Context-Aware Assistant Selection for Improved Inference Acceleration with Large Language Models

Jerry Huang[*]
Mila - Quebec AI Institute
Université de Montréal

Prasanna Parthasarathi
Huawei Noah's Ark Lab

Mehdi Rezagholizadeh
Huawei Noah's Ark Lab

Sarath Chandar
Mila - Quebec AI Institute
Polytechnique Montréal
Canada CIFAR AI Chair

Abstract

Despite their widespread adoption, large language models (LLMs) remain prohibitive to use under resource constraints, with their ever growing sizes only increasing the barrier for use. One noted issue is the high latency associated with auto-regressive generation, rendering large LLMs use dependent on advanced computing infrastructure. Assisted decoding, where a **smaller** *draft* model guides a **larger** *target* model's generation, has helped alleviate this, but remains dependent on alignment between the two models. Thus if the draft model is insufficiently capable on some domain relative to the target model, performance can degrade. Alternatively, one can leverage multiple draft models to better cover the expertise of the target, but when multiple black-box draft models are available, selecting an assistant without details about its construction can be difficult. To better understand this decision making problem, we observe it as a contextual bandit, where a policy must choose a draft model based on a context. We show that even without prior knowledge of the draft models, creating an offline dataset from only outputs of independent draft/target models and training a policy over the alignment of these outputs can accelerate performance on multiple domains provided the candidates are effective. Further results show this to hold on various settings with multiple assisted decoding candidates, highlighting its flexibility and the advantageous role that such decision making can play.

1 Introduction

With the introduction of the Transformer (Vaswani et al., 2017) has emerged the era of large language models (Chowdhery et al., 2022; Touvron et al., 2023; OpenAI, 2024) and the development of LLMs capable of reasoning and acting in astonishingly human-like manner (Kaplan et al., 2020; Wei et al., 2023; Ouyang et al., 2022). However, the use of resource intensive models and techniques remains a pre-requisite and accordingly, methods have been developed and applied to alleviate concerns relating to the practical usability of these models (Dettmers et al., 2022; Dao, 2024). One major area that has observed consistent improvement over time is the *auto-regressive decoding* aspect of text generation, where each generation of a new token requires a complete inference pass through the model, which under-utilizes the property of attention and the ability of modern accelerators (e.g. GPUs, TPUs) to parallelize computations (de Jong et al., 2022; Kim et al., 2023a).

A growing approach towards addressing this is speculative decoding (Xia et al., 2023; Leviathan et al., 2023). In speculative decoding, latency is reduced by minimizing the amount of high-latency sequential computations and replacing them with cheaper ones. Rather than sampling directly from the larger model, the sampling is approximated with samples from a smaller and cheaper model through *accept-reject* sampling. Specifically, a small draft model auto-regressively generates text which is then verified by a larger target model in parallel (Stern et al., 2018; Sun et al., 2021). Thus the large model does not need to generate text repeatedly but rather guides the small model by correcting outputs when it is truly incapable. This can reduce the number of calls to the large LLM, saving both time and memory. However two models are required, along with some similarity in their generative abilities in order for this method to see signficant speedups. While approaches exist to circumvent some of these needs (Yang et al., 2023; Zhang et al., 2023; Li et al., 2024; Cai et al., 2024; Hooper et al., 2024), these are often limited by the need for additional tuning (Liu et al., 2024b; Cai et al., 2024; Li et al., 2024), which is difficult in resource constrained settings, or quality degradation in generations (Kim et al., 2023b). Because of

[*]Correspondance to jerry.huang@mila.quebec

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 5817–5830
November 12-16, 2024

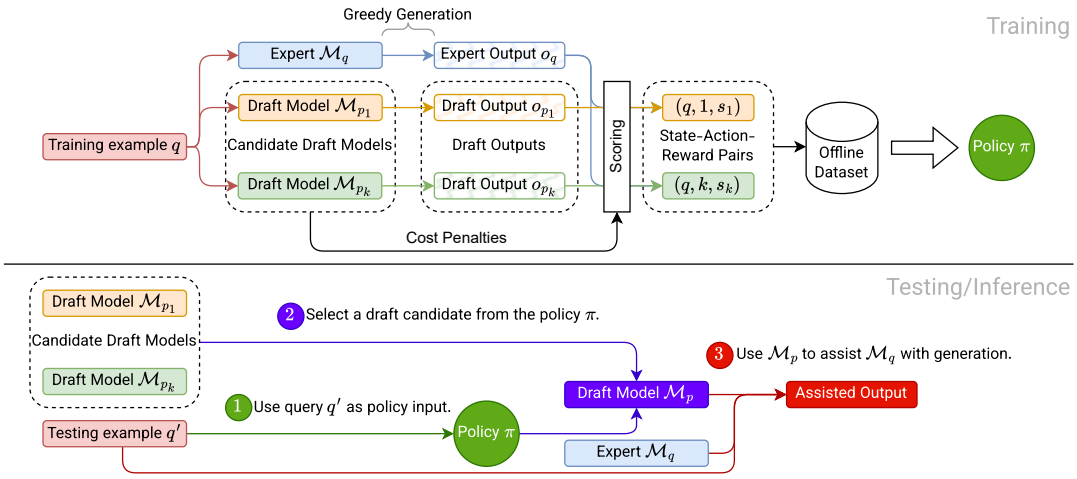

Figure 1: Overview of our methodology. We first train a policy using offline data collected from greedily decoded output from each model, which are scored to produce reward samples. At test time, the policy takes in a query q' to select a draft candidate model, which is then used for assisted generation with the target model.

the evident size-cost tradeoff, this is very efficient *if the draft model is well aligned to the target*.

However, while one can ensure that the final output follows the target distribution (Chen et al., 2023), selecting an inadequate draft model can lead to a lack of acceleration due to the signficant number of rejections that will occur. While other methods allow for changes in the output distribution shift (Kim et al., 2023a; Zhou et al., 2024; Fu et al., 2024) to further speed-up inference, such types of shifts can be problematic in many high-risk scenarios. From this perspective, the presence of multiple draft models, each suited for different settings, can be helpful for inference acceleration without degredations in generation quality. By dynamically choosing a draft model, speedups can be achieved on multiple domains with marginal additional costs. However this requires learning how to choose the best draft option given a context, introducing a decision making problem which needs to be solved.

