**Abstract**

Large Language Models, with improving accuracy in recent decades, suffer from slow inference due to the sequential nature of token-by-token generation and their increasing model complexity. Speculative decoding has emerged as a promising solution for those methods. However, existing methods often suffer from low acceptance rates caused by the draft model generating tokens that misalign with the target model's preferences, and memory-intensive designs that limit efficiency in resource-constrained settings. In this work, we proposed a computationally efficient speculative decoding method, TopK-ImpSpecDec, which built on speculative decoding and incorporating the Top-K candidates selection and importance sampling. Through extensive experiments on multiple datasets and parameter configurations, our approach demonstrates its ability to significantly improve token acceptance rates while also showing the potential to achieve substantial speed-ups in long-text generation, making it a practical solution for accelerating Large Language Model inference.

# 1 Introduction

Large Language Models (LLMs) have revolutionized the field of natural language processing (NLP), powering applications such as chatbots, translation tools, and summarization systems. However, the sequential nature of text generation, where each token generation requires a complete forward pass through the model, introduces significant latency. Speculative decoding[1, 2] has emerged as a promising solution to alleviate these latency issues. This approach speeds up the inference process by using a smaller draft model to generate token candidates, which are then verified by a larger target model.

However, speculative decoding still faces key challenges that limit its effectiveness. One major challenge is the low acceptance rate, as a significant proportion of token candidates generated by the draft model are often rejected by the target model[3]. This not only reduces the speed-up potential of speculative decoding but also risks disrupting the coherence of the surrounding context generated by the draft model, as fallback mechanisms in the target model introduce variance into the decoding process. Another challenge is that current approaches often suffer from high memory usage, particularly when employing methods like multiple draft model frameworks[4] or intermediate feature vector computations[5, 6]. These memory-intensive designs can restrict the practicality of speculative decoding in resource-constrained environments.

**Contributions.** This project makes the following contributions,

- It proposes a memory-efficient speculative decoding method using Top-K candidate selection and importance sampling.
- It thoroughly evaluates the proposed method across various datasets, showing its ability to improve token acceptance rates and its potential to achieve significant speed-ups in long-text generation.

# 2 Related Work

## 2.1 Speculative Decoding

To address the inefficiency of token-by-token generation in LLMs, Schuster et al. and Chen et al. proposed speculative decoding [1, 2]. It uses a smaller draft model to generate multiple tokens each time, which are then verified in parallel by the target model, significantly improving inference speed.

Previous research has explored ways to improve speculative decoding. Predictive Pipeline Decoding[7]

uses the intermediate output of the target model as draft tokens, reducing the redundancy of maintaining and running a separate draft model. Extrapolation Algorithm for Greater Language-model Efficiency(EAGLE)[5] and EAGLE 2[6] introduce intermediate feature vectors of the target model and a look-ahead evaluation mechanism to improve inference speed. Medusa[4] introduces a multi-draft framework that employs multiple smaller draft models to generate diverse token proposals in parallel, thereby improving the acceptance rate. Adaptive Drafting[8] dynamically adjusts the generation strategy of the draft model to enhance token alignment based on feedback from the target model. However, these approaches have some drawbacks. For example, Medusa requires significant memory to handle multiple draft models, making it resource-intensive. Similarly, EAGLE and EAGLE 2 increase memory usage by relying on intermediate feature vectors and look-ahead evaluations, which may restrict their practicality in low-resource environments.

### 2.2 Top-K Sampling

In speculative decoding, Top-K Sampling[9] selects the most probable tokens at a given position, renormalizes their probabilities, and samples one using these weights, thereby improving the acceptance rate. Building on this approach, SpecTr[10] combines Top-K sampling with optimal transport techniques to accelerate decoding while maintaining high-quality outputs. Similarly, Recursive Speculative Decoding (RSD)[11] employs Top-K sampling in a tree-based framework, generating diverse token sequences by sampling without replacement, which enhances both diversity and efficiency in LLM inference. However, both methods can result in significant memory overhead due to the optimal transport techniques in SpecTr and the tree-based structure of RSD.

