Adaptive Beam Search: Enhancing Sequence Prediction with Delayed Pruning

Basit Ali Tramboo[1], **Meet Maratha**[1]

[1]Faculty of Computer Science
Dalhousie University, 6050 University Ave,
Halifax, Canada. NS B3H 4H9

Abstract

The beam search is a cornerstone of modern sequence generation, but often suffers from premature pruning of candidate sequences, discarding paths that may later prove optimal. In this paper, we introduce Adaptive Beam Search with Delayed Pruning, a simple yet effective modification that defers the removal of low-probability hypotheses by a fixed 'prediction window' of size W. Our algorithm interposes a delay of W decoding steps before eliminating the least likely sequence at each iteration, allowing initially weak candidates more opportunity to recover. We systematically explore the impact of varying window sizes on both custom-trained and off-the-shelf pre-trained language models (DistilGPT, GPT-2, GPT-Medium), evaluating performance with SacreBLEU, ROUGE-L, BERTScore, and MAUVE metrics. Experiments demonstrate that a modest delay (window sizes of 2–3) consistently yields higher quality outputs while larger windows incur diminishing returns. The method increases computational cost and implementation complexity for generating these results. Our findings suggest that delayed pruning can serve as an enhancement for applications demanding higher sequence-prediction accuracy, offering a tunable trade-off between search breadth and efficiency.

Introduction

Sequence generation tasks such as machine translation, text summarization, and conversational AI have witnessed remarkable progress with the advent of encoder–decoder architectures, especially Transformers (Vaswani et al. 2023). The Transformer model's self-attention mechanism enables highly parallelizable training and superior performance, achieving state-of-the-art results on benchmarks like WMT 2014 English–German and English–French translation (Vaswani et al. 2023).

Decoding from these models requires effective search strategies to traverse the exponential space of possible output sequences. Greedy search, which selects the single most probable token at each step, often leads to suboptimal sequences by committing too early to high-probability tokens. In contrast, beam search maintains a fixed number of hypotheses ("beam width") at each time step, balancing exploration and exploitation to find globally better sequences (HuggingFace 2022).

Despite its widespread use, beam search suffers from critical limitations. Empirical analyses reveal that larger beam widths can paradoxically degrade translation quality, a phenomenon linked to premature pruning of sequences that initially appear weak but later recover to yield higher overall probability (Cohen and Beck 2019). Furthermore, exact inference studies show that standard beam search often fails to locate the global best-scoring hypothesis, even with very large beams, exposing inherent search errors in the algorithm (Stahlberg and Byrne 2019).

Recent work has begun to explore extensions of beam search that incorporate deeper lookahead or heuristic refinements to address these shortcomings. Lookahead Beam Search, for instance, optimizes over a fixed number of future decoding steps and has been shown to outperform vanilla beam search on machine translation tasks (Jinnai, Morimura, and Honda 2023). However, such approaches typically introduce substantial computational overhead or require complex approximations.

In this paper, we introduce Adaptive Beam Search with Delayed Pruning, a simple modification to the classic beam search algorithm that defers the elimination of low-probability hypotheses by a fixed "prediction window" of size W. By delaying pruning decisions, our method allows initially weak candidates additional context to recover before being discarded. We empirically evaluate this approach on both custom-trained and pre-trained language models, demonstrating consistent improvements in SacreBLEU (Post 2018), ROUGE-L (Lin 2004), BERTScore (Zhang* et al. 2020), and MAUVE (Pillutla et al. 2021) metrics with modest window sizes, while preserving the core efficiency of beam search.

Problem of Study

Beam search is a widely adopted decoding algorithm for sequence prediction tasks such as machine translation, text generation, and speech recognition. Despite its effectiveness, a fundamental limitation lies in its **aggressive and premature pruning strategy**.At each decoding step, beam search retains only the top-K hypotheses based on current scores, **eliminating alternative sequences that may initially appear suboptimal but could evolve into higher-quality outputs as more context becomes available**.