So how can this decision making process be learned? We start by observing this as a contextual bandits (Woodroofe, 1979; Auer, 2003), where the goal is to have a policy select a draft model based on a given query. This also requires rewards from which the policy can learn to estimate and compare the ideality of different actions that can be taken. To this end, we use an offline process to estimate the contextual alignment between different draft models and the target model on a set of training examples (Figure 1), enabling the construction of a dataset that defines the preference a target can have towards specific draft candidates. This enables us to train a policy that can take into account such preferences without knowing further details about the draft models. By deloying this policy at inference time it becomes possible to weigh these preferences, leading to speedups in generation with the target. We further show that this policy is useful when using self-speculative decoding, whereby the draft model is a subset of the target model parameters. To summarize our contributions:

- We frame a speculative decoding scenario as a contextual bandits problem, where multiple draft models serve as arms that each produce a reward, an abstraction of the inference speed-up relative to using the target model on its own which is not known a priori.
- We demonstrate that offline training of a decision making agent through only the similarity between the draft and target model generations, the agent can correctly select which draft model to use for a given input query.
- We show that the policy can balance tradeoffs in draft model alignment and generation speed by incorporating explicit information about the model within the reward.

2 Methodology

2.1 Motivation

Assume a large target model, $\mathcal{M}_e$, incurs large end-to-end latencies that one wants to avoid. Speculative decoding aims to solve the latency issue by using a draft model to approximate the target model. However, as previously discussed, the draft model must be similar to the target model otherwise the

sampling distribution is too different and produce no speedups. Therefore, while draft models can help, they are only reliable when their knowledge distribution resembles that of the target. Accordingly, using only one draft model may not serve well in general if the target has multiple expertises. But by dynamically choosing between different draft models in any given scenario, then benefits from each draft model can be observed as long as the decision maker is competent and efficient.

2.2 Problem Formulation

When presented with a query q, selecting a draft model among multiple unique candidates can lead to varying performance based on the chosen option.

From a contextual bandits lens, q is a context for which there are k arms that each returns an independent reward r. Each of arm corresponds to a different drafter whose reward is the time it takes to generate the output sequence through speculative decoding. Accordingly, each arm can produce a different reward for each q. The objective then consists of learning a policy $\pi(\cdot|q)$ which, for any given context q, can select among the arm which can produce the greatest reward. From a speculative decoding scenario, the goal is to select the draft model whose abilities best align with the target for that given query, as this will minimize the number of times the target model must be invoked.

Randomly choosing a draft model risks significant increases in latency, therefore learning to make the correct decision in a sample efficient manner is important. While the ideal reward is the real/observed speed-up, this can be expensive if the aligment with draft models is unknown. As such, a cheaper proxy may be necessary. However, two factors have a direct effect on the true reward: 1) the alignment between target and drafter and 2) the size of the drafter. This provides an alternative way to collect policy training data: use the draft models auto-regressively and compute alignment scores with the target outputs, then adjust these based on the size of the drafter. Next, we describe how we collect our data to train a policy offline.

2.3 Offline Data Collection

Given a set of queries $\mathcal{Q} = \{q_i\}_{i=1}^n$, we produce outputs based on each q_i for the target model, o_i^e, as well as each of the candidate drafters, $\{o_i^j\}_{j=1}^k$. We then use a similarity metric to compute scores for each candidate output

$$s_i^j = f(o_i^e, o_i^j) \tag{1}$$

as a way to measure the alignment between target and candidates for q_i. It is further possible to incorporate a score for the inference speed. For example, if we consider some relative measure of the inference speed for the specific drafter to be c_i^j, then one can adjust the score as a weighted sum

$$s_i^j = \alpha \cdot f(o_i^e, o_i^j) + (1-\alpha) \cdot c_i^j \tag{2}$$

which takes into account both factors where $\alpha \in [0, 1]$ weighs the two components.

2.4 Decision Making

With the offline dataset, it becomes possible to train a policy π which can independently act on a context by choosing a drafter to use with the target. We consider each (q_i, j, s_i^j) as state-action-reward tuples used to train π.

Within the contextual bandits reformulation, each query-action pair $(q_t, a_t) \in \mathcal{Q} \times \mathcal{A}(q_t)$ is the drafter which produced an observed reward $r(q_t, a_t)$. Here, we use the score s_i^j directly as the reward, as it acts as an estimate for the effectiveness of drafter j on the context. The policy is represented by a mapping $\pi_\theta(a|q)$ from $\mathcal{Q} \times \mathcal{A}$ to $\mathbb{R}$ and we want to find parameters θ^* that maximize

$$J^\pi = \mathbb{E}_{q\sim P_q(\cdot), a\sim\pi(\cdot|q)}[r(q,a)]$$

where P_q is the sampling distribution of the context. As the action space is discrete, integrating over the action space is equivalent to a

$$\int_a \pi_\theta(a|q)da = \sum_{a\in\mathcal{A}(q)} \pi_\theta(a|q) = 1$$

and the gradient with respect to the policy is

$$\nabla_\theta J^{\pi_\theta} = \mathop{\mathbb{E}}_{q\sim P_q(\cdot), a\sim\pi_\theta(\cdot|q)}[\nabla \log \pi_\theta(a|q) r(q,a)]$$

which is equivalent to the REINFORCE (Williams, 2004) policy gradients method and we therefore use it to train our policy.

