Model-free Speculative Decoding for Transformer-based ASR with Token Map Drafting

Tuan Vu Ho, Hiroaki Kokubo, Masaaki Yamamoto, and Yohei Kawaguchi
Hitachi, Ltd., Tokyo, Japan
tuanvu.ho.zt@hitachi.com, hiroaki.kokubo.dz@hitachi.com,
masaaki.yamamoto.af@hitachi.com, yohei.kawaguchi.xk@hitachi.com

***Abstract*—End-to-end automatic speech recognition (ASR) systems based on transformer architectures, such as Whisper, offer high transcription accuracy and robustness. However, their autoregressive decoding is computationally expensive, hence limiting deployment on CPU-based and resource-constrained devices. Speculative decoding (SD) mitigates this issue by using a smaller draft model to propose candidate tokens, which are then verified by the main model. However, this approach is impractical for devices lacking hardware accelerators like GPUs. To address this, we propose *Token Map Drafting*, a model-free SD technique that eliminates the need for a separate draft model. Instead, we leverage a precomputed n-gram token map derived from domain-specific training data, enabling efficient speculative decoding with minimal overhead. Our method significantly accelerates ASR inference in structured, low-perplexity domains without sacrificing transcription accuracy. Experimental results demonstrate decoding speed-ups of $1.27\times$ on the CI-AVSR dataset and $1.37\times$ on our internal dataset without degrading recognition accuracy. Additionally, our approach achieves a 10% absolute improvement in decoding speed over the Distill-spec baseline running on CPU, highlighting its effectiveness for on-device ASR applications.**

***Index Terms*—speculative decoding, token map drafting, transformer-based ASR, speech recognition**

I. Introduction

The adoption of encoder-decoder transformers [1] has significantly advanced automatic speech recognition (ASR), yielding substantial improvements in transcription accuracy [2]–[4]. These models utilize autoregressive decoding to systematically generate tokens, leveraging strong contextual awareness to enhance recognition performance. However, autoregressive inference is computationally expensive, particularly in resource-constrained environments, such as mobile devices that require real-time processing.

To address this challenge, various optimization techniques have been proposed to accelerate transformer inference. These include efficient attention mechanisms [5]–[7], model quantization [8], [9], and knowledge distillation [10], [11], each aiming to reduce computational overhead while maintaining transcription accuracy.

Speculative decoding (SD) [12] is a simple technique aims at accelerating autoregressive transformer inference by leveraging parallelized token proposals. It employs lightweight draft models to predict speculative token sequences, which are subsequently verified and corrected by the primary model. This approach has been further explored and enhanced in subsequent research. For instance, [13] proposed speculative sampling methods to expedite large language model decoding, while [14] focused on continuously updating draft models on observed user queries, reducing distribution shifts and enhancing prediction accuracy.

Recent works, such as [15] and [16], have explored SD in ASR by incorporating mechanisms like knowledge distillation and multi-head verification. While effective in large-scale systems, running an additional draft model locally increases computational overhead, potentially introducing more latency instead of improving decoding speed. The reliance on fine-tuned draft models therefore limits the scalability of these methods to lightweight, CPU-based deployments.

Drawing from statistical language modeling, we propose *Token Map Drafting*, a model-free SD method that leverages precomputed draft tokens derived from domain-specific transcriptions. Our approach generates draft tokens via an efficient n-gram token map, eliminating the need for an additional draft model. As such, it is particularly suited for structured ASR tasks in resource-constrained environments. Furthermore, structured text contexts characterized by low perplexity, such as maintenance commands or repetitive sequences, enhance its effectiveness.

In summary, our key contributions are: (1) we propose the *Token Map Drafting* model to enhance the inference speed of Whisper model on resource-constrained device, and (2) we conducted analysis to achieve optimal speculative decoding speed-up on domain-specific dataset.

II. Speculative Decoding

Introduced by [12], SD is a technique designed to accelerate autoregressive language models by utilizing a lightweight draft decoder to generate multiple candidate tokens in parallel. In conventional autoregressive inference, generating K tokens requires K sequential runs of the model. SD leverages the observation that language modeling tasks often contain simpler subtasks that can be well-approximated by a smaller model. Thus, a draft token sequence is first generated autoregressively using a smaller model and then verified in parallel by the main model in a single step. Since most of the draft token sequence is expected to match the final output of the main model, SD can significantly reduce inference time compared to conventional autoregressive decoding. Figure 1 illustrates the processing steps of speculative decoding.