### 2.3 Importance Sampling

Importance sampling[12] is a statistical method that approximates a target distribution by drawing samples from a simpler distribution and reweighting them to ensure accuracy. While importance sampling has not been directly applied to speculative decoding, it has been utilized in various aspects of LLM training and fine-tuning. For instance, Pan et al. proposed a method named LISA, which applies importance sampling across different layers of LLMs, enabling selective freezing of certain layers to improve fine-tuning efficiency[13]. Xie et al. used importance sampling to align pretraining datasets with a target distribution, optimizing data selection for both general-domain and domain-specific models[14].

Importance sampling aligns with the goal of speculative decoding by refining the alignment between an easy-to-obtain distribution and the target distribution. Additionally, its computational efficiency makes it an ideal choice for integration. By combining importance sampling with the Top-K candidate selection technique borrowed from Top-K sampling, our method avoids the need for complex structures or multiple draft models, significantly reducing computational overhead and memory usage while improving the draft token acceptance rate.

## 3 Methodology

This proposed speculative decoding approach (TopK-ImpSpecDec) combines Top-K candidate selection with importance sampling. While Top-K candidate selection increases the likelihood of containing tokens favored by the target model, importance sampling adjusts the draft model's token probabilities using the target model's distribution, ensuring alignment between the two models. Together, these two techniques improve both the acceptance rate and the decoding speed. A detailed breakdown of this method will be provided in this section.

### 3.1 Top-K Candidate Selection

In Standard Speculative Decoding[15], at each position, the draft model generates a chunk of $\gamma$ tokens sequentially given a prefix context $x_{<t}$. Each token is sampled from the draft model's probability distribution: $x_i \sim q(x_i \mid x_{<i})$ and there is only one single candidate token for each position $x_t, ..., x_{t+\gamma-1}$. Then, the target model evaluates the same tokens using its distribution $p(x_i \mid x_{<i})$ and computes an acceptance probability, if token $x_i$ isn't accepted, a fallback mechanism replaces the rejected token by sampling from the adjusted distribution $p'(x) = norm(max(0, p(x) - q(x)))$ instead[15]. However, since the draft model only proposes one candidate token per position, alternative tokens that may better align with the target model are not considered, which may lead to low acceptance rates and more frequent fallback actions.

The Top-K candidate selection, as an initial step of Top-K sampling, provides an enhancement by considering the $K$-most probable token candidates ($\mathcal{T}_K$) for each token position $i$. These candidates are selected by retrieving the K tokens with the highest probabilities in draft model's distribution $q(x_i \mid x_{<i})$, it is calculated as by Equation (1) where $V$ is the set of all possible tokens that the draft model can generate.

$$\mathcal{T}_K = \{y_j | j \in 1 : K\} = argmax_{y \in V, |T_K| = K} q(x_i \mid x_{<i}) \tag{1}$$

Top-K candidate selection helps address the limitation of the draft model of proposing only one token per position by considering multiple candidate tokens. This increases the likelihood of alignment with the target model, thereby improving the acceptance rate and reducing the occurrence of fallback operations.

### 3.2 Importance Sampling

Importance sampling is a statistical technique used to estimate values efficiently when working directly with a target distribution $p(x)$ is costly. Instead of sampling directly from $p(x)$, samples are drawn from a computationally simpler distribution $q(x)$. To account for the differences between $q(x)$ and $p(x)$, each sample is assigned a weight: $w(x) = \frac{p(x)}{q(x)}$ to determine the relative importance of the sample in approximating the target distribution $p(x)$. By weighting the samples in this manner, it ensures that the results reflect the target distribution $p(x)$ accurately, even though the samples are drawn from $q(x)$.

The connection between speculative decoding and importance sampling is that we aim to align the tokens sampled by the draft model distribution $q(x)$ more closely with the target model distribution $p(x)$. Therefore, based on the selection of the Top-K candidates for each token, we calculate an importance weight for each candidate Equation (2) and normalize these weights.

$$w(y_j) = \frac{p(y_j | x_{<i})}{q(y_j | x_{<i})} \tag{2}$$

Then a token is sampled from the Top K set $\mathcal{T}_K$ with normalized importance weights $w'(y_j)$ using Equation (3) where the probability of selecting each token is proportional to $w'(y_j)$. After obtaining $y_i$ for all positions from 1 to $\gamma$, the remaining steps follow the same process as standard speculative decoding.

$$y_i \sim \text{Categorical}\left(\mathcal{T}_K, \{w'(y_1) : w'(y_k)\}\right) \tag{3}$$

### 3.3 Top-K Importance Sampling Speculative Decoding Algorithm

Algorithm 1 shows the algorithm of our proposed TopK-ImpSpecDec method, with lines 19-29 adopted from Leviathan et al.'s work[15].