3 Experimental Results

3.1 Experimental Setup

Models and Tasks. We select publicly available LLMs to use for our experiments. We conduct a number of experiments, which we motivate by

Draft Model	IWSTL2017 En-De			XSum			Average
	Bleu	Decoding Speedup	Accept (%)	Rouge-L	Decoding Speedup	Accept (%)	Speedup
Auto-Regressive Generation							
Greedy	18.26	1.00× (31.20$_{\pm 0.04}$ ms/token)	-	35.93	1.00× (36.92$_{\pm 0.08}$ ms/token)	-	1.00×
Sampling	9.85	1.00× (31.06$_{\pm 0.06}$ ms/token)	-	29.83	1.00× (37.06$_{\pm 0.06}$ ms/token)	-	1.00×
Greedy Assisted Decoding							
`T5-Small`	18.26	1.10× (28.24$_{\pm 0.15}$ ms/token)	41.68	35.93	0.97× (38.60$_{\pm 0.18}$ ms/token)	28.21	1.03×
`T5-Small-XSum`	18.26	0.83× (37.61$_{\pm 0.19}$ ms/token)	7.23	35.93	1.21× (30.71$_{\pm 0.15}$ ms/token)	40.24	1.02×
Speculative Decoding							
`T5-Small`	9.24	1.10× (28.14$_{\pm 0.17}$ ms/token)	38.21	29.10	0.99× (37.53$_{\pm 0.14}$ ms/token)	26.02	1.04×
`T5-Small-XSum`	9.51	0.83× (37.45$_{\pm 0.18}$ ms/token)	8.75	29.10	1.18× (31.33$_{\pm 0.16}$ ms/token)	38.29	1.01×
Speculative Decoding + Decision Making							
Greedy π_θ	9.76	1.09× (28.56$_{\pm 0.16}$ ms/token)	37.72	29.83	1.17× (31.63$_{\pm 0.19}$ ms/token)	37.76	1.13×
Dynamic π_θ	9.62	1.07× (29.20$_{\pm 0.17}$ ms/token)	36.64	29.45	1.16× (31.88$_{\pm 0.17}$ms/token)	37.34	1.11×

Table 1: Quality, decoding speeds and acceptance rates when using a policy for selecting between different draft models of the same size but specialized on different domains. Across the two domains, both a greedy and dynamic policy can accelerate decoding on both domains, with marginal differences between the exact nature of the policy. Acceptance rate computation is described in Appendix A.

varying the draft options along different axes such as alignment with the target model, sizes of the draft models, architecture of the drafter/target and the level of independence between the draft and target models. Each of these forms a dedicated experiment detailed in the sections that follow.

Data Collection. For each experiment, we collect offline data using task-specific training dataset splits. Each model is used to generate a greedy decoded sample from each, which is used to construct a reward dataset. To score samples against the target model output, we use the Rouge-L score.

Policy Training. To train our policy, the input is a sentence embedding of the query from the target model and the output is a distribution over the drafting candidates. We train on the offline dataset for 3 epochs using a fixed batch size of 64 and AdamW (Loshchilov and Hutter, 2019) with a learning rate of 1e-3 and weight decay 1e-2. All other hyperparameters are set to their default values in PyTorch. In all experiments, our policy consists of a 3 layer multi-layer perceptron. Hidden layers have a fixed dimensions of 512 with a `tanh` activation function. The input dimension is the hidden dimension size of the target model and the output size is the number of drafting options.

Inference. We sample using a temperature T of 1 and draft tokens γ set at 7. We use both a policy that takes the greedy action and another that samples from the output distribution.[1] The policy takes in a sentence embedding of the query and returns a distribution, from which a drafter is sampled and used to assist decoding for the specific query.

3.2 Results

Learning to choose the draft model. For our first experiment, we use a T5 (Raffel et al., 2020) encoder-decoder models. As the target, we use an instruction-finetuned (Wei et al., 2022) `Flan-T5-XXL` (Chung et al., 2022) while our draft candidates are publicly available `T5-Small` models, one the base version and another fine-tuned on text summarization[2]. We evaluate on translation (IWSLT2017 En-De (Cettolo et al., 2017)) and text summarization (XSum (Narayan et al., 2018)).

Table 1 compares when draft models of the same size vary in their domain of expertise. While each model accelerates generation non-trivially within their knowledge domain (En-De for `T5-Small` and XSum for `T5-Small-XSum`), they are largely unhelpful or detrimental when used outside their domain of expertise, as seen with the 1% slowdown from `T5-Small` on XSum and a 17% decrease using `T5-Small-XSum` on En-De. In comparison, the policy ensures acceleration within both domains with neglibile latency from decision making.

This highlights some immediate benefits of policy use, namely that it can identify the correct draft model for a context without any explicit information regarding the draft candidates themselves. Rather, generating sampling outputs from each draft model and the target individually is sufficient to develop a general ability to differentiate between domains through the use of the computed rewards.

[1] Some ablations are presented in Appendix B.

[2] Publicly available checkpoint from (Kim et al., 2023b).

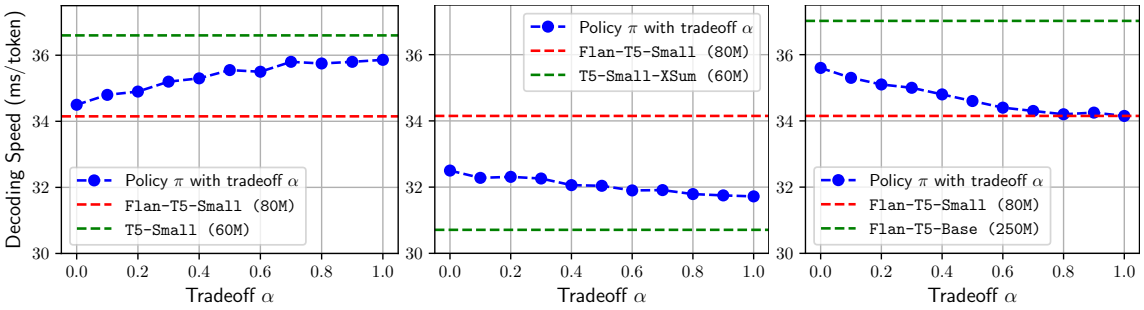

Figure 2: Effect of varying the tradeoff between output alignment and draft model size (controlled through α). Each compares the use of `Flan-T5-Small` as a draft model (red horizontal line). As α increases, the model increasingly uses the smallest draft model for decoding, demonstrating that the offline dataset is sufficient to learn how to balance the quality of the draft model's outputs and the cost of using it. All cases use speculative sampling/decoding.