The detailed steps of speculative decoding are as follows:

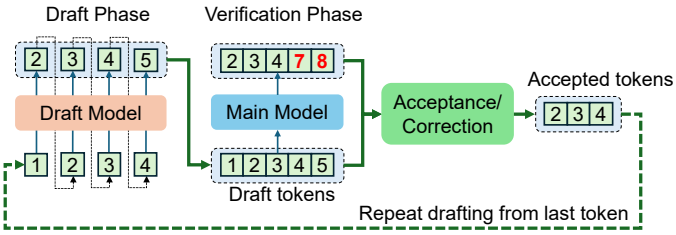

Fig. 1. Overview of speculative decoding. Divergent tokens is highlight in red, which is removed after Acceptance/Correction step.

1) **Draft model prediction:** A lightweight draft model (e.g. distilled model) generates a speculative sequence of k tokens given the current context x. This step is computationally inexpensive and helps in reducing the overall decoding time:

$$\hat{y}_{1:k} = \text{DraftModel}(x) \tag{1}$$

The draft model predicts tokens in parallel based on statistical patterns or simplified learned representations.
2) **Verification by the main model:** The primary (larger) model evaluates the draft sequence token by token by computing the conditional probability of each predicted token:

$$P(y_i|x, y_{1:i-1}) \quad \forall i \in \{1, ..., k\} \tag{2}$$

This step ensures that the speculative tokens comply with the main model expectations. If a token probability is above a predefined threshold, it is considered correct and accepted. Otherwise, the decoding process returns to the predictions of the main model itself.
3) **Acceptance and correction:** If the verification step confirms the draft tokens, they are accepted, and the decoding moves to the next segment. However, if any token does not meet the required confidence threshold, the main model generates a replacement token. This correction step ensures that errors from the draft model do not propagate.
4) **Continuation until completion:** Steps 1–3 are repeated iteratively until the full sequence is generated or an end-of-sequence (EOS) token is reached. If the speculative model provides high-quality predictions, the overall decoding process speeds up significantly while maintaining accuracy.
5) **Finalize output:** The final output sequence is constructed from the combination of accepted speculative tokens and any necessary corrections from the main model. The resulting sequence is computationally efficient and aligned with the quality expected from the primary model.

In the field of ASR, speculative decoding was explored in [15] through knowledge distillation for the Whisper model, demonstrating substantial decoding speed-ups. In this approach, a lightweight distilled model, such as distill-large-v3, serves as the draft model for the main model (e.g., Whisper-large-v3). Candidate tokens are generated using standard autoregressive decoding by the draft model. Since the distilled model and the main model share the same encoder structure, the verification step is performed only on the decoder of the main model.

Similarly, [16] extended speculative decoding by introducing a multi-head decoder with an additional K heads, enabling the prediction of $K+1$ tokens per step to improve efficiency. The $K+1$ draft tokens are then verified using the base head, similar to conventional speculative decoding. Trained on LibriSpeech [17] and VoxPopuli [18], these models achieve a 50% reduction in latency while maintaining competitive accuracy.

III. Proposed Methodology

A. Speculative decoding with Token Map Drafting

While SD effectively accelerates inference, conventional approaches depend on fine-tuning a draft model or modifying the model architecture, making them impractical for resource-constrained devices such as mobile platforms. However, many ASR applications in industrial domains operate within constrained vocabularies and structured sentence patterns, such as in-car commands or measurement descriptions. In such cases, the draft sequence can often be inferred from the first few generated tokens.

We propose a model-free SD method that leverages a precomputed *token map* to efficiently generate draft sequences without requiring an additional draft model By analyzing domain-specific text corpora, we construct a dictionary where each key represents an n-gram token sequence, and the corresponding value contains high-confidence candidate sequences that frequently follow it. Instead of sequentially generating tokens via a draft model, our approach selects speculative sequences from this token map, reducing computational overhead while maintaining decoding accuracy. This method is particularly effective in structured ASR tasks, such as Whisper-based transcription of repetitive commands.