This rigid, step-wise pruning mechanism **fails to ac-**

commodate the dynamic nature of sequence generation, where the relative probability of candidate sequences can shift significantly with the progression of decoding. As a result, beam search can produces **suboptimal final sequences**, articularly in cases where delayed contextual understanding is crucial.

To address this issue, we propose an **Adaptive Beam Search with Delayed Pruning**, which introduces a prediction window mechanism. Instead of pruning at every step, our approach postpones pruning decisions for a fixed or adaptive number of decoding steps, thereby allowing more candidate sequences to develop before elimination. This delay enables the decoder to make **better-informed pruning decisions**, reducing the likelihood of discarding potentially optimal sequences too early.

By incorporating temporal flexibility into the pruning strategy, the proposed method aims to enhance the diversity and quality of generated sequences, offering a robust alternative to the fixed pruning schedule of traditional beam search.

Related Work

Recent efforts have focused on improving beam search by either enhancing search depth or making pruning decisions more adaptive. Jinnai et al. (Jinnai, Morimura, and Honda 2023) introduced *Lookahead Beam Search (LBS)*, which considers future steps before pruning, enabling deeper and more informed search in text generation. Their results highlight the benefit of delaying pruning decisions, which aligns with our approach.

The concept of adaptive pruning has also been explored in model compression. Zhao et al. (Zhao, Jain, and Zhao 2022) proposed an *Adaptive Activation-based Structured Pruning* method that dynamically removes less important components, showing that adaptive mechanisms can optimize both performance and efficiency.

Meister et al. (Meister, Vieira, and Cotterell 2020) explored memory-efficient variants of beam search, emphasizing the trade-off between search quality and computational cost. Wang et al. (Wang et al. 2025) proposed *Adapt-Pruner*, an adaptive structural pruning strategy for small language models, demonstrating the value of flexibility in model behavior.

Building on these ideas, our method introduces a prediction window during decoding to delay pruning decisions. This mechanism allows candidate sequences more time to evolve, reducing premature eliminations and improving the quality of generated sequences.

Methodology

We propose an extension to standard beam search, termed *Adaptive Beam Search with Delayed Pruning*. The goal is to reduce premature pruning by allowing bottom-ranked candidate sequences a short window to develop further before being discarded. This is achieved through a window decay mechanism tied to candidate rank. Below, we outline the algorithm in detail.

Step 1: Standard Beam Expansion

The decoding begins with standard beam search. At time step $t = 0$, the model generates K top candidate sequences (e.g., from the start-of-sequence token). For example, with $K = 2$, suppose the decoder outputs initial tokens “he” and “I”.

At the next step, each of these beams is expanded into their top predictions. Let us say: - “I” expands to “I am” and “I was”, - “he” expands to “he is” and “he were”.

This results in four new candidate sequences: “I am”, “I was”, “he is”, and “he were”.

Step 2: Assigning Prediction Windows

Instead of pruning immediately, we introduce a prediction window W (e.g., $W = 2$) for each candidate.

- Candidates in the top-K ranked by cumulative log-probability are assigned a fixed prediction window W. - Candidates outside the top-K (i.e., bottom-ranked) are assigned a decaying window, updated to $W - 1$.

These windows are stored per beam and updated at each time step.

Step 3: Window Decay and Conditional Pruning

At every subsequent decoding step, beams are expanded as usual. For each candidate: - If its window is greater than zero, it is retained, and its window is decremented by 1. - If its window reaches zero and it still falls outside the top-K, it is pruned (discarded). - Beams that remain in the top-K retain their full window length W.

This mechanism allows weaker candidates a temporary grace period to improve while ensuring that beams are eventually pruned based on informed context.

Step 4: Continuation Until Sequence Completion

The process of beam expansion, window assignment/decay, and conditional pruning continues until: - All beams generate the end-of-sequence (EOS) token, or - A predefined maximum sequence length is reached.

At termination, the candidate with the highest cumulative score is selected as the final output.

Window Size Selection

The prediction window size W plays a critical role in balancing computational efficiency and output quality. A larger window allows weaker candidates more time to evolve but increases memory and computation requirements, while a smaller window may revert behavior closer to standard beam search.