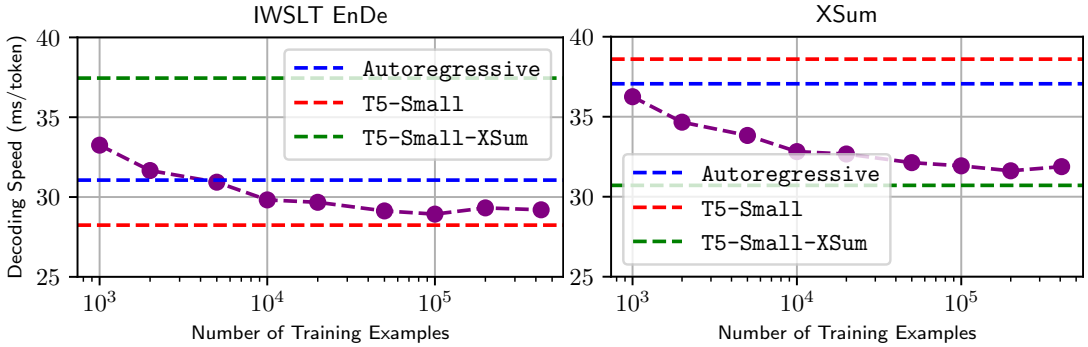

Figure 3: Decoding speed using a dynamic policy as a function of the number of examples used to train the policy, tested on IWSLT2017 EN-DE (left) and XSUM (right). The number of examples is marked in log-scale. Horizontal lines denote decoding speeds of individual drafting options.

DRAFT MODEL	**Decoding Speed**
Auto-regressive	$1.00\times$ ($37.06_{\pm 0.16}$ ms/token)
`T5-Small`	$0.97\times$ ($38.60_{\pm 0.18}$ ms/token)
`T5-Small-XSum`	$1.21\times$ ($30.71_{\pm 0.15}$ ms/token)
`Flan-T5-Small`	$1.09\times$ ($34.15_{\pm 0.17}$ ms/token)
`Flan-T5-Base`	$1.00\times$ ($37.02_{\pm 0.18}$ ms/token)

Table 2: Speeds of different draft models on XSUM with a `Flan-T5-XXL` model expert (averaged over 5 seeds). Observed decoding speed varies as an effect of drafter size and alignment with the expert.

Balancing quality and speed. It is also important that the draft model is sufficiently inexpensive to use relative to the target model. This motivates our second experiment, which is evaluated only on XSUM, but compares draft candidates that vary in terms of size and target model alignment. Multiple draft models are compared: a `Flan-T5-Small` (80M parameters), the same `T5-Small` (60M) models mentioned above, and `Flan-T5-Base` (220M). Table 2 shows the speed-ups earned through speculative decoding using these different draft candidates and a `Flan-T5-XXL` target. Although larger draft models may be better aligned with the target compared to smaller options, using them incurs a latency that can end up being less efficient.

Balancing alignment and efficiency is therefore an issue to consider when deciding between candidates. Figure 2 shows how offline training can help accomplish this. Setting α to vary between the objectives defined in §2.3, where we use fixed inference costs based on the size of the draft models, we observe how a dynamic policy can eventually adapt to the preferences set by the choice of α. For example, as α approaches 1, the policy places increasing preference on the smallest draft model regardless of quality. Meanwhile $\alpha \to 0$ shows increasing preference towards the model that has greatest alignment with the target generations.

This demonstrates the general flexibility that can come with using such a weighting scheme of different rewards, while demonstrating that even simpler proxies for the inference penalty are sufficient to properly balance the two.

How many examples need to be learned to differentiate? It is further necessary to consider

the number of examples that are needed for the decision maker to properly learn to differentiate between different examples. To this end, we investigate how quickly the policy can learn to use the annotated scores within the offline dataset to demonstrate a visible speed-up improvement. We re-use our models from the first experiment, but keep track of the decoding speed as the number of examples used to train our policy π_θ increases.

As we can observe in Figure 3, learning to select the correct model occurs rather quickly, as training for fewer than a total of 10000 examples is sufficient to attain a level of performance that is equivalent to training on the entire offline dataset, which consists of nearly 400 thousand examples. This result demonstrates the general efficiency of this method, as collecting and training the policy on outputs from a minimal amount of examples shows the ability to generalize quite strongly.

Auto-regressive generation as an option. Scenarios exist where the draft models will not be useful, in which case using the target auto-regressively remains the most reasonable option.

To this end, we attempt to observe how providing this option to the decision maker can affect our previous experiments. We repeat the same experiment from Table 1 but allow our policy to learn to choose to generate auto-regressively. To avoid trivially perfect matching of outputs, we sample outputs from the target model and score against the greedy output. Due to the large size of the target compared to the drafters, we use $\alpha = 0.5$ to balance the size and quality scores.

EN-DE	BLEU	Decoding Speed
Greedy π_θ	9.82	1.09× ($28.95_{\pm 0.15}$ ms/token)
Dynamic π_θ	9.76	1.07× ($29.34_{\pm 0.13}$ ms/token)
XSUM	**ROUGE-L**	**Decoding Speed**
Greedy π_θ	30.02	1.17× ($31.64_{\pm 0.17}$ ms/token)
Dynamic π_θ	29.53	1.16× ($31.73_{\pm 0.18}$ ms/token)

Table 3: Decoding speeds under a dynamic decision making regime where auto-regressive generation is a decoding option, on IWSLT2017 EN-DE and XSUM.

Table 3 shows that adding the auto-regressive option does not degrade decoding speed or generation quality and in fact may accelerate overall decoding. This may be because some contexts are out-of-distribution for all candidate draft model; in these cases, auto-regressive generation can be much more efficient by a signficant margin.

DRAFT CANDIDATE	ACCURACY	Decoding Speed
Auto-Regressive	10.67	1.00× ($22.03_{\pm 0.16}$ ms/token)
`T5-Small`	10.99	0.78× ($28.08_{\pm 0.18}$ ms/token)
`T5-Small-XSum`	10.09	0.74× ($29.91_{\pm 0.16}$ ms/token)
Greedy π_θ	10.64	0.95× ($23.54_{\pm 0.19}$ ms/token)
Dynamic π_θ	10.38	0.95× ($23.45_{\pm 0.18}$ ms/token)

Table 4: Decoding speeds on GSM8K (test set) with a `Flan-T5-XXL` expert. Inference on the latter two tasks is negligibly different from Table 3.

