HARP: Hesitation-Aware Reframing in Transformer Inference Pass

Romain Storaï and Seung-won Hwang[*]
Computer Science and Engineering, Seoul National University
{romsto,seungwonh}@snu.ac.kr

Abstract

This paper aims to improve the performance of large language models by addressing the variable computational demands in inference steps, where some tokens require more computational resources than others. We present HARP, a simple modification to "off-the-shelf" Transformer forward pass. Drawing from hesitation and the framing effect in decision-making, HARP selectively applies additional computation when the model encounters uncertainty during token generation. Our method mimics human cognitive processes by pausing at difficult decision points and reframing inputs for a different perspective. Unlike other approaches, HARP is model-agnostic, training-free, and easy to implement. We evaluate our method across various downstream tasks and model sizes, demonstrating performance improvements up to +5.16%. Notably, HARP achieves these gains while maintaining inference times twice faster than beam search. Simple and yet with significant gains, HARP provides insights into the potential of adaptive computation for enhancing the performance of Transformer-based language models.

1 Introduction

Causal language models based on the Transformer architecture (Vaswani et al., 2017) use a constant number of layer traversals to generate each new token. While this architecture is beneficial to provide easy parallelization during training (Dehghani et al., 2019), it may not fully leverage the model's full potential during inference, where tokens are generated sequentially. Research in *adaptive computation* (*e.g.* Graves, 2017; Leviathan et al., 2023; Elhoushi et al., 2024; Leviathan et al., 2024) suggests that inference steps are not equally challenging, with some being "harder" and others "easier." Intuitively, these more challenging tokens would benefit from additional computational resources to improve accuracy. Unfortunately, the current Transformer architecture treats each token equally—regardless of its difficulty—potentially leading to imprecision and performance drops.

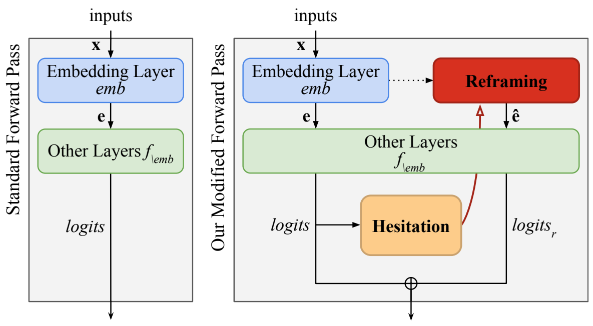


Figure 1: The left side represents the Transformer's vanilla forward pass, while the right side illustrates the modified forward pass, HARP, which selectively applies additional computation by reframing inputs when the model hesitates. This improves performance on "harder" tokens without the need for retraining.

To address this limitation, a trend in language models has been to simply scale up models in size (Brown et al., 2020), allowing for more computation. While the difficult tokens benefit from this scaling, it also leads to unnecessary overhead for easier tokens. To tackle this uniform additional computation, Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023) employs a bigger model to verify and correct the tokens generated by the smaller model, acting as an external verifier. Considering the smaller model as the original model, the larger model performs additional computations to ensure the quality of the generated tokens. This enables the system to spend more computational resources on complex tokens while preserving efficiency for easier ones. However, this approach requires the involvement of an external model. Similarly, Goyal et al. (2024) add fixed pauses during inference, using "pause tokens," to allow for additional computation on harder to-

[*] Corresponding author.

kens. While their method improves the generation of harder tokens, it requires full training and fine-tuning, and it still applies uniform additional computation when simpler tokens do not need the extra time.

We draw inspiration from human behaviors to allow models to perform additional computations for "harder" steps without relying on external models or requiring retraining. Two cognitive effects stand out: the (1) **hesitation** and the (2) **framing effect**. First, *hesitation* reflects uncertainty in decision-making. Humans tend to pause and reconsider when faced with difficult decisions (Shenhav et al., 2013) such that more effort is spent on complex inputs—aligning with the idea of "harder" tokens during inference. Second, the *framing effect* indicates that how information is presented can influence judgment and response (Kahneman, 2012). It implies that a different representation of the same input can lead to better outcomes. In the following, we will refer to this another-view representation as "*reframing*" the inputs.

Building on these human-inspired concepts, we introduce Hesitation-Aware Reframed Forward Pass (HARP), a plug-and-play modification to the Transformer's forward pass. To illustrate, we summarize HARP on the right side of Figure 1. We begin with the standard forward pass, processing the inputs through the embedding layer and subsequent layers to produce initial logits. We then evaluate the *hesitation* of the model by computing its uncertainty over the logits (detailed in Section 3.2). If the model is not hesitating, we directly output the initial logits. However, when the model is in a hesitation state, we *reframe* the inputs to infuse a different representation (explained in Section 3.3). To do so, we perturb the embeddings and perform a new forward pass. Finally, we combine the logits from the original and the additional forward pass, producing a final output incorporating both perspectives.

By selectively reframing inputs, HARP mimics human reconsideration under uncertainty, achieving up to 5.16% performance improvements with minimal additional cost. Our method outperforms beam search (Kasai et al., 2024) across multiple tasks, offering higher accuracy and significantly faster inference. The code[1] is publicly available.

Our contributions can be summarized as follows:

- We introduce a selection process based on token-level uncertainty to determine when additional computation is beneficial during inference and demonstrate its importance for improving model performance.
- We evaluate a novel approach to reframing inputs at inference time using dropout on the embeddings.
- We combine these two components to create HARP (Hesitation-Aware Reframed Forward Pass), a method that generally applies to any decoding algorithms, and evaluate its performance across various tasks, model sizes, and in combination with existing techniques.

[1] https://github.com/romsto/HARP

2 Related Works

2.1 Adaptive Computation in Transformers

Adaptive computation in Transformers (ACT) can be categorized into efficiency-focused and performance-focused categories. Efficiency-focused ACT has been the primary focus of research, aiming at improving efficiency by reducing computation for "easier" inference steps (Leviathan et al., 2023; Chen et al., 2023; Elhoushi et al., 2024). These approaches often involve using smaller models or skipping layers when processing less challenging tokens, thereby optimizing computational resources and resulting in inference speedups.