To evaluate the impact of W on decoding performance, we empirically tested multiple window sizes across various text generation tasks. Our results indicate that using a prediction window of size $W = 2$ or $W = 3$ consistently yields the best trade-off between quality and efficiency. Larger values of W showed diminishing returns, while smaller values risked premature pruning.

These findings suggest that a moderate prediction window offers sufficient lookahead to retain promising candidates without incurring significant computational overhead.

This delayed pruning strategy allows the decoder to make better-informed decisions by leveraging additional context before committing to the pruning operation, resulting in more coherent and higher-quality output sequences.

Experiments and Results

To evaluate our delayed-pruning beam search, we ran three experiments. In the first experiment we applied the modified beam search to a model trained from scratch, replacing the standard beam search step with our delayed-pruning window and measuring the change in sequence quality. In the second experiment we plugged the same delayed-pruning logic into a suite of off-the-shelf pre-trained language models (DistilGPT, GPT-2, GPT-Medium) and, instead of just BLEU, compared performance across SacreBLEU, ROUGE-L, BERTScore, and MAUVE. In the third experiment we kept the model fixed (both scratch and pre-trained), while varying the pruning delay W from 1 to 4 steps to see how the window size hyperparameter impacts both output quality and decoding efficiency.

Experiment 1: Scratch Model

In this experiment, we developed a lightweight sequence-to-sequence model from scratch to evaluate the efficacy of our proposed adaptive beam search with delayed pruning. Our model follows a conventional encoder–decoder architecture: both the encoder and decoder incorporate an embedding layer followed by a Long Short-Term Memory (LSTM) unit, and the decoder is equipped with an additional fully connected layer to predict the next word in the vocabulary. The task for this model was to translate German sentences into English using the WMT14 de-en dataset (Bojar et al. 2014).

We selected this minimal model design to ensure that the effects of our adaptive search method could be observed without the confounding influence of overly complex architectures. To keep training efficient, we trained the model on a subset of 100,000 examples for 30 epochs and evaluated its performance on the complete validation set. The best-performing model—determined by validation performance—was then used for our decoding experiments.

For evaluation, we employed the SacreBLEU score, which quantifies the overlap between the predicted translation and the reference translation while accounting for token frequency. We compared two decoding strategies on the validation set: the standard k-beam search with a fixed beam size of 5, and our proposed method, which utilizes a delayed pruning window of size 3. The standard beam search achieved a SacreBLEU score of 0.66, whereas our adaptive method yielded a score of 0.67. The comparative chart of the models can be viewed in Fig. 1 Although the improvement in accuracy was marginal, these results demonstrate that our method maintains competitive translation quality as seen by the increase in score on the same data with the same model.

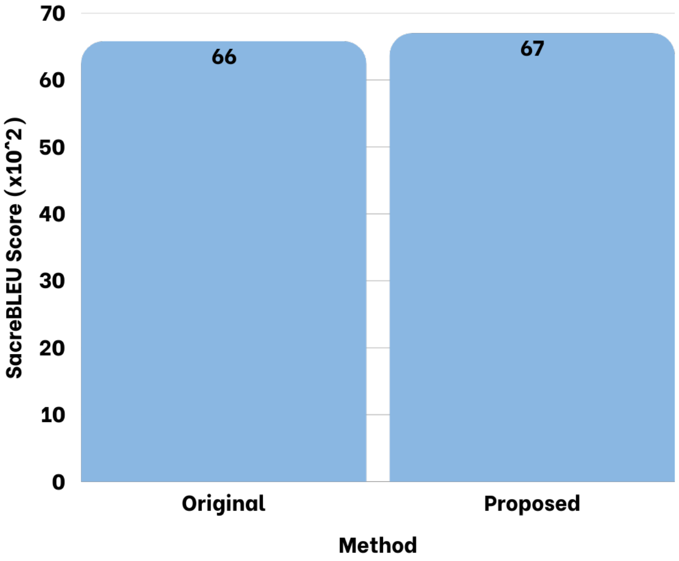

Figure 1: Chart comparing scaled SacreBLEU scores of proposed vs. original k-beams method on custom models