Figure 2 demonstrate an example of our proposed SD method. The encoder processes a Mel-spectrogram input, and the decoder employs cross-attention. The decoding process begins with an entry sequence and a candidate sequence of tokens. Both sequences are expanded speculatively, and a sequence matching process identifies shared segments between them. Following validation, the tokens after the matched portion are determined, and the decoding process continues iteratively.

B. Token map construction

To build the token map, we begin by converting each sentence into a sequence of tokens using a pretrained tokenizer. The token map functions as a dictionary containing key-value pairs. Each key represents an n-gram sequence, and the corresponding value is a list of token sequences that follow the key sequence in the dataset. The length of n-gram sequence can vary 1 to N.

To optimize decoding efficiency, we rank and prune candidate token sequences based on token length and frequency

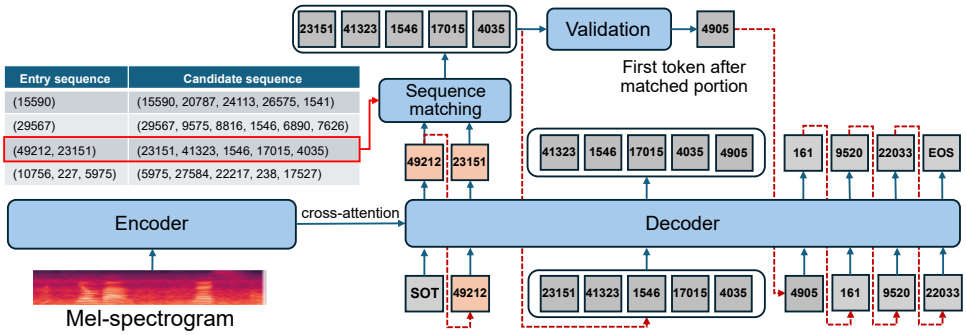

Fig. 2. Overview of our proposed speculative decoding with *Token Map Drafting*. 'SOT' represents the Start Of Transcript token, and 'EOS' denotes the end-of-sequence token.

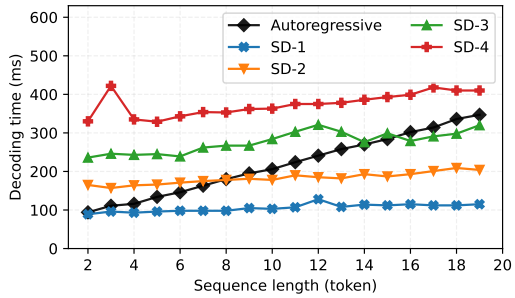

Fig. 3. Decoding time comparison between standard autoregressive decoding and SD. SD-K represents speculative decoding with K candidate token sequences.

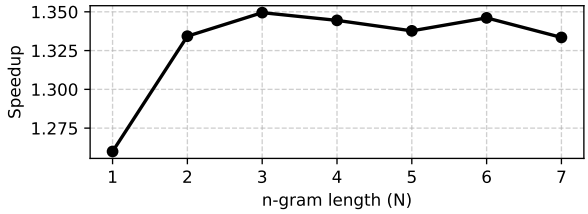

Fig. 4. Speedup analysis of processing times with varying n-gram length N.

thresholds, prioritizing longer sequences to maximize speedup while omitting low-confidence candidates. Since SD does not always outperform standard autoregressive decoding in every scenario, we empirically analyze inference time under varying candidate numbers and sequence lengths. Figure 3 shows that SD achieves speedup only when candidate sequence lengths exceed certain thresholds (e.g., 9 tokens for 2 candidates, 16 for 3 candidates). Beyond 3 candidates, SD show less effective. Based on these findings, we iteratively merge candidates with their nearest matches until the condition is met. When the decoder output misaligns with a speculative candidate, we handle mismatches by truncating the decoder state at the first unmatched token and resuming AR decoding until the next n-gram match is found. This ensures minimal additional delay without requiring a full re-run of the decoder.

To determine the optimal n-gram length N, we measure speedup rates across different values of N using Whisper-small on an internal dataset. Figure 4 shows that speedup peaks at $N = 3$, which we adopt for subsequent experiments.