**Algorithm 1** Speculative Decoding with Top-K importance sampling

**Inputs:** Draft model $M_q$, Target model $M_p$, Chunk size $\gamma$, Top-K size $K$, Prefix sequence *prefix*

**for** $i = 1$ ***to*** $\gamma$ **do**
  ▷ Sample Top-K candidates
  $q(x_i) \leftarrow M_q(x_i \mid \text{prefix} + [x_1, \ldots, x_{i-1}])$
  $\mathcal{T}_K \leftarrow \text{argmax}_{x \in \mathcal{V}, |\mathcal{T}_K| = K}\, q(x_i)$
  ▷ Run $M_p$ in parallel.
  $p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
    $\text{M}_p(\text{prefix}), \ldots, M_p(\text{prefix} + [x_1, \ldots, x_\gamma])$
  ▷ Reweight Top-K candidates using importance sampling.
  **for** $\{y_j | j = 1 : K\} \in \mathcal{T}_K$ **do**
    $q(y_j) \leftarrow M_q(y_j \mid \text{prefix} + [x_1, \ldots, x_{i-1}])$
    $p(y_j) \leftarrow M_p(y_j \mid \text{prefix} + [x_1, \ldots, x_{i-1}])$
    $w(y_j) \leftarrow \frac{p(y_j)}{q(y_j)}$
  **end for**
  ▷ Normalize weights.
  $w'(y_j) \leftarrow \frac{w(y_j)}{\sum_{y_j \in \mathcal{T}_K} w(y_j)}$
  ▷ Sample one token from the Top-K candidates using the normalized importance weight $w'(x)$
  $y_i \sim \text{Categorical}\left(\mathcal{T}_K, \{w'(y_1) : w'(y_k)\}\right)$
**end for**
▷ Determine the number of accepted guesses $n$.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p(y_i)}{q(y_i)}\} \cup \{\gamma\})$
▷ Adjust the distribution from $M_p$ if needed.
$p'(x) \leftarrow p_{n+1}(x)$
**if** $n < \gamma$ **then**
  $p'(x) \leftarrow \text{norm}(\max(0, p_{n+1}(x) - q_{n+1}(x)))$
**end if**
▷ Return one token from $M_p$, and $n$ tokens from $M_q$.
$z \sim p'(x)$
**return** *prefix* + $[x_1, \ldots, x_n, z]$

# 4 Experiments and Results

## 4.1 Experiment Settings

We evaluate the proposed TopK-ImpSpecDec method using draft and target models from the LLaMA family to ensure compatibility in tokenization and alignment in probability distributions, which are crucial for improving token acceptance rates and simplifying the integration of speculative decoding[16]. Specifically, we used a customized small-size LLaMA-160M[17] model as the draft model and LLaMA-7B[17] as the target model. All experiments were completed on an NVIDIA A100 40GB GPU.

To address the substantial gap between the draft and target models in speculative decoding, we trained the draft model using a combination of classification Loss($L_{\text{cls}}$) and regularization losses($L_{\text{reg}}$) using Equation (4) [5] and set $w_{\text{cls}} = 0.1$. $L_{\text{cls}}$ computes the KL divergence between the draft model's token probabilities and the target model's probabilities, ensuring alignment of their output distributions. $L_{\text{reg}}$ uses Smooth $L1$ loss[5] applied after projecting the draft model's hidden states to match the target model's higher-dimensional space, enhancing representation alignment. This training approach effectively brought the draft model's probability distribution closer to that of the target model, improving its performance in speculative decoding.

