Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using *Only* Attention Maps

Yung-Sung Chuang[†] **Linlu Qiu**[†] **Cheng-Yu Hsieh**[‡] **Ranjay Krishna**[‡]
Yoon Kim[†] **James Glass**[†]
Massachusetts Institute of Technology[†] University of Washington[‡]
yungsung@mit.edu

Abstract

When asked to summarize articles or answer questions given a passage, large language models (LLMs) can hallucinate details and respond with unsubstantiated answers that are inaccurate with respect to the input context. This paper describes a simple approach for detecting such *contextual hallucinations*. We hypothesize that contextual hallucinations are related to the extent to which an LLM attends to information in the provided context versus its own generations. Based on this intuition, we propose a simple hallucination detection model whose input features are given by the ratio of attention weights on the context versus newly generated tokens (for each attention head). We find that a linear classifier based on these *lookback ratio* features is as effective as a richer detector that utilizes the entire hidden states of an LLM or a text-based entailment model. The lookback ratio-based detector—***Lookback Lens***—is found to transfer across tasks and even models, allowing a detector that is trained on a 7B model to be applied (without retraining) to a larger 13B model. We further apply this detector to mitigate contextual hallucinations, and find that a simple classifier-guided decoding approach is able to reduce the amount of hallucination, for example by 9.6% in the XSum summarization task.[1]

1 Introduction

Despite the utility and impressive capabilities of large language models (LLMs), their tendency to generate hallucinations, i.e., content that deviates from facts or contextually relevant information (Ji et al., 2023), presents a significant challenge in their deployment. In this work, we focus on the scenarios where the model is provided with the correct facts within the input context but still fails to generate accurate outputs, a phenomenon we term *contextual hallucination*. Despite the simplicity of this setup, LLMs struggle with contextual hallucinations, frequently producing errors in tasks such as summarization and document-based question answering (e.g., Table 1), which can cause serious issues in applications such as retrieval-augmented generation (RAG) (Lewis et al., 2020), even when correct documents are retrieved.

Most prior studies that propose methods to combat hallucination focus on the scenario *without* any input context, where the hallucinations arise from the LLMs' parametric knowledge. These works detect and mitigate hallucinations by generally using the LLM's representations, such as hidden states (Burns et al., 2023; Azaria and Mitchell, 2023), MLP outputs (Zhang et al., 2024; Simhi et al., 2024), attention block outputs (Zhang et al., 2024; Simhi et al., 2024) and attention head outputs (Li et al., 2024; Chen et al., 2024b; Simhi et al., 2024). In contrast, the provided contextual information plays a key role in detecting contextual hallucinations. Insofar as attention (more so than other model internals) provides a human-meaningful measure of how much weight is given to the context during generation, this motivates the use of signals from the attention maps for hallucination detection and mitigation.

To leverage signals from attention maps, we start by hypothesizing that contextual hallucinations are related to the extent to which an LLM attends to the provided contextual information. Concretely, we propose a simple feature called *lookback ratio*, which is computed as the ratio of attention weights on the given context versus the newly generated tokens. At each time step, we calculate this lookback ratio for each attention head, and train a linear classifier, which we call the *Lookback Lens*, to detect contextual hallucinations based on the lookback ratio features, as illustrated in Figure 1. The *Lookback Lens* performs on par with, and sometimes even surpasses, more complex feature-based detectors that utilize hidden states from LLMs or text-

[1]Source code: github.com/voidism/Lookback-Lens

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1419–1436
November 12-16, 2024

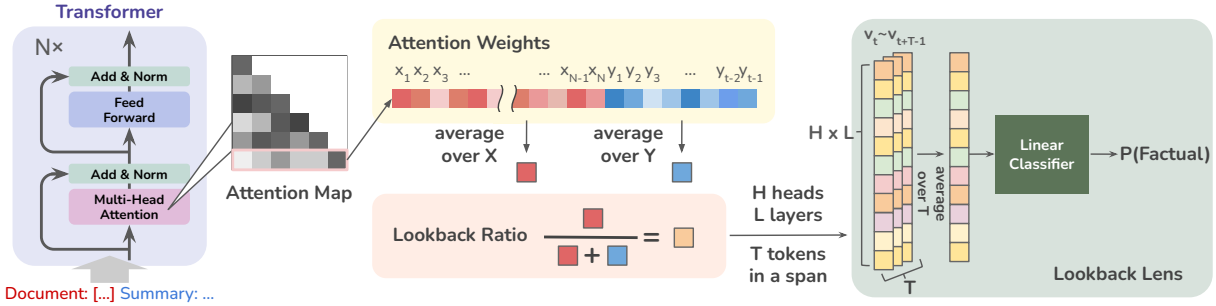

Figure 1: An illustration of the *Lookback Lens*. We extract attention weights and calculate the lookback ratios for all layers and all heads. We train a linear classifier on the concatenated features to predict truthfulness of the generation.