Experiment 2: Pre-trained models with different metrics

We evaluated three publicly available, autoregressive language models spanning a broad range of capacities—DistilGPT2 (88.2 million parameters) (Sanh et al. 2019), GPT-2 (137 million parameters) (Radford et al. 2019a), and GPT-2 Medium (380 million parameters) (Radford et al. 2019b) — on a next-token generation task drawn from the WikiText-103 benchmark, which comprises over 100 million tokens from high-quality Wikipedia articles (Merity et al. 2016). For each model, we randomly sampled 2,000 passages from the evaluation dataset, provided the first 10 tokens as a prompt, and generated the subsequent 50 tokens under both standard k-beam search (where beam width is 5) and our delayed-pruning method (where window size is 3). We then compared their outputs using four metrics: SacreBLEU (Post 2018), ROUGE-L (Lin 2004), BERTScore (Zhang* et al. 2020), and MAUVE (Pillutla et al. 2021).

SacreBLEU As we had explained the SacreBLEU metric in the above section, we used it over here to evaluate the accuracy of the three selected models on the selected task. As all of the models are mainly used for story generation, which is generation or completion of a story based on some input prompt, we expected them to not perform well in this metric. The reason behind this hypothesis was because the SacreBLEU score basically checks the occurrence of tokens in the predicted sequence that are present in the target sequence (Post 2018), and as these models are trying to generate a story, it is very likely that they generate tokens that are not at all present in the target sequence. This hypothesis was supported by our results, which we can see in Fig. 2, where we see that all of the models performed poorly in this evaluation.

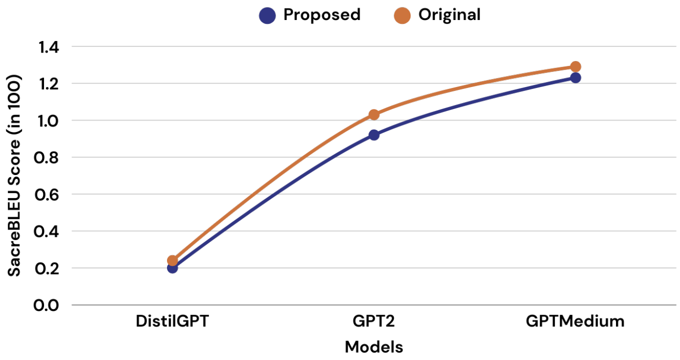

Figure 2: Chart comparing scaled SacreBLEU scores of proposed vs. original k-beams method on pre-trained models

We also notice that the original k-beams method performed better than the proposed method slightly by around 0.02 points. We expected this as the original k-beams selects the best option for each next token, whereas our method is trying to explore more options. So most of the tokens that will match between the target and predicted sequence would be some of the initial tokens, and normal k-beams will have more such tokens as it is exploring less in comparison to our proposed method.

ROUGE-L Next, we used the ROUGE-L metric to evaluate the accuracy of the three selected models on the selected task. The ROUGE-L score basically checks the Longest Common Subsequence (LCS) between the target and predicted sequences. The higher the length of this subsequence the higher the score(Lin 2004). As all of the models are mainly used for story generation, which is generation or completion of a story based on some input prompt, we expected them to not perform well in this metric. The reason behind this hypothesis is similar to the one we thought of in the SacreBLEU metric, where most of the tokens that will match will mostly be the first few tokens. This hypothesis was supported by our results, which we can see in Fig. 3, where we see that all of the models performed in the range of 6-12% accuracy in this evaluation.