In contrast, performance-focused ACT, which includes our method HARP, targets the "harder" inference steps, prioritizing performance gains over efficiency improvements. Our approach shares motivations with the work of Goyal et al. (2024), who allocate more computation by extending the model's vocabulary with pause tokens. These prepend tokens to the generation instruct the model to perform additional fixed computations, resulting in higher accuracy. However, while effective, the pause tokens method requires retraining and fine-tuning of the model.

Unlike the pause tokens approach or scaled-up models using efficiency-focused ACT, HARP is a training-free, model-agnostic, and plug-and-play method that can be applied to any Transformer-based model without the need for retraining, making it an advantageous solution within the performance-focused ACT framework.

In parallel to adaptive computation work, some studies have explored methods to enhance reasoning capabilities in LLMs (*e.g.* Zelikman et al., 2022; Zelikman et al., 2024; Hosseini et al., 2024; Andukuri et al., 2024). While these approaches can be seen as incorporating extra computation, their focus diverges from ours, as they improve reasoning through fine-tuning rather than optimizing token-level computation selectively.

2.2 Uncertainty Estimation in Language Modeling

Uncertainty quantification (Abdar et al., 2020) on token-level is not a well-explored area. Most of the existing works focus on the evaluation of sequence-level uncertainty (*e.g.* Arteaga et al., 2024; Manakul et al., 2023; Kuhn et al., 2023) or on a higher level. In contrast, our work focuses on token-level uncertainty—the uncertainty in the probability distribution over the vocabulary for predicting the next token. Luo et al. (2024) introduce a ratio-based method to measure uncertainty. While this approach offers an intuitive interpretation of uncertainty, it lacks some theoretical grounding and might fail to capture more subtle hesitation as it relies on only the two highest probabilities of the distribution. Therefore, we use the Shannon Entropy (Shannon, 1948) as our uncertainty estimator. It is an information-theoretic uncertainty measure that captures the amount of information in a probability distribution. Entropy represents the expected number of bits required to resolve the uncertainty of a prediction. Higher entropy indicates more uncertainty, while lower entropy suggests a more confident prediction.

2.3 Reframing at Inference Time

Reframing data into different perspectives is a well-established technique for training machine learning models, particularly through Multi-View Learning (MVL) (Chaudhuri et al., 2009). In MVL, models are trained on multiple representations of the same data, which improves generalization. However, these techniques are restricted to the training phase and are not designed to be applied during inference (Xu et al., 2013), our target.

Parallely, NEFTune (Jain et al., 2024) brings a promising direction. It introduces noise into embeddings during the training to improve instruction-based fine-tuning. Although NEFTune targets the training phase only, we hypothesize that a similar approach—injecting noise into embeddings—could be beneficial during inference as well. By adding noise into embeddings at inference time, the model could gain a new perspective of the same inputs, potentially improving its ability to handle ambiguous inputs. While NEFTune uses random uniform noise, our work explores different noise approaches, utilizing dropout on the embeddings to induce a new representation. As detailed in Appendix A, we find that dropout leads to more consistent improvements.

3 Methods

In this section, we present our Hesitation-Aware Reframed Forward Pass (HARP) method. We aim to perform an additional forward step from a different perspective when the model encounters uncertainty. We begin by reviewing the standard forward pass of transformers. Then, we introduce the two key components of HARP: quantifying uncertainty during inference and reframing inputs. Finally, we will integrate these components to present the complete algorithm.

3.1 Preliminary: Transformer Forward Pass

We recall the forward pass of a Transformer model as we will modify its architecture. It processes input tokens to generate logits, representing unnormalized probabilities for predicting each token in the sequence, including the next token. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be the tokenized input sequence of length n, where each x_i is a token ID. For simplicity, we denote the embedding layer as $emb(\cdot)$, while $f_{\setminus emb}(\cdot)$ represents the rest of the model, consisting of N serial layers. Each layer includes a multi-head self-attention sublayer, a fully connected sublayer, and layer normalizations. We deliberately omit the discussion of positional encodings and last-layer projection. First, the embedding layer emb maps each input token to a dense vector representation: $\mathbf{e} = emb(\mathbf{x})$, where $\mathbf{e} \in \mathbb{R}^{n \times d}$ and d is the embedding dimension. The embedded inputs are then processed through the rest of the model.

Thus, the forward pass can be concisely expressed as:

$$logits = f_{\setminus emb}(emb(\mathbf{x})) \quad (1)$$

where $logits \in \mathbb{R}^{n \times |V|}$ and $|V|$ is the vocabulary size. The resulting $logits$ contains unnormalized predictions for each input position. The last position's logits are used to predict the next token.

3.2 Uncertainty Estimation

We want to quantify the model's uncertainty for each newly generated token. To do this, we focus on the logits of the last position, which are used to predict the next token. First, the logits are normalized using the SOFTMAX function to obtain a probability distribution $\mathbb{P}$ over the vocabulary V.

The SOFTMAX ensures that each value in the distribution is between 0 and 1 and that the total sum of probabilities over V equals 1, *i.e.*, $\sum_{i=1}^{|V|} P(v_i \mid \mathbf{x}) = 1$, where v_i represents a token in the vocabulary V and $P(v_i \mid \mathbf{x})$ is the probability of token v_i, knowing $\mathbf{x}$, under the distribution $\mathbb{P}$.

To measure the uncertainty of the distribution $\mathbb{P}$, we then use Shannon entropy (Shannon, 1948). The entropy SHANNON is defined as:

$$\textsc{Shannon}(\mathbb{P}) = -\sum_{i=1}^{|V|} P(v_i \mid \mathbf{x}) \log_2 P(v_i \mid \mathbf{x}) \tag{2}$$

3.3 Reframing Inputs

Our objective is to perturb the embeddings, which represent the model's understanding of the data, to present the input sequence to the model from an alternate perspective.

We follow the approach of NEFTune (Jain et al., 2024), which injects random noise into the embeddings during training. NEFTune generates this noise by sampling values from a uniform distribution in $[-1, 1]$, then scaling the noise based on sequence length, embedding dimension, and a tunable parameter. To explore alternative perturbation methods, we experimented with various noise strategies. We choose to use dropout among them, as it consistently yields the best results. In Appendix A, we demonstrate that dropout performs better than NEFTune's random uniform noise in this context.