$$L = L_{\text{reg}} + w_{\text{cls}} \cdot L_{\text{cls}} \tag{4}$$

### 4.2 Dataset and Preprocessing

We evaluate our approach using three datasets: WikiText-103[18], Alpaca[19], and CNN/DailyMail[20], each with distinct characteristics. WikiText-103 features extensive vocabulary and diverse content and is used to evaluate our approach's ability to maintain long-term coherence and contextual alignment during open-ended text generation. The other two datasets focus on more structured tasks: Alpaca focuses on instruction-following tasks, providing insights into the performance of the approach in goal-oriented scenarios, while CNN/DailyMail, a widely used benchmark for summarization, assesses the approach's ability to generate clear and precise summaries.

For preprocessing, WikiText-103 was prepared by removing empty samples and filtering for desired context length in order to split each sample into two parts: a context section as input prompt and a target section as the output. For Alpaca, each sample included an instruction, an optional input, and an output. The instruction and input (if present) were merged into a single prompt. Both the prompt and output were tokenized and truncated to maintain fixed lengths. For the CNN/DailyMail dataset, preprocessing involved filtering out empty article-summary pairs and adding a prompt "Summarize the article:" to each article. Articles were truncated to the maximum context length, while summaries were limited to a fixed number of tokens, ensuring compatibility with the model's training format.

### 4.3 Baseline Methods

To evaluate the effectiveness of our proposed TopK-ImpSpecDec method, we compared it with two baseline methods. The first baseline method is Direct Target Generation (DTG), where the target model directly generates tokens without the assistance of the draft model. This method serves as a reference point to assess the generation speed of traditional sequential decoding, providing a basis for comparing the speed improvements achieved by speculative decoding. The second baseline method is the Standard Speculative Decoding(Std-SpecDec), which is used to compare the improvements brought by our proposed method within the domain of speculative decoding.

### 4.4 Evaluation Metrics

We evaluated our proposed method and the baselines using the following metrics.

#### 4.4.1 Acceptance Rate

The acceptance rate($\alpha$) measures the proportion of tokens proposed by the draft model that are successfully accepted by the target model during decoding (Equation (5)). A higher acceptance rate indicates that the draft model closely approximates the target model's probability distribution, ensuring better alignment between the two models. Therefore, the acceptance rate is essential to assess the effectiveness of speculative decoding in balancing efficiency and accuracy.

$$\alpha = \frac{\text{Number of Accepted Tokens}}{\text{Total Proposed Tokens}} \tag{5}$$

#### 4.4.2 Speed-Up

The speed-up metric (Equation (6)) measures the degree of acceleration achieved by speculative decoding compared to DTG. A higher speed-up indicates that speculative decoding significantly reduces the time required for text generation. Therefore, speed-up helps demonstrate how a speculative decoding method can improve inference speed.

$$\text{Speed-Up} = \frac{\text{Time Taken by DTG}}{\text{Time Taken by Speculative Decoding}} \tag{6}$$

### 4.5 Key Parameters Setting

In this experiment, the key parameters are chunk size ($\gamma$), temperature ($T$), and the number of Top-K candidates ($K$). A higher $\gamma$ allows faster generation by reducing iterations but risks lower acceptance rates because generating larger chunks of tokens reduces the draft model's ability to refine its predictions to align with the target model's expectations. In contrast, a lower $\gamma$ improves alignment at the cost of slower generation. $T$ controls sampling randomness, such that higher $T$ enhance diversity but reduce acceptance rates, which can lead to more fallback events. Lower $T$, on the other hand, improve alignment and increase acceptance rates. The $K$ limits sampling to the K most probable tokens. A higher $K$ improves alignment with the target model, boosting acceptance rates but increasing computational costs and potentially slowing inference. Lower $K$ may reduce acceptance rates but may save computational costs and speed up inference.

Our experiment tested two temperature settings, $T = 0$ and $T = 1$. For each dataset, we compare the performance of our approach and standard speculative decoding under three different $\gamma$ values to analyze the trade-off between generation speed and alignment with the target model. Additionally, we fixed $\gamma$ and $T$ values to test three different $K$, which allowed us to observe how different numbers of candidate tokens influence overall performance. The combination of these experiments provides a comprehensive understanding of the impact of key parameters on the balance between acceptance rate, inference speed, and token diversity.