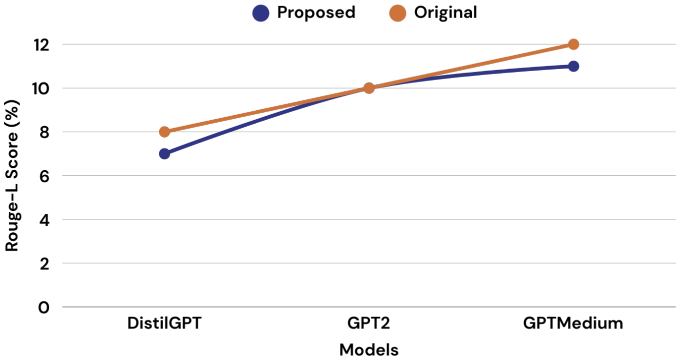

Figure 3: Chart comparing ROUGE-L scores of proposed vs. original k-beams method on pre-trained models

We also notice that the original k-beams method performed better than the proposed method on the smallest and largest model i.e. DistilGPT and GPT-2 Medium respectively. Whereas, our proposed model matches the original k-beams in evaluation score for the GPT-2 model. The scores were supposed to be low based on the same hypothesis as the one provided in SacreBLEU, but we see that as the metric starts becoming more relevant to the data points and the task of the model.

BertScore Next, we used the BertScore metric to evaluate the accuracy of the three selected models on the selected task. The BertScore basically checks the semantic similarity between the target and predicted sequences. The higher the similarity between the sequences the higher the score (Zhang* et al. 2020). As all of the models are mainly used for story generation, which is generation or completion of a story based on some input prompt, we expected them to perform somewhat well in this metric. The reason behind this hypothesis is based on the idea that whatever the story these models will generate will be somewhat related to the input prompt, which will make it highly close to the target sequence. This hypothesis was supported by our results, which we can see in Fig. 4, where we see that all of the models performed very well with scores in the range of 80% accuracy in this evaluation.

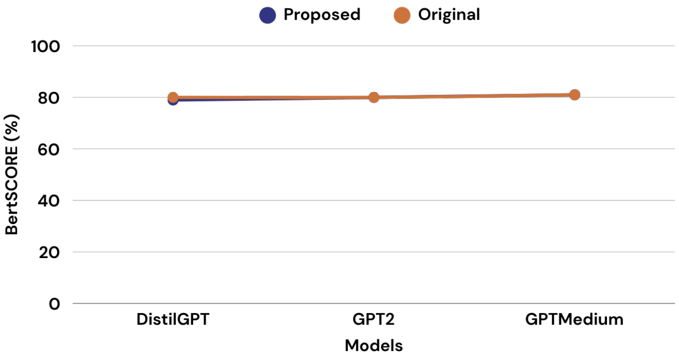

Figure 4: Chart comparing BertScore of proposed vs. original k-beams method on pre-trained models

We also notice that the original k-beams method perform almost identical to our proposed method. We somewhat expected this as both the sequences generated in original and the proposed method will be somewhat related to the target sequence due to input prompt, so they will have somewhat high scores in general. But them having almost similar score was something that shocked us, showing that even when we were exploring different sequences we were not going out of context for the predictions.

Mauve Score Finally, we used the Mauve metric to evaluate the accuracy of the three selected models on the selected task. The Mauve score basically checks the if the two sequences are similar in their ”feel” and ”style” of the text. It is majorly used for models like the ones we are testing, where a language model writes creative stories based on some input prompt. You can use Mauve to check if the generated stories have a similar flow and diversity as a collection of human-written stories. A higher Mauve score means the generated text is more similar in quality to the human-written examples (Pillutla et al. 2021). As all of the models are mainly used for story generation, which is generation or completion of a story based on some input prompt, we expected them to perform somewhat well in this metric. The reason behind this hypothesis is based on the idea that all of these models were industry standard at one point and they used this metric to show their accuracy. This hypothesis was supported by our results, which we can see in Fig. 5, where we see that all of the models performed very well with scores in the range of 20% accuracy in this evaluation, which is very good for models that are kind-of old.