Let $\mathbf{e} = emb(\mathbf{x})$ be the original embeddings of the input sequence $\mathbf{x}$, and let δ be the dropout rate. The reframed embeddings $\hat{\mathbf{e}}$ are then obtained as follows:

$$\hat{\mathbf{e}} = \textsc{Dropout}(\mathbf{e}, \delta) \tag{3}$$

where DROPOUT randomly sets a fraction δ of the elements in the embeddings to zero.

3.4 Hesitation-Aware Reframed Forward Pass (HARP)

Our method, HARP, adapts the standard Transformer forward pass by adding an additional computation step when the model exhibits uncertainty. This procedure is outlined in Algorithm 1, using the same colors as in Figure 1 for clarity.

Algorithm 1 HARP: **H**esitation-**A**ware **R**eframed Forward **P**ass Step

Inputs: input sequence $\mathbf{x}$, embedding layer $emb(\cdot)$, rest of the model $f_{\setminus emb}(\cdot)$
Hyperparameters: uncertainty threshold θ, dropout rate δ, combination factor β

```
1:  $\mathbf{e} \leftarrow emb(\mathbf{x})$
2:  $logits \leftarrow f_{\setminus emb}(\mathbf{e})$
3:  $\mathbb{P} \leftarrow \textsc{Softmax}(logits)$
4:  ▷ If the model is uncertain, we perform one more forward pass but reframed.
5:  if $\textsc{Shannon}(\mathbb{P}) > \theta$ then
6:      $\hat{\mathbf{e}} \leftarrow \textsc{DropOut}(\mathbf{e}, \delta)$
7:      $logits_r \leftarrow f_{\setminus emb}(\hat{\mathbf{e}})$
8:      $logits \leftarrow \beta * logits + (1 - \beta) * logits_r$
9:  end if
10: return $logits$
```

First, we perform the standard Transformer forward pass (presented in Section 3.1). From the input sequence $\mathbf{x}$, we compute the token embeddings $\mathbf{e}$ and the logits, denoted as $logits$.

Next, we estimate the uncertainty at the current step using Shannon Entropy (detailed in Section 3.2). To achieve this, we normalize the logits and compute the Shannon entropy.

If the entropy is below a predefined uncertainty threshold θ, the model is considered confident in its generation. In this case, we return the logits from the standard forward pass, and the current generation step ends.

However, if the uncertainty exceeds the threshold θ, the model is uncertain. In this case, we initiate a second forward pass but with a reframed input sequence (explained in Section 3.3). The reframed embeddings, $\hat{\mathbf{e}}$, are obtained by applying dropout with a rate δ to the original embeddings $\mathbf{e}$. We then perform a second forward pass using $\hat{\mathbf{e}}$, which outputs reframed logits, denoted as $logits_r$. This additional forward pass requires recomputation as it cannot leverage the KVCache (see Section 8).

Finally, the original and reframed logits are combined using a combination factor β to produce the final output logits.

$$logits = \beta \cdot logits + (1 - \beta) \cdot logits_r \tag{4}$$

The combination factor β controls the balance between original and reframed logits. We empirically find that setting $\beta = 0.5$ balances the contributions of both passes effectively.

The dropout-based perturbation forces the model to consider different representations of the input. By selectively introducing additional computations only when necessary, HARP balances computational efficiency and prediction reliability.

Our method introduces three hyperparameters: the uncertainty threshold θ, the dropout rate δ, and the combination factor β. In our experiments, we set these parameters to values identified as optimal during preliminary testing. The optimal choices for these hyperparameters may vary across tasks or models and can benefit from further tuning; we provide a preliminary study on the impact of θ in Appendix C and leave a more in-depth exploration of all hyperparameters for future work.

4 Experimental Set-Up

4.1 Models

We consider decoder-only models of size 3.8B, 7B, and 8B as they are the standard in recent times. Particularly, we use LLaMA-3.1 (Dubey et al., 2024), Mistral 7B v0.3 (Jiang et al., 2023) and Phi 3.5 Mini (Abdin et al., 2024) to cover different scales and architectures. We consider only aligned models (denoted as "*Instruct*") for their simplicity of evaluation and greater performance. *Aligned models* (Christiano et al., 2017) refer to models that are fine-tuned to better follow instructions, using methods such as supervised fine-tuning or reinforcement learning from human feedback (RLHF). All the models are loaded using quantized INT8 precision to fit the GPU and accelerate inference.

4.2 Datasets

As our method targets “off-the-shelf” LLMs, we consider five datasets covering varied downstream tasks and output formats in order to reproduce the variety of missions they can encounter.

- **GSM8K** (Cobbe et al., 2021) is a famous mathematical reasoning benchmark where the task involves solving grade school-level math word problems. The output is a free text, typically a detailed solution or numerical answer.
- **CommonSenseQA (CsQA)** (Talmor et al., 2019) is a multiple-choice question benchmark that tests commonsense reasoning and general understanding abilities of the model. The output is one selected option from five possible choices.
- **LAMBADA** (Paperno et al., 2016), a language modeling benchmark focused on text understanding and completion, where the task is to predict the final word of a passage based on its context. The output is a single word.
- **MMLU Pro** (Wang et al., 2024)—an enhancement of Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) dataset—evaluates models on a wide range of academic and professional subjects. The task consists of answering multiple-choice questions, with the output being one selected option from up to ten possible choices.
- **CNN/Daily Mail (CNN/DM)** (Nallapati et al., 2016) dataset is used for text summarization tasks, where the goal is to generate concise summaries of news articles. The output is free text in the form of a summary.

Prompt details can be found in Appendix D. All the datasets are evaluated with a zero-shot prompt.