### 4.6 Results

Table 1 shows the performance results of Std-SpecDec and TopK-ImpSpecDec across three datasets under different conditions with a fixed $K = 50$.

| **Dataset** | **Temp.** | $\gamma$ | Std-Spec. Decoding (Acc. Rate) | Std-Spec. Decoding (Speed-Up) | TopK-Imp.Spec. Decoding (Acc. Rate) | TopK-Imp.Spec. Decoding (Speed-Up) |
|---|---|---|---|---|---|---|
| WikiText-103 | $T_1 = 1$ | $\gamma_1 = 5$ | 25.29 | 1.02× | 77.58 | 1.01× |
| | | $\gamma_2 = 10$ | 22.82 | 1.30× | 77.40 | 1.29× |
| | | $\gamma_3 = 15$ | 20.26 | 1.64× | 76.76 | 1.60× |
| | $T_2 = 0$ | $\gamma_1 = 5$ | 45.07 | 1.01× | 88.34 | 1.00× |
| | | $\gamma_2 = 10$ | 36.96 | 1.31× | 88.28 | 1.28× |
| | | $\gamma_3 = 15$ | 28.30 | 1.64× | 86.34 | 1.61× |
| Alpaca | $T_1 = 1$ | $\gamma_1 = 5$ | 24.56 | 1.17× | 78.46 | 1.15× |
| | | $\gamma_2 = 10$ | 21.80 | 1.54× | 77.94 | 1.49× |
| | | $\gamma_3 = 15$ | 21.48 | 1.40× | 78.56 | 1.37× |
| | $T_2 = 0$ | $\gamma_1 = 5$ | 24.37 | 1.81× | 97.50 | 1.20× |
| | | $\gamma_2 = 10$ | 21.07 | 1.55× | 98.62 | 1.59× |
| | | $\gamma_3 = 15$ | 20.67 | 1.44× | 99.13 | 1.47× |
| CNN/DailyMail | $T_1 = 1$ | $\gamma_1 = 4$ | 19.83 | 1.05× | 73.90 | 1.05× |
| | | $\gamma_2 = 8$ | 19.20 | 1.31× | 73.05 | 1.27× |
| | | $\gamma_3 = 15$ | 16.08 | 1.43× | 74.18 | 1.40× |
| | $T_2 = 0$ | $\gamma_1 = 4$ | 9.95 | 1.21× | 87.77 | 1.04× |
| | | $\gamma_2 = 8$ | 31.37 | 1.29× | 86.38 | 1.29× |
| | | $\gamma_3 = 15$ | 23.72 | 1.44× | 84.63 | 1.43× |

Table 1: Performance Comparison of TopK-ImpSpecDec vs. Std-SpecDec Across Datasets

#### 4.6.1 Acceptance Rate Analysis

As show in Figure 1, across all datasets, TopK-ImpSpecDec significantly improves acceptance rates compared to Std-SpecDec by better aligning the draft model's outputs with the target model's preferences. As show in Figure 2, while acceptance rates generally decline for both methods as chunk size $\gamma$ increases, TopK-ImpSpecDec mitigates this drop more effectively, and even demonstrates instances of an increase in acceptance rates. This is particularly advantageous for long-form text generation, where decoding larger chunks efficiently is essential. This higher acceptance rates in TopK-ImpSpecDec reduce the need for fallback tokens from the target model, which may create

inconsistencies in style, vocabulary, or fluency due to abrupt deviations from the surrounding context generated by the draft model. By maximizing token acceptance, TopK-ImpSpecDec ensures smoother transitions, leading to more cohesive and natural text generation.

