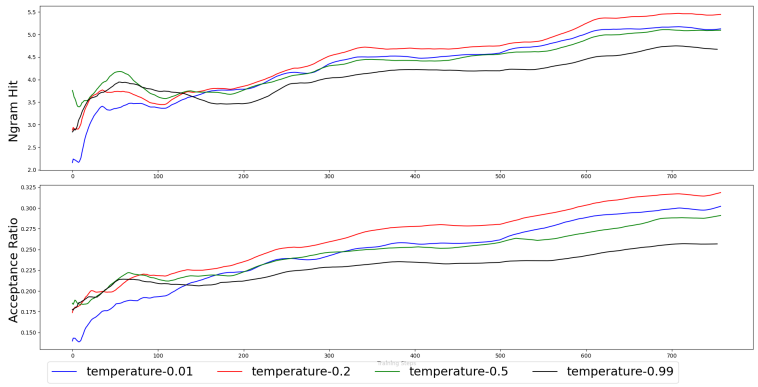

Figure 8: Evolution of n-gram tokens learning with Llama-68M and Llama3-8B on **MBPP+HumanEval** dataset. **N-gram hit** refers to the number of n-grams proposed by drafter in each response, averaged over query samples. **Acceptance ratio** refers to the quantity $\min(1, p/q)$ in speculative decoding, it is averaged over proposal steps with an n-gram token and is indicative of the distribution alignment between the drafter and target. These two metrics show cross-vocabulary distillation learns to slowly align n-gram distributions to the target and utilize them more in inference.

From Figure 8, it can be seen that during the training of $\mathcal{L}_{\text{DM}} + \lambda\mathcal{L}_{\text{N-gram}}$ on **MBPP+HumanEval** dataset, the average acceptance ratio for the n-grams is steadily improving, showing the impact of the loss function for better alignment of the n-grams. Furthermore, the impact is clearly visible even across different temperatures. Similarly as training progresses, the average number of successful cache hit for n-gram tokens per query is also improving.

D.4 N-gram Cache Growth Rate

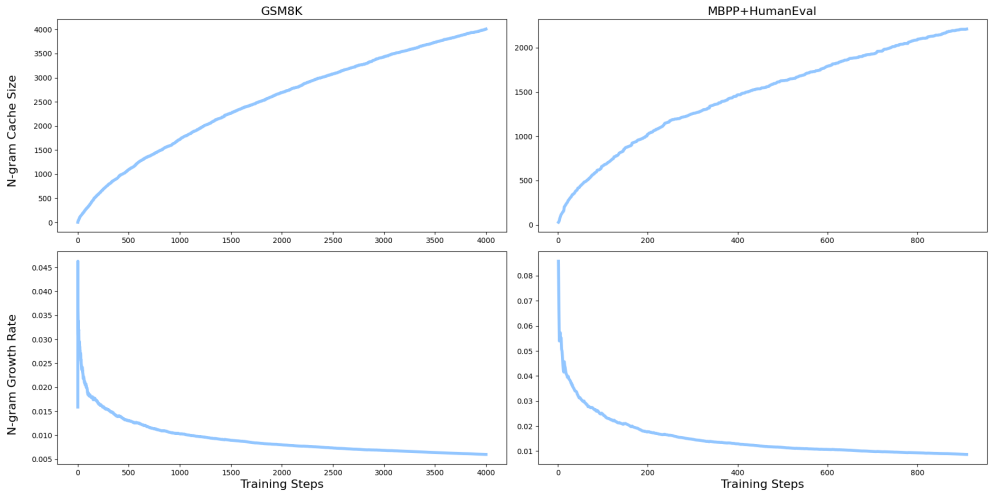

Figure 9: N-gram cache growth rate for Llama-68M drafter and Qwen2-7B target on GSM8K (4K subset) and MBPP+HumanEval datasets.

From Figure 9, We can define the growth rate of the N-gram cache to be growth_rate $=$ cache_size/training_tokens. From the tables, we can observe that the cache grows sub-linearly with number of tokens processed in online learning. The growth rate of the N-gram cache can also be seen to continually decrease.

D.5 N-gram Cache Scalability

Table 14 indicates how speedup is impacted by different max cache size and cache eviction policies: Least Recently Used (LRU) that evicts the n-gram that hasn't been accessed for the longest time, Least Frequently Used (LFU) that removes the n-gram with the lowest access frequency. "1/4" and "1/2" refer to using max cache size as a quarter and half of the full final cache size that we previously trained.

Among all eviction policies and max cache sizes, both evaluation metrics are better than using no cache. Specifically LRU shows diminishing returns over the growth of the cache. Meanwhile LFU plateaus at an earlier stage, with performance matching the full cache even with a quarter of the size. This could be due to LFU capturing most of the high frequency and important n-grams, compared to LRU which only captures the recent n-grams. In the context of on-device deployment, this implies with proper selection of cache size and eviction strategy, we can potentially attain robust performance given only constrained memory resources, highlighting the practical feasibility of our approach.

Table 14: Cache size scaling on MBPP+HumanEval

Cache Size	Cache Type	Acc Rate	Speedup
No Cache	-	0.219	1.29x
1/4	LRU	0.254	1.33x
	LFU	0.259	1.36x
1/2	LRU	0.261	1.35x
	LFU	0.263	1.36x
Full Cache	-	0.267	1.36x