4.3 Evaluation

Since HARP is generally applicable to any decoding methods, we compare different decoding methods (Shi et al., 2024) both with and without the HARP forward pass modification. We use greedy search decoding—a deterministic method that fetches the highest probable token—and nucleus sampling search—a stochastic one that samples the next token (Shi et al., 2024). In addition, we evaluate beam search decoding (Kasai et al., 2024) without HARP as this decoding method usually results in more accurate results. The hyperparameters used in our experiments are fixed at a dropout rate of $\delta = 0.20$ and an uncertainty threshold of $\theta = 1.0$. We set the temperature for nucleus sampling to $\tau = 0.6$ and the top-p to 0.9. In the case of beam search, the number of beams is $b = 3$ with a top-k of 5. We chose to apply length-normalization (Wu et al., 2016) as we found it beneficial to tasks such as summarization and reasoning. This length penalty is fixed to $\alpha = 0.6$ for every datasets, as we seek to compare HARP with model- and task-agnostic methods. We empirically found the value by evaluating a model on a subset of three datasets (CNN/DM, GSM8K and LAMBADA). When evaluating Chain-of-Thought (Wei et al., 2024) prompting, we prepend “`Let's think step-by-step.`” to the generation of the model. We evaluate a subset of each dataset with a batch size of 1, without

Models	Methods	Datasets					Relative Cost overhead to Vanilla
		CsQA	GSM8K	LAMBADA	MMLU Pro	CNN/DM	
LLaMA-3.1 Instruct (8B)	Vanilla (Greedy)	78.79	76.88	30.86	46.42	32.44	
	Beam Search	79.29	76.38	31.35	48.21	33.17	x2.79
	Ours (Greedy)	**80.30** (+1.52)	**78.39** (+1.51)	**36.02** (+5.16)	**48.21** (+1.79)	**34.03** (+1.59)	**x1.16**
	Vanilla (Nucleus)	**79.80**	73.00	27.44	42.55	30.81	
	Ours (Nucleus)	79.29 (-0.51)	**74.00** (+1.00)	**31.38** (+3.94)	**43.45** (+0.90)	**32.38** (+1.57)	**x1.17**
Mistral v0.3 Instruct (7.25B)	Vanilla (Greedy)	70.37	43.62	45.15	31.76	29.11	
	Beam Search	**70.99**	**50.53**	45.70	**33.11**	28.71	x3.07
	Ours (Greedy)	**70.99** (+0.62)	48.40 (+4.79)	**49.76** (+4.64)	31.76 (0.00)	**29.57** (+0.47)	**x1.24**
	Vanilla (Nucleus)	**70.74**	29.38	45.02	31.76	28.72	
	Ours (Nucleus)	70.21 (-0.53)	**31.88** (+2.50)	**48.26** (+3.24)	**33.79** (+2.03)	**28.72** (0.00)	**x1.29**
Phi 3.5 Mini Instruct (3.82B)	Vanilla (Greedy)	77.20	72.50	32.76	29.65	26.10	
	Beam Search	77.72	73.00	**33.60**	**33.78**	25.79	x3.18
	Ours (Greedy)	**78.24** (+1.04)	**73.00** (+0.50)	33.44 (+0.68)	32.75 (+3.10)	**26.97** (+0.87)	**x1.26**
	Vanilla (Nucleus)	77.04	71.50	31.85	28.82	**25.30**	
	Ours (Nucleus)	**77.55** (+0.51)	**74.50** (+3.00)	**32.99** (+1.14)	**34.53** (+5.71)	25.19 (-0.11)	**x1.27**

Table 1: Performance comparison of the original model (Vanilla) with various decoding methods: greedy search, nucleus sampling, beam search, and our HARP modified forward pass in both greedy and nucleus sampling settings. Numbers in parentheses indicate performance gain or loss relative to Vanilla for each decoding method. The cost column shows the relative inference time based on Vanilla's corresponding decoding method, averaged over five datasets. Reported scores reflect accuracy, except for CNN/DM, where the ROUGE-1 score is reported.

caching (KVCache), on a single RTX3090 (24GB) using the same seed for every example. The accuracy is reported for all datasets except CNN/DM, where we evaluate the ROUGE-1 score. Moreover, we record the generation time of each method.

5 Results

HARP improves the performance of all tasks. Our method demonstrates consistent performance improvements across all tasks. As shown in Table 1, in the greedy setting, HARP outperforms both the vanilla model and beam search decoding in most scenarios. The improvements are particularly notable for tasks like LAMBADA, with gains of up to 5.16%. When applied to nucleus sampling, HARP outperforms the vanilla model in most cases, especially in math-reasoning tasks like GSM8K.

On GSM8K, Goyal et al. (2024) achieved a +1.00% improvement (from 7.50% to 8.50% accuracy) with their training and fine-tuning pause token method. In comparison, HARP delivers an even higher improvement of +1.51% on the same dataset, using a more performant model and without any retraining.

In summary, our HARP allows further improvements as high as +5.16% of high-end models that already show strong performance on many tasks without requiring any fine-tuning. This demonstrates the potential of HARP to enhance state-of-the-art models.

HARP is model-agnostic. We show that HARP also consistently improves models ranging from 3 to 8 billion parameters, including LLaMA-3.1, Mistral v0.3, and Phi 3.5 Mini. It illustrates its robustness and wide applicability across diverse language models.

Models	Methods	Datasets		
		CsQA	GSM8K	MMLU Pro
LLaMA	CoT	74.95	75.48	48.19
	Ours CoT	**75.75** (+0.80)	**76.58** (+1.10)	**49.35** (+1.16)
Mistral	CoT	70.64	**48.00**	32.37
	Ours CoT	**75.23** (+4.59)	46.00 (-2.00)	**32.95** (+0.58)

Table 2: Accuracy comparison of Chain-of-Thought (CoT) and HARP applied to CoT, using greedy decoding. Numbers in parentheses indicate gain or loss relative to the standard CoT approach.

HARP works with advanced techniques. As shown in Table 2, advanced techniques, such as Chain-of-Thought (CoT) (Wei et al., 2024) prompting, are further enhanced with HARP. For instance, applying the modified forward pass to CoT prompting with the LLaMA model improves accuracy by 0.8%, 1.16%, and 1.10% on CommonsenseQA, MMLU Pro, and GSM8K, respectively. This shows our method can seamlessly combine with other existing methods, further enhancing performance.