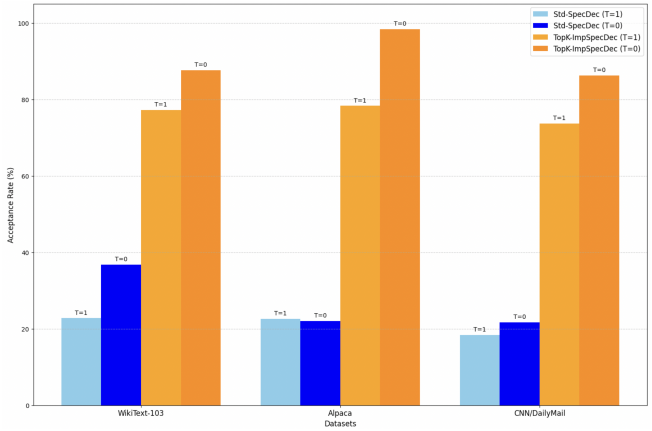


Figure 1: Average Acceptance Rate Across Datasets and Decoding Temperatures (T=0, T=1)

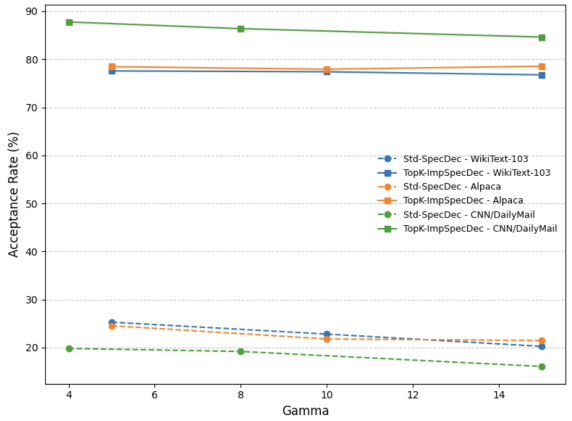


Figure 2: Acceptance Rate vs. $\gamma$ for Different Datasets and Decoding Methods (Temp.=1)

### 4.6.2 Speed-up Analysis

Across all datasets, TopK-ImpSpecDec achieves competitive speed-ups compared to Std-SpecDec. Additionally, we observed that TopK-ImpSpecDec has the potential to demonstrate better speed-ups than the standard approach in deterministic decoding and with larger chunk sizes (eg. $T_2 = 0, \gamma = 10$). This makes it ideal for fast inference in large-scale text generation where Std-SpecDec may lack speed-up strength.

In addition, we initially worried that as the value of $K$ increases, the speed-up would decrease due to the computational overhead introduced by a larger $K$. However, as shown in Table 2, increasing $K$ may actually have the potential to boost the speed-up. This is because the computational overhead introduced by a larger $K$ is effectively balanced by the improved acceptance rate, which reduces the need for fallback mechanisms and therefore save time.

| **Dataset** | **Temp.** | $\gamma$ | **Top-K** | TopK-Imp.Spec Decoding (Acc. Rate) | TopK-Imp.Spec Decoding (Speed-Up) |
|---|---|---|---|---|---|
| WikiText-103 | $T_1 = 1$ | $\gamma = 10$ | $K_1 = 10$ | 61.87 | 1.29× |
| | | | $K_2 = 25$ | 74.11 | 1.27× |
| | | | $K_3 = 50$ | 77.40 | 1.29× |
| Alpaca | $T_1 = 1$ | $\gamma = 10$ | $K_1 = 10$ | 56.25 | 1.51× |
| | | | $K_2 = 25$ | 68.55 | 1.52× |
| | | | $K_3 = 50$ | 77.94 | 1.49× |
| CNN/Daily Mail | $T_1 = 1$ | $\gamma = 8$ | $K_1 = 10$ | 52.54 | 1.29× |
| | | | $K_2 = 25$ | 66.71 | 1.30× |
| | | | $K_3 = 50$ | 73.05 | 1.27× |

Table 2: Performance Comparison of TopK-ImpSpecDec Across Datasets with Varying $K$

## 5 Discussion and Conclusion

This project introduces an innovative approach that combines Top-K candidate selection and importance sampling to enhance speculative decoding. We thoroughly evaluated the method across various datasets, and the results show significant improvements in token acceptance rates over standard speculative decoding. These improvements reduce the reliance on fallback tokens, ensuring

smoother token generation and minimizing abrupt deviations introduced by target model-generated tokens. Moreover, as both importance sampling and Top-K candidate selection are designed to be relatively space-efficient, our approach ensures strong performance even in memory-constrained environments.