We further verify whether the policy can ignore draft models when they are not useful. We experiment by including GSM8K to our tasks, which only the target is aligned. Since neither draft model can accelerate inference on this task, the policy should ideally avoid drafting for examples from this setting. Since GSM8K is significantly smaller than EN-DE and XSUM, we reduce the number of examples to match all datasets in terms of size.

Table 4 shows auto-regressive generation to (unsurprisingly) outperform assisted generation. However, using a policy shows comparable speed to auto-regressive generation, indicating that it learns to ignore the draft models due to the stark contrast in the greedy outputs from each model.

Generalization to Multi-Task Drafters. To demonstrate the applicability of this method to more general settings, in particular cases where the draft models may be competent at multiple tasks, we further apply our policy-based selection method to SpecBench (Xia et al., 2024) using a `Vicuna-33B` (Chiang et al., 2023) target with smaller draft models (Table 5). Given the size of SpecBench (480 examples, divided equally into 6 tasks), we use this exclusively as a test-set. To train our policy, we use the original task datasets from which SpecBench examples were extracted and sample even amounts of examples from each (2000). For MT-BENCH, there are only 80 total examples which are all included in the test set but which we sample with replacement to use for a training set. Accordingly, results on this task may be over-confident. Because `Vicuna` models are decoder-only Transformers, we adjust the sentence representation to be the final hidden representation of the input sequence. Our results show that our initial findings from a `T5` architecture hold, suggesting that such a policy-based training method is both robust and generalizable to different settings.

Ablation with self-drafting. Despite the benefits of assisted decoding, drafting relies on the avail-

Method	MT-BENCH	TRANS.	SUM.	QA	MATH	RAG	AVG.
Auto-regressive	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
`Vicuna-68m`	1.74×	1.28×	1.74×	1.54×	1.70×	1.63×	1.60×
`Vicuna-160m`	1.52×	1.10×	1.51×	1.36×	1.50×	1.44×	1.40×
`LLaMA-68m`	1.40×	1.32×	1.41×	1.41×	1.48×	1.49×	1.41×
Policy π_θ + `Vicuna-68m/160m`	1.74×	1.28×	1.73×	1.54×	1.70×	1.62×	1.59×
Policy π_θ + `Vicuna-68m/LLaMA-68m`	1.72×	1.31×	1.73×	1.53×	1.69×	1.63×	1.60×
Policy π_θ + `Vicuna-160m` and `LLaMA-68m`	1.51×	1.31×	1.49×	1.39×	1.49×	1.47×	1.41×

Table 5: Acceleration on SpecBench using a `Vicuna-33B` expert with `Vicuna-68m`, `Vicuna-160m` and `LLaMA-68m` as draft candidates. We use a greedy policy to select a drafter for a given query.

DRAFT CANDIDATE	ALPACA	TRIVIAQA
Auto-Regressive Generation		
Auto-regressive	1.00× ($26.43_{\pm 0.14}$ ms/token)	1.00× ($6.12_{\pm 0.12}$ ms/token)
Intermediate Drafting		
Layer 8 Drafting	0.77× ($34.24_{\pm 0.15}$ ms/token)	0.77× ($7.97_{\pm 0.14}$ ms/token)
Layer 16 Drafting	0.81× ($32.51_{\pm 0.13}$ ms/token)	0.69× ($8.87_{\pm 0.15}$ ms/token)
Layer 24 Drafting	0.73× ($36.07_{\pm 0.16}$ ms/token)	0.69× ($8.93_{\pm 0.13}$ ms/token)
Layer 32 Drafting	0.50× ($52.82_{\pm 0.19}$ ms/token)	0.60× ($10.21_{\pm 0.13}$ ms/token)
Intermediate Drafting + Policy		
Greedy Policy π_θ	0.80× ($32.85_{\pm 0.16}$ ms/token)	0.66× ($9.41_{\pm 0.14}$ ms/token)
Dynamic Policy π_θ	0.82× ($32.17_{\pm 0.15}$ ms/token)	0.66× ($9.34_{\pm 0.11}$ ms/token)
Intermediate Drafting + Policy + Auto-regressive Option		
Greedy Policy π_θ	0.99× ($26.75_{\pm 0.18}$ ms/token)	0.96× ($6.47_{\pm 0.12}$ ms/token)
Dynamic Policy π_θ	0.95× ($27.58_{\pm 0.19}$ ms/token)	0.91× ($6.78_{\pm 0.13}$ ms/token)

Table 6: Results in a scenario for deciding when to early exit. Even using a self-drafting model with multiple early exists, the policy is capable of maintaining performance by choosing an appropriate action.

ability of small draft models that

(1) Share a vocabulary with the target.

(2) Align with the target on the tasks of interest.

Such models can be difficult to obtain, leading to *self-drafting* (Yang et al., 2023; Li et al., 2024; Hooper et al., 2024), where the draft model exists within the target. To explore the differences with this setting, we conduct an additional ablation.

We use a `LLaMA-2-13B-Chat` model (Touvron et al., 2023) using early exits, following (Kavehzadeh et al., 2024) with the use of a single language modeling head for all exits. While other methods exist, these can possess a combinatorial number of potential draft options and necessitate pre-determined path flows during inference (Zhang et al., 2023). Meanwhile, methods that use additional language modeling heads for parallel decoding require additional parameters which can both become irreconcilable with resource constraints or degrade generation quality (Cai et al., 2024).

Results on ALPACA and TRIVIAQA, conducted on each dataset independently, under this setup (Table 6) show that although intermediate layer drafting results in a decrease in decoding speed, using a policy can minimize performance loss in particular with the presence of an auto-regressive option, highlighting that the proposed offline policy learning approach has potential for self-drafting as well. This demonstrates the use of a policy remains a useful manner to fall back to the most effective decoding options. Furthermore, when considering the case where the auto-regressive option is not available, we note that the policy methods are capable of recovering to a performance similar to the best case intermediate drafter on ALPACA. While this is not the case with TRIVIAQA, this is perhaps attributed to the short answers within this dataset.