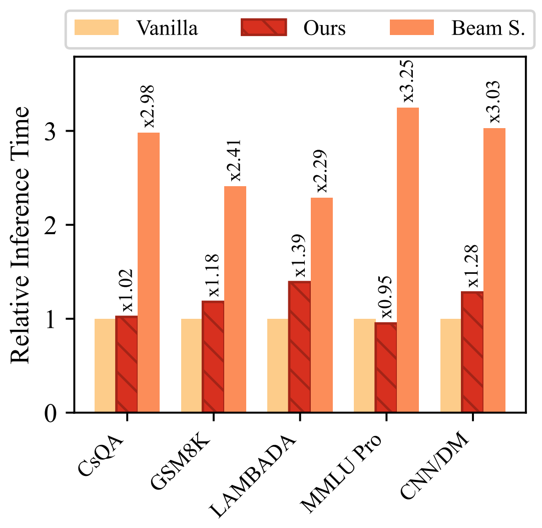


Figure 2: LLaMA 3.1 Instruct (8B) average relative inference time of the original model greedy search (Vanilla), beam search decoding (Beam S.), and our HARP (Ours) using greedy search decoding. Values and other models are detailed in Table 6.

HARP is faster than Beam Search. Figure 2 reveals that despite the additional computation, the inference time of HARP remains close to that of the vanilla generation, introducing a minimal additional computational cost. The inference time for our method is consistently under twice that of the original model without caching across all tasks. In contrast, beam search—while most of the time yielding lower performance than HARP—induces a significantly higher inference cost. This is the indicator that our method strikes a balance between performance gains and inference time.

Beam search shows significant delays, often higher than x2.4 the time required by the greedy search generation, whereas HARP achieves inference times lower than x1.4. Additional results in Table 6 validate this efficiency by comparing other models. For example, the Mistral model increases inference time by only x1.17 to x1.49, while beam search increases it by x2.51 to x3.72, compared to the original model without KVCache.

On average, our method results in an x1.25 increase in inference time across all tasks and models, demonstrating its efficiency compared to more computationally intensive approaches like beam search, which takes more than twice as long.

6 Analysis

Uncertainty guides the additional computations. To investigate the effectiveness of our uncertainty-based approach (denoted as 'w\ SHANNON'), we compare it with a method that unconditionally adds an extra forward step for every token (denoted as 'w\o SHANNON'). This comparison helps us understand whether the improvements come solely from the additional computation or from our uncertainty-selected approach. Table 3 presents the results across multiple tasks and two models.

Our analysis reveals that unconditionally adding an extra forward step is not universally beneficial. While it improves performance compared to the vanilla model in most cases, especially in tasks like GSM8K and LAMBADA, it also results in lower or equal performance on tasks like multiple-choice questions (CommonsenseQA and MMLU Pro).

The uncertainty-conditional method outperforms the unconditional approach, even though it is sometimes marginal. This suggests that uncertainty is an excellent signal for selecting the steps requiring extra computation. A key advantage of the selective approach is its computational efficiency. As shown in the Cost column of Table 3, the uncertainty-based method requires significantly lower computational overhead than the unconditional method.

While additional computation can generally improve performance, our uncertainty-based approach offers a more nuanced and efficient solution. It achieves comparable or superior results to unconditional computation while balancing performance gains and computational costs.

Uncertainty pinpoints key reasoning steps. We analyze the output generations of HARP by highlighting the tokens that require extra computation in Appendix B. Upon reviewing some examples, particularly in problem-solving tasks, we observe that high-uncertainty states usually appear at the start of each reasoning step. This mirrors human decision-making, where individuals take more time to consider the stages of reasoning to construct a valid response than the actual content of the reasoning.

Additionally, we note that HARP tends to generate shorter sequences than the original forward pass. Across every task (except ones with next-word prediction), we observe an average reduction of 5.5% in output length when applying HARP. This shortening effect is not caused by promoting the end-of-sequence (EOS) token as the most likely token earlier in uncertain cases. Instead, this effect appears from variations in token selection during earlier stages of generation. HARP results in more

Models	Methods	Datasets					Relative Cost
		CsQA	GSM8K	LAMBADA	MMLU Pro	CNN/DM	
LLaMA-3.1 Instruct (8B)	w\o SHANNON	77.17 (-1.62)	**78.39** (+1.51)	**36.08** (+5.22)	44.88 (-1.50)	32.66 (+0.22)	x2.31
	w\ SHANNON	**80.30** (+1.52)	**78.39** (+1.51)	36.02 (+5.16)	**48.21** (+1.79)	**34.03** (+1.59)	**x1**
Mistral v0.3 Instruct (7.25B)	w\o SHANNON	70.37 (0.00)	45.74 (+2.12)	49.56 (+4.41)	**31.76** (0.00)	28.62 (-0.49)	x1.68
	w\ SHANNON	**70.99** (+0.62)	**48.40** (+4.79)	**49.76** (+4.64)	**31.76** (0.00)	**29.57** (+0.47)	**x1**

Table 3: Study of the impact of unconditional additional step (w\o SHANNON) and HARP uncertainty-based additional step (w\ SHANNON) using greedy decoding. The cost column denotes the relative inference time with w\ SHANNON averaged over the five datasets. Numbers in parentheses indicate performance gain or loss relative to the original model performance.

Models	Methods	Datasets					Relative Cost
		CsQA	GSM8K	LAMBADA	MMLU Pro	CNN/DM	
LLaMA-3.1 Instruct (8B)	1-step	**80.30**	78.39	**36.02**	**48.21**	34.03	**x1**
	2-steps	79.80 (-0.50)	76.88 (-1.49)	35.52 (-0.50)	47.19 (-1.02)	33.92 (-0.11)	x1.52
	4-steps	79.70 (-0.60)	**80.40** (+2.01)	35.02 (-1.00)	43.73 (-4.48)	**34.29** (+0.26)	x1.97
Mistral v0.3 Instruct (7.25B)	1-step	**70.99**	**48.40**	**49.76**	**31.76**	29.57	**x1**
	2-steps	70.47 (-0.52)	46.38 (-2.02)	48.26 (-1.50)	**31.76** (0.00)	29.66 (+0.09)	x1.11
	4-steps	70.37 (-0.62)	45.32 (-3.08)	48.15 (-1.61)	**31.76** (0.00)	**29.83** (+0.26)	x1.37

Table 4: Performance comparison using a different number of reframing steps across all the datasets. "x-steps" indicates that the model can perform up to x additional forward passes when uncertainty exceeds the threshold. The cost column denotes the average relative inference time with HARP (1-step). Numbers in parentheses indicate performance gain or loss relative to HARP.

concise responses, especially for reasoning-based tasks and summarizing tasks.