However, a limitation of our approach lies in its speed-up performance. While it achieves competitive speed-up compared to standard speculative decoding, it does not consistently surpass it in all scenarios. The additional computations required for Top-K candidate selection introduce slight overhead, which can offset potential gains in decoding speed. Nevertheless, our method shows promise for achieving better speed-ups than the standard approach when chunk sizes are larger. This makes it well-suited for fast inference in large-scale text generation.

Future work could focus on further optimizing the approach to reduce computational overhead and enhance decoding speed, as well as conducting more detailed experiments with different datasets to demonstrate the advantages of this method in terms of acceptance rate and speed-up when applied to larger chunk sizes.

## References


[1] T. Schuster, A. Fisch, J. Gupta, M. Dehghani, D. Bahri, V. Q. Tran, Y. Tay, and D. Metzler, "Confident adaptive language modeling," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7833–7849, 2022.

[2] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," *arXiv preprint arXiv:2302.01318*, 2023. [Online]. Available: https://arxiv.org/abs/2302.01318

[3] M. Khoshnoodi, V. Jain, M. Gao, M. Srikanth, and A. Chadha, "A comprehensive survey of accelerated generation techniques in large language models," *arXiv preprint arXiv:2405.13019*, 2024, under review.

[4] T. Chen, S. Cheng, Z. Li, T. Fu, T. Zhang, X. Chen, Z. Wu, Q. Chen, Y. Li, and J. Han, "Medusa: Simple framework for accelerating llm generation with multiple decoding heads," *arXiv preprint arXiv:2401.10774*, 2024.

[5] Q. Chen, Y. Li, Z. Wu, Y. Wang, T. Zhang, Z. Liu, W. Zhao, T. Fu, S. Yan, and J. Han, "Eagle: Speculative sampling requires rethinking feature generation," *arXiv preprint arXiv:2401.12442*, 2024.

[6] Q. Chen, Y. Li, Z. Wu, T. Zhang, Z. Liu, Y. Wang, T. Fu, and J. Han, "Eagle 2: Better speculative sampling with larger support," *arXiv preprint arXiv:2402.14952*, 2024.

[7] Z. Song, X. Hong, H. Zhang, Y. Xu, and J. Tang, "Predictive pipeline decoding: Learning to speculate token-by-token," *arXiv preprint arXiv:2309.06979*, 2023.

[8] Y. Li, Q. Chen, X. Chen, Z. Wu, T. Zhang, T. Fu, and J. Han, "Adaptive drafting: Aligning token generation for efficient speculative decoding," *arXiv preprint arXiv:2402.16847*, 2024.

[9] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *arXiv preprint arXiv:1805.04833*, 2018.

[10] Z. Sun, A. T. Suresh, J. H. Ro, A. Beirami, H. Jain, and F. Yu, "Spectr: Fast speculative decoding via optimal transport," *arXiv preprint arXiv:2401.08161*, 2024.

[11] W. Jeon, M. Gagrani, R. Goel, J. Park, M. Lee, and C. Lott, "Recursive speculative decoding: Accelerating llm inference via sampling without replacement," *arXiv preprint arXiv:2402.14160v2*, 2024. [Online]. Available: https://arxiv.org/abs/2402.14160v2

[12] Wikipedia contributors, "Importance sampling," https://en.wikipedia.org/wiki/Importance_sampling, 2024, accessed: 2024-12-15.

[13] R. Pan, X. Liu, S. Diao, R. Pi, J. Zhang, C. Han, and T. Zhang, "Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning," *arXiv preprint arXiv:2403.17919*, 2024.

[14] S. M. Xie, S. Santurkar, T. Ma, and P. Liang, "Data selection for language models via importance resampling," *arXiv preprint arXiv:2302.03169*, 2023. [Online]. Available: https://arxiv.org/abs/2302.03169

[15] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *Proceedings of the 40th International Conference on Machine Learning*, ser. PMLR 202. Honolulu, Hawaii, USA: PMLR, 2023.

[16] M. Yan, S. Agarwal, and S. Venkataraman, "Decoding speculative decoding," *arXiv preprint arXiv:2402.01528*, 2024.

[17] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Roziere, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[18] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016.

[19] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[20] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.