4 Discussion

LLM Routing. LLMs have demonstrated remarkable capabilities across a range of tasks, but there exists wide variation in their costs and capabilities. Very broadly, more capable models tend to be more expensive than less capable models. This leads to a dilemma when deploying LLMs in the real-world - routing all queries to the largest, most capable model leads to the highest-quality responses but can be expensive, while routing queries to smaller models can save costs but may result in lower-quality responses. Similarly, not all models may be well suited for the same set of tasks, meaning that routing to the most suitable model can be of great importance as well.

Our work shares a great deal of similarity with this notion of model routing, or selecting the best model based on the query. In particular, the set of draft models can be considered to be a group of sub-networks, similar to a Mixture-of-Experts (MoE) (Shazeer et al., 2017) style paradigm. The policy meanwhile acts as a router to the correct sub-network. More advanced routing techniques (Fedus et al., 2021; Ong et al., 2024) have been explored as a way to leverage the multitude of LLMs that exist

in the wild, but have yet to be widely used within downstream settings such as speculative decoding.

Adaptive Speculative Decoding. Speculative decoding methods require the use of many predefined hyper-parameters which can signficantly influence acceleration, with even minor changes having noticable effects. Recent work has begun to explore how to decouple this process, such as by dynamically selecting the number of drafting tokens to generate at each decoding step (Wang et al., 2024; Liu et al., 2024a). Kavehzadeh et al. (2024) further discussed dynamically selecting a model per instance, however their method is limited to their specific setup due to needing to compute confidence scores after generation at early exits.

While we do not introduce a new decoding algorithm, we make a first attempt to make the speculative decoding adaptive through the ability to switch between multiple draft models based on the input. However, more complex levels of adaptivity may be necessary as each decoding step may not be the same, necessitating perhaps a need to carefully adjust different hyperparameters through the process in order to maximize acceleration.

Decision Making for Assisted Decoding. Assisted decoding can require making multiple decisions. One of these is determining an ideal number of draft tokens to decode at each step. Another relates to how to reject tokens, which commonly uses either greedy (Xia et al., 2023) or sampling-based token-matching heuristics (Leviathan et al., 2023). However, there are trade-offs when enforcing specific choices, which requires further investigation to better understand how to tune such techniques.

This work proposes adding an additional decision at the beginning of the decoding process, namely at the beginning of the process under the assumption that multiple drafting options exist. While we limit ourselves to make a more complete analysis within a more self-contained setting, various ways to have these methods co-exist within one larger pipeline are possible. However such work is left for future exploration due to the non-trivial nature of understanding how different choices and effect overall reported results in conjunction.

Measuring Alignment Between Outputs. We observe that token-level similarity scores are effective for training the decision maker, which can be attributed to the fact that assisted decoding itself relies on matching the token-level distribution of outputs. As such, if the greedy-decoded output from a draft model highly resembles the target output, it follows that this will be represented by a higher degree of similarity between the probabilty distributions in the logit space, which can then lead to fewer rejections when sampling.

However, such metrics have limitations (Deutsch et al., 2022) due to capturing primarily superficial elements of text, where marginal differences in distribution have large effects on the output text. Furthermore, different metrics may overfit specific tasks, necessitating the need for better measures of draft/target alignment, which can hopefully lead to better estimation of rewards for training improved policies, either by desigining better metrics themselves or by learning to compare features at different levels of granularity (ex. target and draft logits against text outputs). Additionally, semantic meaning also can play an important role, as outputs with signficant structure may still possess the same meaning, something that token-level similarity metrics will not adequately capture.

Speculative Decoding as Approximate Inference. Speculative decoding can be analogized as a form of approximate inference where due to the intractability of performing inference with a model of interest, approximation methods are used to learn an estimate of the model. While training the draft model is equivalent to performing variational inference (i.e. approximating an intractable distribution with a surrogate), this can be expensive. Accordingly, training only a policy can be seen as weighing a set of fixed distributions to act as a better surrogate for the target model.

Some works have further attempted to study speculative decoding from this angle. In particular, Zhou et al. (2024) explore such a process by building a draft model through the use of KL-divergence losses, effectively building a posterior distribution of the target model based on likelihood information from the draft output. Liu et al. (2024b) meanwhile explore the same technique as the distribution of examples changes, building a draft model that can adapt to changing user inputs. Such settings also could perhaps benefit from multiple draft models, where conditioning on the query can enable more effective adaptation of draft models to better generalize to unseen settings.

Hosting Multiple Draft Models. An important aspect of this method relates to the need to host multiple draft models in conjunction with the expert.

This can incurr additional costs, in particular if the expert and selected drafter do not reside in the same device. While methods such as self-drafting avoid this issue and the possibility to create minimally-sized drafters generally alleviates the concern of excessive memory usage, one particular aspect of consideration remains hardware level optimizations which can best enable for the selected drafters to be loaded at maximal speed, avoiding additional latency that can result from the bandwidth constraints that relate to data transfer between devices.

5 Conclusion

This work presents the first work at attempting to integrate assisted generation within a setting where multiple black-box draft candidates exist. When no a-priori knowledge of which draft candidate is best suited for assisting the decoding of a given example, the problem can be modeled as a contextual bandits problem, where the goal is to estimate the unknown reward from each drafting option. Our work demonstrates that offline RL presents an efficient method for learning to distinguish the available options and provide accelerated decoding across various examples within this setting, with a logical way to collect offline data from models for learning. Our results and ablations show that learning a policy with this approach can adapt to general preferences while accounting for more complex aspects of the decision making, highlighting its robustness. Furthermore, such a method is scalable and robust to the introduction of more draft models or the removal of draft models, presenting a viable alternative to settings where a uniquely superior draft model may be unavailable.