One additional representation is sufficient for reframing. When facing uncertainty, HARP reframes the inputs only once before pursuing the next generation step. To explore the effect of multiple reframings, we experiment with adding extra forward passes while the uncertainty remains higher than θ (capping it at a maximum number of steps).

Surprisingly, Table 4 shows that increasing the number of additional steps often leads to a decline in performance. This interesting outcome suggests that while a single additional representation can provide a useful alternative perspective on the inputs, too many representations may be penalizing. In datasets like CommonsenseQA, LAMBADA, and MMLU Pro, we observe a consistent drop in accuracy with more reframing steps.

Although GSM8K achieves a notable +2.01% increase in accuracy using four additional steps with LLaMA, these gains come at the cost of significantly higher computation time, scaling up to x1.97 in the 4-step method.

We hypothesize that more than two representations might confuse and distract the model from the original task, decreasing precision. This aligns with cognitive theories, suggesting that while considering reframings enhances problem-solving, too many representations increase cognitive load (Sweller, 1988), reducing performance. In a more mathematical way, this behavior may be due to the random noise introduced during reframing and the way logits are combined. While random noise can sometimes be beneficial, it can also be detrimental, and increasing the number of reframings increases the likelihood of harmful noise. Additionally, as the logits from the vanilla and reframed representations are averaged (see Equation 4), introducing more than one reframing step reduces the weight of the vanilla logits, disproportionately favoring the reframed ones, contributing to the higher likelihood of harmful representations.

7 Conclusion

This paper presented a novel method, HARP, designed to enhance language model inference without requiring retraining or architectural modifications. Our approach uses a selection process based on token-level uncertainty to identify when additional computation is advantageous and an innovative method for reframing inputs during inference. We demonstrated that HARP can achieve significant performance improvements across various tasks and model sizes, with accuracy gains of up to 5.16%. Importantly, HARP maintains inference efficiency compared to methods like beam search, with only an x1.25 average increase in inference time over the standard model. While HARP provides a promising proof of concept, its real-world application is currently limited by challenges such as cache invalidity and the introduced randomness.

8 Limitations

Our study has several limitations that need to be addressed in future research.

First, due to resource and time constraints, our experiments were limited in scope. Evaluations were conducted on quantized models and subsets of each dataset, and we implemented custom generation methods. Additionally, the method has yet to be tested on larger language models, particularly those with 70 billion parameters or more. Although we attempt to cover a range of tasks, there are other language challenges that our method has not been tested on. Furthermore, our experiments did not leverage widely used libraries, such as LM-Evaluation-Harness (Gao et al., 2021), which could provide more standardized benchmarking. Consequently, while initial results demonstrate potential, further work is required to confirm whether HARP generalizes effectively across tasks and scales. HARP serves as a proof of concept, demonstrating the potential of uncertainty-aware adaptive computation in improving inference performance.

Furthermore, our current implementation faces some challenges in inference efficiency. The method could slow down batch processing since uncertain tokens require additional computation. Moreover, performance might be impacted when using HARP with models that employ key-value caching (KVCache). Embedding dropout may temporarily invalidate the KVCache, causing a VRAM spike during each reframing step. In the worst-case scenario—where all tokens are affected—VRAM usage could briefly double. Other approaches could be explored to mitigate this, such as directly perturbing the KVCache to perform reframing or doing layer-specific perturbations.

Looking forward, there are promising ideas for future work, in addition to exploring alternative uncertainty measures or perturbation methods. One intriguing direction would be to explore the application of the uncertainty selection mechanism during the fine-tuning process rather than only at inference time. This could involve integrating a "pause token," as proposed by Goyal et al. (2024), during training to teach the model when to allocate additional computation. Such an approach could enable models to increase computation when needed autonomously. Another direction could involve adapting beam search decoding with hesitation and reframing. Creating new beams when uncertainty is encountered and keeping track of all the branches may enhance the results even more. Finally, combining the proposed method with speculative decoding (Leviathan et al., 2023; Chen et al., 2023) could offer further optimizations, enabling gains in both efficiency and performance.

9 Ethics Statement

In this work, we propose modifying the forward pass for improved performance. While we evaluate different models on datasets, we acknowledge that we have not evaluated the impact of our method on safety, including concerns such as toxicity, bias, or content moderation. Our goal is to enhance accuracy, but broader implications—such as generating harmful content or sensitive applications—remain unexplored.

Acknowledgements

We are grateful to Jihyuk Kim, Jongho Kim, and Jongyoon Kim for their relecture, comments, and feedback.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. 2020. A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges.

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, and Harkirat Behl et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. 2024. STar-GATE: Teaching language models to ask clarifying questions. In *First Conference on Language Modeling*.

Gabriel Y. Arteaga, Thomas B. Schön, and Nicolas Pielawski. 2024. Hallucination detection in llms: Fast and memory-efficient finetuned models. *Preprint*, arXiv:2409.02976.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda et al. Askell. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Kamalika Chaudhuri, Sham M. Kakade, Karen Livescu, and Karthik Sridharan. 2009. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 129–136, New York, NY, USA. Association for Computing Machinery.

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.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4302–4310, Red Hook, NY, USA. Curran Associates Inc.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2019. Universal transformers. *Preprint*, arXiv:1807.03819.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, and Angela Fan et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, and Niklas et al. Muennighoff. 2021. A framework for few-shot language model evaluation. Zenodo.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*.

Alex Graves. 2017. Adaptive computation time for recurrent neural networks. *Preprint*, arXiv:1603.08983.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. NEFTune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, and Lucile Saulnier et al. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Daniel Kahneman. 2012. *Thinking, fast and slow*. Penguin, London.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2024. A call for clarity in beam search: How it works and when it stops. In *LREC/COLING*, pages 77–90.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2024. Selective attention improves transformer. *Preprint*, arXiv:2410.02703.