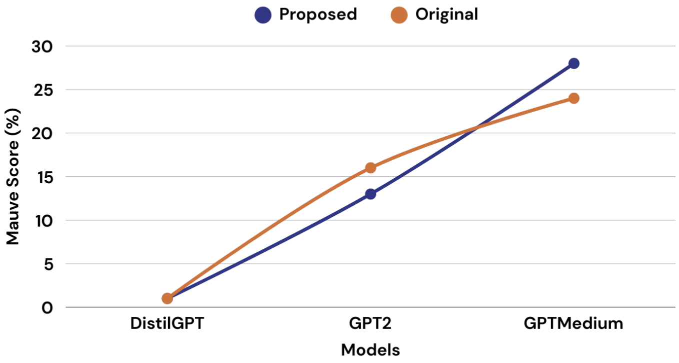

Figure 5: Chart comparing Mauve score of proposed vs. original k-beams method on pre-trained models

We also notice that the original k-beams method performed similar to our proposed method on the smaller model, better on the GPT-2 model, whereas our proposed method out-preformed original k-beams by a huge margin in the high-tier GPT-Medium model. We noticed an improvement of 4% in the accuracy of Mauve score of this mode in comparison to the original k-beams model. This showed us first proper signs of our proposed method giving better results in comparison to the original k-beams.

Experiment 3: Varying Window Sizes

In this experiment we used the scratch model we trained and the pre-trained GPT-2 medium model to evaluate any increase or decrease in the accuracy of the model for the required task based on the window size of the proposed method. To perform this evaluation we used the models and ran each of them with a constant beam width of 5, but a varying window size from 1 to 4. When the window size is 1, it is basically like a the original k-beams method where we are just interested in the immediate log probabilities, so to save some time we started our evaluation from the window size of 2.

Scratch Model In this experiment we used the scratch model we trained and tried to evaluate it with varying window size from 1 to 4, with a constant beam width of 5, for the task of German to English sentence translation. As we did before we are using the SacreBLEU score as the evaluation metric for this task.

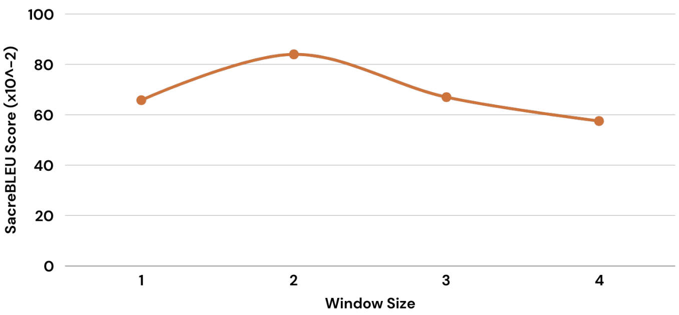

Figure 6: Chart comparing scaled SacreBLEU score of proposed vs. original k-beams method on the scratch model with varying window sizes

The results of this evaluation can be seen in Fig. 6. In it notice that our proposed method does out-perform the original k-beams method by a significant amount when the window size is 2. But we also notice that at the window sizes away from 2, the accuracy drops significantly. We have tried to address and provide a hypothesis for it in the discussion section of this paper.

Pre-trained Model In this experiment we used the pre-trained GPT-2 medium model and evaluate it with varying window size from 1 to 4, with a constant beam width of 5, for the task of story generation from an input that is a short subsequence of the original text. As we got the best improvement for the this model with the Mauve score, we are using it over here for the purpose of evaluation.

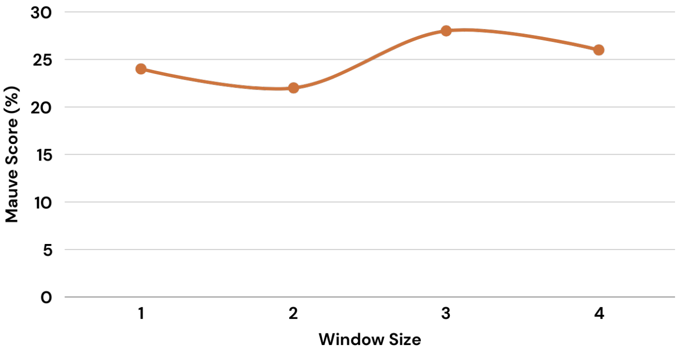

Figure 7: Chart comparing Mauve score of proposed vs. original k-beams method on the pre-trained GPT-2 medium model with varying window sizes

The results of this evaluation can be seen in Fig. 7. In it notice that our proposed method does out-perform the original k-beams method by a significant amount when the window size is 3. But again we notice that at the window sizes away from 3, the accuracy drops significantly. We have tried to address and provide a hypothesis for it in the discussion section of this paper.