based entailment models trained on extensively annotated datasets. We can further integrate this detector during decoding to derive a *Lookback Lens Guided Decoding* strategy which can reduce contextual hallucinations by 9.6% from LLaMA-2-7B-Chat in the XSum summarization task. Furthermore, our use of "higher level" attention map features makes it possible to transfer the detector across models *without* retraining, allowing a LLaMA2-13B-Chat model to use the same detector that has been trained on LLaMA-2-7B-Chat, and still reduce hallucinations by 3.2% in XSum. These results collectively highlight the potential of combating contextual hallucination by leveraging the information from attention maps.

2 Contextual Hallucinations Detection

2.1 Lookback Lens

To *detect* contextual hallucinations in LLMs, we introduce a lookback ratio, a measure based on the attention distribution of a transformer model. Given a transformer with L layers, each with H heads, the model processes an input sequence of context tokens $X = \{x_1, x_2, \ldots, x_N\}$ of length N followed by a set of newly generated tokens $Y = \{y_1, y_2, \ldots, y_{t-1}\}$ to generate the next token y_t. For time step t, and for each head, we calculate the ratio of attention weights focused on the context tokens versus the newly generated tokens. Formally, for each head h in layer l, we define:

$$A_t^{l,h}(\text{context}) = \frac{1}{N}\sum_{i=1}^{N} \alpha_{h,i}^{l},$$

$$A_t^{l,h}(\text{new}) = \frac{1}{t-1}\sum_{j=N+1}^{N+t-1} \alpha_{h,j}^{l},$$

where $\alpha_{h,i}^{l}$ and $\alpha_{h,j}^{l}$ are softmax-ed attention weights assigned to context tokens X and new tokens Y respectively. The lookback ratio $\text{LR}_t^{l,h}$ for head h in layer l at time step t is then calculated as:

$$\text{LR}_t^{l,h} = \frac{A_t^{l,h}(\text{context})}{A_t^{l,h}(\text{context}) + A_t^{l,h}(\text{new})}.$$

To utilize these lookback ratios as input features in detecting hallucinations, we concatenate the lookback ratios across all heads and layers into a feature vector for the time step t:

$$\mathbf{v}_t = [\text{LR}_t^{1,1}, \text{LR}_t^{1,2}, \ldots, \text{LR}_t^{L,H}].$$

Given a text span of interest $\{y_t, y_{t+1}, ..., y_{t+T-1}\}$, we average the corresponding lookback ratio vectors $\{\mathbf{v}_t, \mathbf{v}_{t+1}, ..., \mathbf{v}_{t+T-1}\}$ into a single vector $\bar{\mathbf{v}}$. We then employ a logistic regression classifier $\mathcal{F}$ to predict if the span is factual (1) or hallucinated (0) based on the averaged lookback ratio vector.

$$P(y = 1|\bar{\mathbf{v}}) = \mathcal{F}(\bar{\mathbf{v}}) = \sigma(\mathbf{w}^\top \bar{\mathbf{v}} + b),$$

where σ denotes the sigmoid function, $\mathbf{w}$ is the weight vector, and b is the bias term of the classifier.

Defining Span The *Lookback Lens* predicts the probability of hallucinations over spans. We consider two ways to obtain spans for a given sequence: *predefined spans* or *sliding window*.

1) Predefined Spans: When the hallucinated and non-hallucinated span annotations are available, we directly train the classifier to differentiate between them. This is a clean setting where all spans are either hallucinated or non-hallucinated.

Dataset	Examples	Correct
CNN/DM	1000	49.6%
NQ	2655	67.8%

Table 1: Dataset statistics and GPT-4o evaluation results on responses greedy decoded by LLaMA-2-7B-chat.