Ziqin Luo, Haixia Han, Haokun Zhao, Guochao Jiang, Chengyu Du, Tingyun Li, Jiaqing Liang, Deqing Yang, and Yanghua Xiao. 2024. Sed: Self-evaluation decoding enhances large language models for better generation. *Preprint*, arXiv:2405.16552.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Amitai Shenhav, Matthew M. Botvinick, and Jonathan D. Cohen. 2013. The expected value of control: An integrative theory of anterior cingulate cortex function. *Neuron*, 79(2):217–240. Funding Information: This work is supported by the C.V. Starr Foundation (A.S.), the National Institute of Mental Health R01MH098815-01 (M.M.B), and the John Templeton Foundation. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation.

Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. 2024. A thorough examination of decoding methods in the era of llms. *CoRR*, abs/2402.06925.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *Preprint*, arXiv:1304.5634.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. 2024. Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

A NEFTune versus Dropout

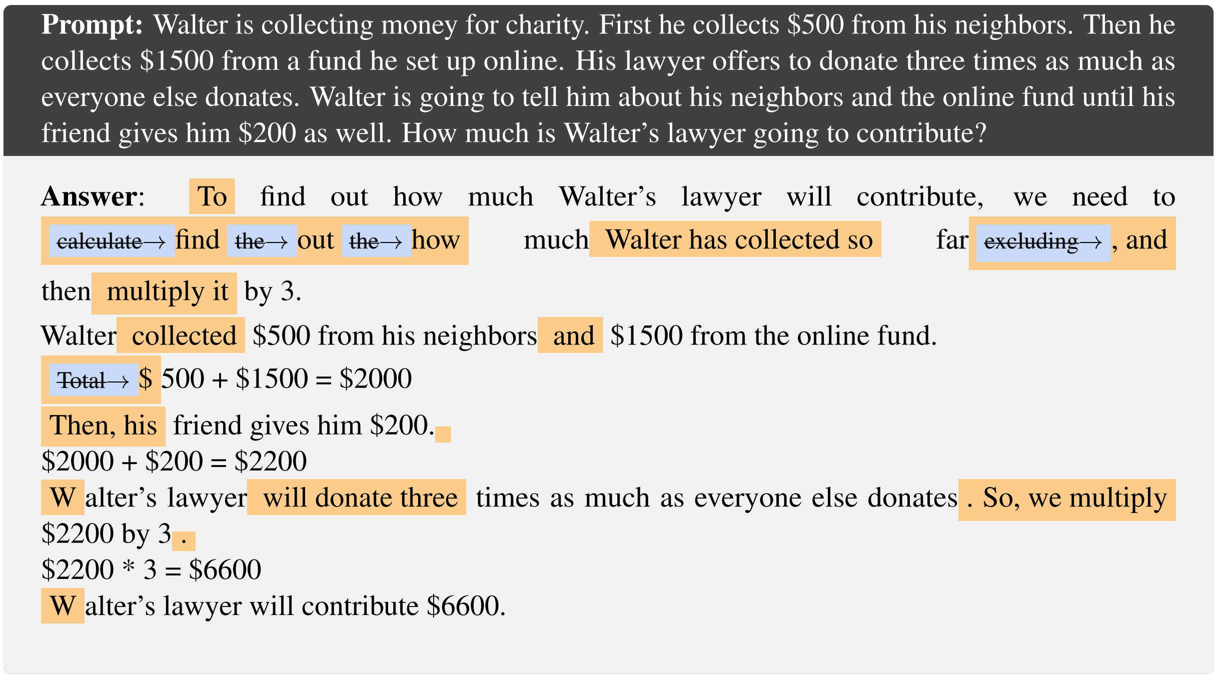
Methods	Datasets				
	CsQA	GSM8K	LAMBADA	MMLU Pro	CNN/DM
HARP (NEFTune$_{\alpha=5}$)	**80.30**	**78.89**	32.69	66.26	43.03
HARP (NEFTune$_{\alpha=10}$)	**80.30**	78.39	**38.23**	66.26	43.03
HARP (DROPOUT)	**80.30**	78.39	36.02	**71.43**	**48.21**

Table 5: Comparison of NEFTune noise (with hyperparameter α in $(5, 10)$) and dropout ($\delta = 20\%$) in the HARP method using LLaMA-3.1 Instruct 8B with greedy decoding. We report accuracy for each dataset, except CNN/DM, for which we report the ROUGE-1 score.

Table 5 compares HARP using NEFTune noise and dropout. It reveals that while NEFTune improves performance compared to the vanilla model (Table 1) by adding scaled uniform noise to embeddings, dropout offers more consistent enhancements. Specifically, dropout performances better on MMLU Pro and CNN/DM while having a close score to NEFTune on other datasets. Moreover, NEFTune$_{\alpha=5}$ results in a lower score than the vanilla forward pass in the LAMBADA next word prediction task. Dropout is a better candidate for infusing a different perspective representation into the model during inference time.

B Uncertainty in practice

In Figure 3, we present an answer generated with HARP modified forward pass, where we highlight the steps with a SHANNON entropy higher than the threshold θ. These uncertain steps happen right before a new decision is made. We also notice that, unlike humans, the model does not encounter uncertainty while playing with numbers. In this generation, HARP adjusted the top-1 predictions for five tokens before reframing. The initial top-1 predictions, before reframing, are highlighted in blue in Figure 3. Therefore, orange tokens without blue tokens are the ones where the initial prediction was already the current final token.



Prompt: Walter is collecting money for charity. First he collects \$500 from his neighbors. Then he collects \$1500 from a fund he set up online. His lawyer offers to donate three times as much as everyone else donates. Walter is going to tell him about his neighbors and the online fund until his friend gives him \$200 as well. How much is Walter's lawyer going to contribute?