Discussion and Limitations

Distinction Between Beam Width (K) and Prediction Window (W):

In adaptive beam search with delayed pruning, two key parameters govern the decoding process: the beam width (K) and the prediction window (W). While they both influence the quality of sequence generation, they serve fundamentally different purposes and should not be conflated.

The beam width (K) controls the number of candidate sequences retained at each decoding step. A higher K allows the algorithm to explore a broader search space, potentially improving performance but at the cost of increased computational complexity. Conversely, a smaller K reduces memory and computation requirements but may lead to missing optimal sequences due to narrow exploration.

In contrast, the prediction window (W) introduced in our method dictates the temporal delay in pruning decisions. Rather than pruning after each decoding step, our approach allows sequences to evolve for W steps before pruning occurs. This temporal buffer enables weakly scoring candidates to develop and potentially outperform early front-runners, especially when later context clarifies their relevance.

While K determines the breadth of the search at any given moment, W governs the depth of lookahead before making pruning decisions. Increasing K adds more beams; increasing W gives those beams more time to prove their potential.

To draw a simple analogy:

> K is how many candidates you allow in the room; W is how long you observe them before deciding who stays.

This distinction is essential for understanding the contribution of delayed pruning. Without it, one might incorrectly assume that increasing W has the same effect as increasing K, which is not the case.

Efficiency Considerations with Larger Prediction Windows

In our method, the prediction window size W controls how long lower-ranked beams are retained before being pruned. While increasing W allows weaker sequences to develop more context before elimination, we observe that efficiency degrades as W increases. This observation aligns with findings in prior work such as Lookahead Beam Search (LBS) by Jinnai et al. (Jinnai, Morimura, and Honda 2023), which investigates the trade-offs between shallow and deep search in text generation.

Exponential Increase in Computational Cost (Jinnai, Morimura, and Honda 2023) demonstrate that deeper lookahead in beam search leads to exponentially higher computational cost, while offering only marginal improvements in output quality beyond a certain depth. Our delayed pruning mechanism, which retains more candidate sequences for W steps, similarly increases the effective search depth. As W grows, so does the number of active beams, leading to higher inference time and memory usage.

Instability in Beam Ranking Another insight from LBS is that beam quality can fluctuate across decoding steps. In our case, beams retained longer due to a large W may not actually recover in rank, contributing noise to the search process. This rank instability introduces inefficiency, as it requires computational resources without improving the final outcome.

Diminishing Marginal Utility While moderate values of W (e.g., $W = 2$ or $W = 3$) offer a good balance between pruning robustness and runtime, we find that larger windows provide diminishing returns. Most candidate sequences that ultimately succeed tend to improve within the first few steps. Beyond that, additional context does not meaningfully change their fate, making continued evaluation unnecessary.

Trade-Off with Resource Usage Our empirical findings echo LBS: greater depth or window size increases search quality only to a point. After that, the overhead in model inference, softmax computation, and beam management outweighs the benefits. Furthermore, batch decoding and GPU parallelism are negatively affected by retaining variable-length beam structures longer than needed.

Table 1: Efficiency impact of increasing prediction window size W.

Factor	**Effect of Large W (Our Work)**	**Similar Effect in LBS (Jinnai, Morimura, and Honda 2023)**
Memory Usage	Increases due to more retained beams	Grows with deeper lookahead
Computation per Step	Slower decoding	Slower due to multi-step rollouts
Rank Instability	Low-quality beams may persist	Beam scores fluctuate with lookahead depth
Marginal Utility	Diminishes beyond $W = 3$	Declines past 2–3 lookahead steps
Efficiency	Degrades with larger W	Lower throughput at high depth

These insights reinforce our design decision to keep W small (typically 2 or 3), where it provides the most benefit in preserving candidate diversity while keeping the decoding process tractable.