Nevertheless, areas of further development exist. For example, learning in an online fashion may render this method more broadly applicable. Alternatively, exploring how to dynamically choose drafters at every decoding step rather than per example, as well as combining this direction of work with that which attempts to adaptively choose the speculation length at every step, are feasible ways of combining our findings with concurrent work in the hopes of reaping the benefits of all methods.

6 Limitations

This work has a few limitations which define the scope of future work.

Choice of draft models and data domains

Results may stem from the distinct boundaries that exist between domains/tasks. In settings where such boundaries are not well defined, outcomes may differ. However technical limitations and the absence of sufficent pre-trained models for comparision makes this difficult to explore immediately.

Additional storage and memory

The usage of multiple models that draft independently requires additional memory, which can be be more difficult to manage when there are explicit constraints on this front (self-drafting avoids this due to the use of a single model). Furthermore, collecting an offline dataset can be difficult in some specific scenarios where inference is burdensome, for example when input/output sequences are very long, or when many offline examples are required.

Self-Drafting

We work on a setting where we do not conduct any additional training of parameters that are explicitly linked to the language model itself, whether they are existing parameters or new paramters added as a result of the method. While there are ways in which our explored method can be applied to these as well, computational limitations make it difficult to rigorously conduct such studies at the moment and we leave it to future work for this reason.

7 Ethics Statement

This paper discusses the concept of dynamically choosing between of multiple black-box draft models for speculative decoding, proposing an offline reinforcement learning approach for adaptively selecting a good draft model for assistance. Our results are relate to the decoding speed of models, which is unlikely to lead to ethical concerns or problematic interpretations of such results.

8 Acknowledgements

Jerry Huang recieved financial support under the form of a National Science and Engineering Research Council (NSERC) Canada Graduate Scholarship, a Fonds de Recherche du Québec Nature et technologies (FRQNT) Training Scholarship and a Bourse d'Excellence Hydro-Québec. Sarath Chandar is supported by a Canada CIFAR AI Chair, the Canada Research Chair in Lifelong Machine Learning and a NSERC Discovery Grant.

References

Peter Auer. 2003. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null):397–422.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.

Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit K. Sanghai, Fei Sha, and William Cohen. 2022. Fido: Fusion-in-decoder optimized for stronger performance and faster inference. *ArXiv*, abs/2212.08153.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

Daniel Deutsch, Rotem Dror, and Dan Roth. 2022. On the limitations of reference-free evaluations of generated text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10960–10977, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam M. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:120:1–120:39.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. *Preprint*, arXiv:2402.02057.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Sophia Shao. 2024. Speed: Speculative pipelined execution for efficient decoding. *Preprint*, arXiv:2310.12072.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Parsa Kavehzadeh, Mojtaba Valipour, Marzieh Tahaei, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2024. Sorted LLaMA: Unlocking the potential of intermediate layers of large language models for dynamic inference. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2129–2145, St. Julian's, Malta. Association for Computational Linguistics.

Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Sophia Shao, and Amir Gholami. 2023a. Full stack optimization of transformer inference. In *Architecture and System Support for Transformer Models (ASSYST @ISCA 2023)*.

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

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024a. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *Preprint*, arXiv:2406.03853.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Ion Stoica, Zhijie Deng, Alvin Cheung, and Hao Zhang. 2024b. Online speculative decoding.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

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

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947, Online. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Siqi Wang, Hailong Yang, Xuezhu Wang, Tongxuan Liu, Pengbo Wang, Xuning Liang, Kejie Ma, Tianyu Feng, Xin You, Yongjun Bao, Yi Liu, Zhongzhi Luan, and Depei Qian. 2024. Minions: Accelerating large language model inference with adaptive and collective speculative decoding. *Preprint*, arXiv:2402.15678.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Ronald J. Williams. 2004. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Michael Woodroofe. 1979. A one-armed bandit problem with a concomitant variable. *Journal of the American Statistical Association*, 74(368):799–806.

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

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Preprint*, arXiv:2401.07851.

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

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

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2024. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*.

A Experimental Details

A.1 Baselines

To baseline and compare our architectural constraints against (Leviathan et al., 2023), we partially benchmark our experiments against theirs. These are presented in Table 7 and 8. We conduct this as we suspect a difference in both system architechture used for experiments as well as for implementation of models.

We observe that their results are generally show speedups that are consistently 2.5 to 3.0× as large as ours, with minor deviations. We attribute this to the usage of different computational resources and potential implementation differences. Additionally, given the small amount of variation in the relative differences between the observed and reported speedups, we contend that these differences are not due to errors in implementation.

A.2 Technical Details

All experiments are conducted on a machine with a single NVIDIA A100 GPU with 8 CPU cores. We run all experiments using PyTorch and HuggingFace models.

A.3 Hyperparameter Configurations

Details about hyperparameters we use within our experiments are detailed here.

A.4 Templates

For each example, we use a specific prompt based on the dataset from which the data originates (see Table 10). We follow the templates provided originally by Raffel et al. (2020) and Chung et al. (2022).

A.5 Cost Function

When considering multiple draft candidates, we use the following simple function for generating fixed costs for the different models. Suppose the candidates have $P = \{p_1, p_2, \ldots, p_k\}$ parameters. Then the cost for the models are

$$c_i = 1 - \frac{e^{p_i}}{e^{p_j}}$$

where $j = \underset{i}{\text{argmax}} \ p_i$.

A.6 Accept Rate Computation

To compute the accept rate of tokens, we define the **number of generated tokens** in a given draft as the total number of tokens generated by the draft model (this is equivalent to γ). The **number of accepted tokens** in a given draft is the number of generated tokens that are validated as correct by the target model. When a token is rejected within a draft, all subsequent tokens are considered rejected as well. The accept rate is then the quotient of the total number of accepted tokens divided by the total number of generated tokens.

A.7 Computing Wall-Clock Performance

To compute the wall-clock time when using a policy, we include the amount of time used to infer on the policy. However, we do not include the time needed to generate the sentence representation.

This is because upon generating the original sentence representation, the large model's KV cache can be updated to store these for the future verification passes, meaning that they do not need to be recomputed again in the future. As such, we treat this initial pass through as being part of the first verification pass.