Answer: To find out how much Walter's lawyer will contribute, we need to ~~calculate~~→ find ~~the~~→ out ~~the~~→ how much Walter has collected so far ~~excluding~~→ , and then multiply it by 3.
Walter collected \$500 from his neighbors and \$1500 from the online fund.
~~Total~~→ \$ 500 + \$1500 = \$2000
Then, his friend gives him \$200.
\$2000 + \$200 = \$2200
W alter's lawyer will donate three times as much as everyone else donates . So, we multiply \$2200 by 3 .
\$2200 * 3 = \$6600
W alter's lawyer will contribute \$6600.

Figure 3: Answer to the given prompt generated using HARP. Orange tokens highlight additional forward steps (*i.e.* tokens where uncertainty is higher than θ). ~~Blue tokens~~ represent the model's top-1 predictions prior to reframing.

Models	Methods	Datasets				
		CsQA	GSM8K	LAMBADA	MMLU Pro	CNN/DM
LLaMA 3.1 Instruct (8B)	Vanilla (Greedy)	**1.09**	**47.01**	**0.80**	24.46	**128.39**
	Beam Search	3.24 (x2.98)	113.27 (x2.41)	1.84 (x2.29)	79.59 (x3.25)	388.72 (x3.03)
	Ours (Greedy)	1.11 (x1.02)	55.28 (x1.18)	1.11 (x1.39)	23.20 (x0.95)	164.37 (x1.28)
	Vanilla (Nucleus)	**1.09**	**47.28**	**0.76**	**28.13**	**125.53**
	Ours (Nucleus)	1.12 (x1.02)	55.04 (x1.16)	1.11 (x1.46)	24.36 (x0.87)	169.19 (x1.35)
	Vanilla (CoT)	**22.28**	**41.92**	-	**49.75**	-
	Ours (CoT)	31.97 (x1.43)	49.99 (x1.19)	-	63.98 (x1.29)	-
Mistral v0.3 Instruct (7.25B)	Vanilla (Greedy)	**13.63**	**95.85**	**1.40**	**112.00**	**187.57**
	Beam Search	50.72 (x3.72)	240.47 (x2.51)	4.40 (x3.16)	319.76 (x2.86)	578.84 (x3.09)
	Ours (Greedy)	16.16 (x1.19)	111.94 (x1.17)	1.97 (x1.41)	133.16 (x1.19)	231.45 (x1.23)
	Vanilla (Nucleus)	**16.19**	**99.08**	**1.36**	**110.82**	**198.16**
	Ours (Nucleus)	20.77 (x1.28)	121.23 (x1.22)	2.03 (x1.49)	133.02 (x1.20)	244.85 (x1.24)
	Vanilla (CoT)	**21.95**	**114.47**	-	**129.55**	-
	Ours (CoT)	31.59 (x1.44)	135.08 (x1.18)	-	154.16 (x1.19)	-
Phi 3.5 Mini Instruct (3.82B)	Vanilla (Greedy)	**24.64**	**54.07**	**0.73**	**93.42**	**188.01**
	Beam Search	85.77 (x3.48)	177.82 (x3.29)	2.06 (x2.84)	270.75 (x2.90)	636.49 (x3.39)
	Ours (Greedy)	33.57 (x1.36)	57.22 (x1.06)	0.97 (x1.33)	113.25 (x1.21)	255.51 (x1.36)
	Vanilla (Nucleus)	**26.49**	**56.63**	**0.73**	**94.00**	**188.27**
	Ours (Nucleus)	37.07 (x1.40)	59.75 (x1.06)	0.97 (x1.34)	109.94 (x1.17)	262.50 (x1.39)

Table 6: Average inference time in seconds for various models and methods across downstream datasets. Numbers in parentheses indicate the relative inference time compared to each model's Vanilla corresponding method. On average, HARP is x1.25, while beam search is x3.01, making it almost 2.5 times faster than beam search. This table complements Figure 2.

C Impact of the hesitation threshold

Tuning the hyperparameter θ is beyond the scope of this paper. However, we provide a preliminary study to analyze its impact. We evaluate two datasets with different θ values and report relative accuracy and inference time in Table 7 and Table 8 respectively.

As θ increases, inference speed improves thanks to fewer reframing steps—leading to reduced computation. However, this comes at the cost of lower accuracy. Conversely, a too small θ can also degrade accuracy. This aligns with the discussion in Section 6 on multi-step reframing, where additional reframing may cause the model to "overthink" and lose track of improvements.

This preliminary study is based on a limited set of tasks, and further investigation is needed to better understand the selection of θ.

Datasets	Value of θ							
	0.6	0.8	**1.0**	1.2	1.4	1.6	1.8	2.0
CsQA	x1.03	x1.02	x1.01	x1.00	x1.00	x1.00	x1.00	x1.00
LAMBADA	x1.53	x1.49	x1.40	x1.33	x1.31	x1.29	x1.27	x1.26

Table 7: Relative inference time of HARP using LLaMA-3.1 Instruct 8B with different values of θ compared to the Vanilla method.

Datasets	Value of θ							
	0.6	0.8	**1.0**	1.2	1.4	1.6	1.8	2.0
CsQA	+1.72	+2.12	+1.52	+1.31	+1.72	+1.52	+1.41	+1.31
LAMBADA	+5.12	+5.21	+5.16	+5.12	+5.08	+5.02	+4.95	+4.85

Table 8: Relative accuracy of HARP using LLaMA-3.1 Instruct 8B with different values of θ compared to the Vanilla method.

D Prompts

This section presents the prompt for evaluating models on the different datasets. "*{. . . }*" refers to the data extracted from the dataset itself for each example.

Multiple-choice Question Prompt

General Instructions: Please read the multiple-choice question below carefully and select ONE of the listed options. Please write "The answer is LETTER".
Question: `{question}`
Options:
`{options}`

Figure 4: Multiple-choice Question prompt (CommonsenseQA and MMLU Pro).

GSM8K Prompt

`{question}`

Figure 5: GSM8K prompt.

LAMBADA Prompt

You have to predict the next word of the sentence given the context. Your answer should be a single word.
Context:
`{context (minus one sentence)}`
Sentence to continue:
`{last sentence (minus one word)}`

Figure 6: LAMBADA prompt.

CNN/DM Prompt

Summarize the given article. Do not output something else than the summary.
Article:
`{article}`

Figure 7: CNN/DailyMail prompt.