We summarize our proposed method in the below pseudo-algorithm:

Algorithm 1 Token Map Construction Algorithm

Require: Training transcriptions $\mathcal{D}$
Ensure: Token map $\mathcal{T}$
for sentence $s \in \mathcal{D}$ **do**
  Tokenize s using pre-trained tokenizer
  Extract n-grams ($n = 1, 2, ..., N$) from tokens
  Add extracted n-grams to token tree $\mathcal{T}$ as key
end for
for n-gram k from token map $\mathcal{T}$ key **do**
  Extract all token sequences in $\mathcal{D}$ that follow k as candidates sequence
  Add extracted candidates sequences to $\mathcal{T}[key]$
end for
Prune $\mathcal{T}$ based on condition in Figure 3
return $\mathcal{T}$

IV. Experiments

In this section, we describe our experimental setup for evaluating the performance of our proposed method against the baseline. All experiments were conducted on a laptop with Intel Core i5 CPU. The implementation was developed in $C++$ using the CTranslate2 [1] library.

A. Datasets and Setup

We evaluate the performance of speculative decoding using two datasets:

- **CI-AVSR:** [19] A dataset designed for continuous integration of audio-visual speech recognition, containing diverse utterances across various speakers and environments.
- **Our Internal Dataset:** A specialized dataset consisting of phrases from maintenance tasks. Each phrase typically

[1] https://github.com/OpenNMT/CTranslate2/

TABLE I
COMPARISON OF DIFFERENT DRAFT MODELS IN SPECULATIVE DECODING. THE TABLE HIGHLIGHTS DIFFERENCES IN SPEEDUP (S), ACCEPTANCE RATE (A_r), AND AVERAGE ACCEPTANCE LENGTH (A_l) ACROSS DATASETS.

Method	Main Model	Draft Model	CI-AVSR Dataset			Internal Dataset		
			S	A_r	A_l	S	A_r	A_l
Baseline	Whisper-large-v3	Whisper-turbo-v3	1.02	44.1	3.46	1.27	85.8	8.41
Proposed method	Whisper-large-v3	Token map	**1.27**	38.6	3.42	**1.37**	85.6	**3.92**
	Whisper-medium	Token map	1.12	**56.0**	2.32	1.36	95.9	3.95
	Whisper-small	Token map	1.09	51.1	**1.83**	1.35	**97.1**	3.96

includes a device or part name followed by measurement values at the end of the phrase, simulating real-world structured speech input.

As a baseline method, we compare with the Distill-spec method [15] using the Whisper-large-v3 [2] along with Whisper-large-v3-turbo [3] as a draft model, as Distil-Whisper-large-v3 only supports the English language. For our proposed method, we use different Whisper variants, including Whisper-large-v3, Whisper-medium, and Whisper-small to evaluate the performance.

B. Evaluation metrics

To assess the effectiveness of speculative decoding, we use the following benchmark metrics:

- **Speedup (S):** Measures the relative improvement in decoding time compared to baseline autoregressive decoding.

$$S = \frac{T_{\text{baseline}}}{T_{\text{speculative}}} \tag{3}$$

where T_{baseline} is the decoding time using standard autoregressive inference and $T_{\text{speculative}}$ is the decoding time using speculative decoding.

- **Acceptance rate (A_r):** The proportion of speculative tokens accepted by the main model:

$$A_r = \frac{N_{\text{accepted}}}{N_{\text{total}}} \tag{4}$$

where N_{accepted} is the number of speculative tokens accepted, and N_{total} is the total number of speculative tokens proposed.

- **Average acceptance length (A_l):** The average number of consecutive speculative tokens accepted before a correction is required:

$$A_l = \frac{\sum L_{\text{accepted}}}{N_{\text{sequences}}} \tag{5}$$

where L_{accepted} is the number of speculative tokens accepted in a sequence, and $N_{\text{sequences}}$ is the total number of decoded sequences.

[2]https://huggingface.co/openai/whisper-large-v3

[3]https://huggingface.co/openai/whisper-large-v3-turbo

C. Results

Table I compares the performance of different draft models in speculative decoding, highlighting speedup (S), acceptance rate (A_r), and average acceptance length (A_l) across two datasets. The proposed token map method consistently outperforms the Whisper-turbo draft model in terms of speedup, particularly on the Internal Dataset, where it achieves up to 1.37× speedup compared to 1.27× for the baseline. This suggests that the token map method is more effective at accelerating inference in structured domain-specific transcriptions.