Conclusion and Future Work

In this work, we proposed *Adaptive Beam Search with Delayed Pruning*—a modification to the standard beam search algorithm that introduces a prediction window to delay pruning decisions. Our method allows initially weak sequences additional decoding steps to recover, improving output quality in tasks like translation and story generation. Through empirical evaluations on both scratch-trained and pre-trained models, we demonstrated consistent performance gains, particularly in metrics like MAUVE and BERTScore, while highlighting the trade-offs between prediction window size and computational efficiency.

Looking ahead, future research could explore adaptive, data-driven window sizing strategies rather than fixed values. Additionally, integrating this approach with more recent Transformer-based architectures and experimenting on multimodal tasks such as image captioning or video-to-text generation could further validate its utility. Finally, extending delayed pruning to non-autoregressive generation models presents an exciting direction for enhancing decoding robustness.

Role and Contribution

Both authors, **Basit Ali Tramboo** and **Meet Maratha**, have contributed equally to this work, sharing responsibilities across ideation, implementation, experimentation, and documentation. The project was born out of mutual interest in improving sequence generation methods and involved consistent collaboration at every stage.

Basit Ali Tramboo primarily focused on the design and development of the delayed pruning mechanism within the beam search algorithm. He was primarily responsible for building the custom encoder–decoder model from scratch using LSTM-based architectures and implementing the adaptive pruning logic. He handled data preprocessing, training on the WMT14 dataset, and conducted the experimental evaluation using the SacreBLEU metric. His efforts also extended to performance tuning and visualizing results for the scratch-based model comparisons.

Meet Maratha led the integration of the proposed algorithm into pre-trained Transformer-based language models such as DistilGPT, GPT-2, and GPT-2 Medium. He orchestrated extensive evaluations on the WikiText-103 dataset using advanced metrics like ROUGE-L, BERTScore, and MAUVE, and was responsible for analyzing the impact of varying window sizes across model scales. He also managed benchmarking scripts, runtime efficiency tracking, and detailed metric-based visualizations.

References

Bojar, O.; Buck, C.; Federmann, C.; Haddow, B.; Koehn, P.; Leveling, J.; Monz, C.; Pecina, P.; Post, M.; Saint-Amand, H.; Soricut, R.; Specia, L.; and Tamchyna, A. s. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 12–58. Baltimore, Maryland, USA: Association for Computational Linguistics.

Cohen, E.; and Beck, C. 2019. Empirical Analysis of Beam Search Performance Degradation in Neural Sequence Models. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 1290–1299. PMLR.

HuggingFace. 2022. HuggingFace. https://huggingface.co/docs/transformers/v4.28.1/generation_strategies?utm_source=chatgpt.com. Accessed: 2025-03-30.

Jinnai, Y.; Morimura, T.; and Honda, U. 2023. On the Depth between Beam Search and Exhaustive Search for Text Generation. *arXiv preprint arXiv:2308.13696*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Meister, C.; Vieira, T.; and Cotterell, R. 2020. Best-First Beam Search. *arXiv preprint arXiv:2007.03909*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models. arXiv:1609.07843.

Pillutla, K.; Swayamdipta, S.; Zellers, R.; Thickstun, J.; Welleck, S.; Choi, Y.; and Harchaoui, Z. 2021. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. In *NeurIPS*.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, 186–191. Belgium, Brussels: Association for Computational Linguistics.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019a. Language Models are Unsupervised Multitask Learners.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC*$^2Workshop$.

Stahlberg, F.; and Byrne, B. 2019. On NMT Search Errors and Model Errors: Cat Got Your Tongue? arXiv:1908.10090.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2023. Attention Is All You Need. arXiv:1706.03762.

Wang, B.; Pan, R.; Diao, S.; Pan, X.; Zhang, J.; Pi, R.; and Zhang, T. 2025. Adapt-Pruner: Adaptive Structural Pruning for Efficient Small Language Model Training. *arXiv preprint arXiv:2502.03460*.

Zhang*, T.; Kishore*, V.; Wu*, F.; Weinberger, K. Q.; and Artzi, Y. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Zhao, K.; Jain, A.; and Zhao, M. 2022. Adaptive Activation-based Structured Pruning. *arXiv preprint arXiv:2201.10520*.