Additionally, one could theoretically save on the policy inference by performing batched inference on many examples at once. However, this is not particularly applicable in practice, where different inputs arrive at different times. As such, we treat each example individually and include these times within the per-example speeds.

B Different Sampling Hyperparameters

We run our ablations based on our setup for our first experiment.

Target Model	Draft Model	Temperature	Draft Tokens	Ours	Original	Relative Difference
T5-XXL	None	0	-	1.00× (19.6 ms/token)	1.0×	-
T5-XXL	T5-Small	0	7	1.21× (16.2 ms/token)	3.4×	2.81×
T5-XXL	T5-Base	0	7	0.96× (20.4 ms/token)	2.8×	2.91×
T5-XXL	T5-Large	0	7	0.61× (32.1 ms/token)	1.7×	2.78×
T5-XXL	T5-Small	1	7	1.03× (19.1 ms/token)	2.6×	2.53×
T5-XXL	T5-Base	1	5	0.83× (23.5 ms/token)	2.4×	2.88×
T5-XXL	T5-Large	1	3	0.56× (35.3 ms/token)	1.4×	2.52×

Table 7: Reproduced translation results

Target Model	Draft Model	Temperature	Draft Tokens	Ours	Original	Relative Difference
T5-XXL	None	0	-	1.00× (31.8 ms/token)	1.0×	-
T5-XXL	T5-Small	0	5	0.99× (32.1 ms/token)	3.1×	3.13×
T5-XXL	T5-Base	0	5	0.87× (36.4 ms/token)	3.0×	3.43×
T5-XXL	T5-Large	0	3	0.59× (53.8 ms/token)	2.2×	3.72×
T5-XXL	T5-Small	1	5	0.86× (33.0 ms/token)	2.3×	2.39×
T5-XXL	T5-Base	1	5	0.85× (37.5 ms/token)	2.2×	2.59×
T5-XXL	T5-Large	1	3	0.57× (48.6 ms/token)	1.7×	2.60×

Table 8: Reproduced summarization results

Hyperparamter	**Value**
Optimizer	AdamW
Learning Rate	0.001
Weight Decay	0.01
β_1	0.9
β_2	0.99
ϵ	1e-8

Table 9: Optimization Hyperparameters

Task	**Prompt**
EnDe	`translate English to German: {input}`
XSum	`summarize: {input}`
GSM8K	`Q: {input}`

Table 10: Prompts for the different tasks

B.1 Effect of Number of Draft Tokens

We test our method with 5, 7 and 10 draft tokens in Table 11 and Table 12.

B.2 Effect of Temperature

We test our method with varying temperature values in Table 13 and Table 14.

We noted through ablations that increasing temperature past $T = 1$ resulted in a signficant slowdown in the decoding speed as well as the quality of the sampled generations. As such, we only present results on values of $T \leq 1$.

DRAFT MODEL	$\gamma = 5$	$\gamma = 7$	$\gamma = 10$
Speedup			
Auto-regressive		1.00× (31.06 ms/token)	
`T5-Small`	1.16× (28.93 ms/token)	1.19× (28.14 ms/token)	1.16× (28.95 ms/token)
`T5-Small-XSUM`	0.80× (38.92 ms/token)	0.83× (37.61 ms/token)	0.82× (38.04 ms/token)
Greedy Policy π_θ	1.09× (28.45 ms/token)	1.09× (28.56 ms/token)	1.09× (28.53 ms/token)
Dynamic Policy π_θ	1.06× (29.53 ms/token)	1.07× (29.20 ms/token)	1.06× (29.77 ms/token)

Table 11: Varying the number of drafted tokens for assisted generation on IWSLT2017 EN-DE. $T = 1$ for all cases.

DRAFT MODEL	$\gamma = 5$	$\gamma = 7$	$\gamma = 10$
Speedup			
Auto-regressive		1.00× (37.06 ms/token)	
`T5-Small`	0.96× (38.84 ms/token)	0.97× (38.60 ms/token)	0.96× (38.98 ms/token)
`T5-Small-XSUM`	1.19× (31.20 ms/token)	1.21× (30.71 ms/token)	1.19× (31.39 ms/token)
Greedy Policy π_θ	1.15× (32.30 ms/token)	1.17× (31.63 ms/token)	1.14× (32.44 ms/token)
Dynamic Policy π_θ	1.15× (32.33 ms/token)	1.16× (31.88 ms/token)	1.14× (32.50 ms/token)

Table 12: Varying the number of drafted tokens for assisted generation on XSUM. $T = 1$ for all cases.

DRAFT MODEL	$T = 0.5$	$T = 0.9$	$T = 1$
Speedup			
Auto-regressive		1.00× (31.06 ms/token)	
`T5-Small`	1.06× (29.00 ms/token)	1.09× (28.36 ms/token)	1.10× (28.14 ms/token)
`T5-Small-XSUM`	0.83× (37.70 ms/token)	0.81× (37.53 ms/token)	0.83× (37.61 ms/token)
Greedy Policy π_θ	1.11× (27.83 ms/token)	1.11× (27.94 ms/token)	1.09× (28.56 ms/token)
Dynamic Policy π_θ	1.07× (29.02 ms/token)	1.11× (28.26 ms/token)	1.07× (29.20 ms/token)

Table 13: Varying temperature for assisted generation on IWSLT2017 EN-DE. $\gamma = 7$ for all cases.

DRAFT MODEL	$T = 0.5$	$T = 0.9$	$T = 1$
Speedup			
Auto-regressive		1.00× (37.06 ms/token)	
`T5-Small`	0.98× (38.21 ms/token)	0.98× (38.31 ms/token)	0.97× (38.60 ms/token)
`T5-Small-XSUM`	1.18× (31.31 ms/token)	1.19× (31.23 ms/token)	1.21× (30.71 ms/token)
Greedy Policy π_θ	1.16× (31.93 ms/token)	1.18× (31.43 ms/token)	1.17× (31.63 ms/token)
Dynamic Policy π_θ	1.17× (31.50 ms/token)	1.18× (31.37 ms/token)	1.16× (31.88 ms/token)

Table 14: Varying temperature for assisted generation on XSUM. $\gamma = 7$ for all cases.