Despite its efficiency, the token map method exhibits a lower acceptance rate (A_r) on the CI-AVSR dataset, which may indicate challenges in adapting to more diverse or spontaneous speech patterns. The lower average acceptance length (A_l) compared to the Whisper-turbo model on the Internal Dataset suggests that speculative token predictions are more conservative, possibly reducing the risk of incorrect acceptance. Overall, these results demonstrate that our proposed model-free approach achieves competitive or superior performance without requiring an additional draft model, making it well-suited for deployment on resource-constrained devices.

V. CONCLUSION

This paper introduced Token Map Drafting, a model-free speculative decoding technique designed to accelerate autoregressive ASR inference in structured, low-perplexity domains. By utilizing a precomputed n-gram token map instead of a separate draft model, our approach significantly reduces computational overhead, achieving up to 1.37× speedup on a domain-specific dataset and demonstrating a 10% absolute improvement over the distill-spec baseline on CPU-based systems. These results highlight the method's potential for real-time, on-device ASR applications, particularly in resource-constrained environments.

While our experimental results suggest that the proposed method maintains transcription accuracy, we recognize that the accuracy of the main model plays a critical role in decoding efficiency, particularly under conditions where the main model may introduce recognition errors which cause inconsistency with the Token Map. Future work will investigate the relationship between main model accuracy and decoding performance to better understand these dynamics and further optimize our approach.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International conference on machine learning*. PMLR, 2023, pp. 28 492–28 518.

[3] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Procs. of INTERSPEECH*, 2020.

[4] J. Ao, R. Wang, L. Zhou, C. Wang, S. Ren, Y. Wu, S. Liu, T. Ko, Q. Li, Y. Zhang, Z. Wei, Y. Qian, J. Li, and F. Wei, "SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2022, pp. 5723–5738.

[5] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in *International Conference on Learning Representations (ICLR)*, 2024.

[7] Y. Li, L. Lai, Y. Shangguan, F. N. Iandola, Z. Ni, E. Chang, Y. Shi, and V. Chandra, "Folding Attention: Memory and Power Optimization for On-Device Transformer-based Streaming Speech Recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 901–11 905.

[8] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in neural information processing systems*, vol. 35, pp. 30 318–30 332, 2022.

[9] O. Rybakov, P. Meadowlark, S. Ding, D. Qiu, J. Li, D. Rim, and Y. He, "2-bit Conformer quantization for automatic speech recognition," in *Proc. of INTERSPEECH*, 2023.

[10] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in neural information processing systems*, vol. 33, pp. 5776–5788, 2020.

[11] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020.

[12] 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. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 19 274–19 286. [Online]. Available: https://proceedings.mlr.press/v202/leviathan23a.html

[13] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, and L. Sifre, "Accelerating large language model decoding with speculative sampling," *arXiv preprint arXiv:2302.01318*, 2023.

[14] X. Liu, L. Hu, P. Bailis, A. Cheung, Z. Deng, I. Stoica, and H. Zhang, "Online speculative decoding," in *Proc. of the 41st International Conference on Machine Learning*. JMLR.org, 2024.

[15] S. Gandhi, P. von Platen, and A. M. Rush, "Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling," *arXiv preprint arXiv:2311.00430*, 2023.

[16] Y. Segal-Feldman, A. Shamsian, A. Navon, G. Hetz, and J. Keshet, "Whisper in Medusa's Ear: Multi-head Efficient Decoding for Transformer-based ASR," *arXiv preprint arXiv:2409.15869*, 2024.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[18] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," *Annual Meeting of the Association for Computational Linguistics*, pp. 993–1003, 2021.

[19] W. Dai, S. Cahyawijaya, T. Yu, E. J. Barezi, P. Xu, C. T. S. Yiu, R. Frieske, H. Lovenia, G. I. Winata, Q. Chen, X. Ma, B. E. Shi, and P. Fung, "CI-AVSR: A Cantonese Audio-Visual Speech Dataset for In-car Command Recognition," *ArXiv*, vol. abs/2201.03804, 2022.