2) Sliding Window: In practice, we do not have any predefined spans during decoding, thus we need a sliding window setup that iterates over all possible spans. Specifically, we process the sentences into fixed-sized chunks and train the classifier to predict a label of 0 if any hallucinated content exists within a chunk, and 1 otherwise. Here, the annotated data is only used for creating labels, not for the span segmentation. This is more realistic for classifier-guided decoding, but it presents greater challenges because a chunk can contain both hallucinated and non-hallucinated content.

2.2 Experimental Setup

Data Training the *Lookback Lens* requires labels for hallucinated and non-hallucinated examples. To obtain these examples, we first prompt LLaMA-2-7B-Chat (Touvron et al., 2023) to greedy decode responses for 1,000 summarization examples from the CNN/DM dataset (See et al., 2017) and 2,655 QA examples from the Natural Questions (Kwiatkowski et al., 2019) following the setup of Liu et al. (2024). More details are shown in Appendix A. Although being prompted to generate correct responses, the decoded responses will contain both hallucinated and non-hallucinated information as the LLaMA model is still not perfect. Then, we employed GPT-4o (OpenAI, 2024) to verify the truthfulness of these responses and provide span-level annotations on hallucinated segments (detailed prompts in Appendix B.1).

Additionally, we performed a pilot study of human annotation on a subset of 70 examples of the summarization task (details in Appendix B.2), confirming a 97% consistency rate between GPT-4o annotations and human judgments, and validating the reliability of the automated annotations. We show LLaMA-2-7B-Chat's results on both tasks, as evaluated by GPT-4o, in Table 1. The results show that the generated summaries from LLaMA-2-7B-Chat still exhibit hallucinations about half of the time, highlighting the challenge of summarization tasks.

Baselines We compare our detection method against several baselines: **1) Text-based entailment classifier:** We fine-tune the DeBERTa-v3-base (He et al., 2021) model on the same dataset of CNN/DM and NQ as a natural language entailment (NLI) task. Additionally, we include the results from a state-of-the-art entailment model (Vectara, 2023) trained on a huge amount of annotated NLI data (see details in Appendix C.1).

2) Hidden states-based classifier: We train classifiers using the same setting as the *Lookback Lens* but used input features from the hidden states of LLaMA-2-7B-Chat from its 24th, 28th, and 32nd layers instead of the lookback ratio. This baseline resembles a broad range of existing methods in the literature (Azaria and Mitchell, 2023; Simhi et al., 2024). Our selection of layers followed the findings outlined in Azaria and Mitchell (2023), which used layers 32, 28, 24, and 20 of a 32-layer LLM for detecting hallucinations. They find that layers near the 28th layer are most effective (see Table 3 and 4 in Azaria and Mitchell (2023)).

We include additional experiments for leveraging multiple layers or all layers in predicting contextual hallucinations in Appendix D.2, but the results are not significantly better than using the 28th layer. Some papers suggest attention block outputs could be more useful for detecting hallucinations (Campbell et al., 2023; Li et al., 2024), we include the additional comparative experiments in Appendix D.3, but the difference between hidden states and attention block outputs is relatively small.

2.3 Results

Our results are presented in Table 2. We consider both predefined span segmentation and sliding window with a window size of 8. We include the two-fold validation setting on the source task and the out-of-domain transfer setting on the target task, with the tasks either question answering (QA) or summarization (Sum.). We find that the *Lookback Lens* achieves slightly better performance than the hidden states-based classifier and significantly outperforms the NLI models (SoTA and our impl.). The advantage of the *Lookback Lens* over the hidden states-based classifier is more significant in the sliding window settings, as shown in the right-hand side of Table 2.

Additionally, we observe that the hidden states-based classifier tends to overfit the training sets during the two-fold validation, and present a substantial performance drop when transferred to out-of-domain tasks. In contrast, *Lookback Lens*, while not always fitting the training set perfectly, consistently exhibits better performance when applied to out-of-domain tasks. This contrast highlights the effectiveness and generalizability of the lookback ratio features we extract from the attention maps.

Method	Source	Target	Predefined Span			Sliding Window = 8		
			Source ⟶ Target			Source ⟶ Target		
			Train	Test	Transfer	Train	Test	Transfer
Text based NLI								
SoTA NLI	–	Sum.	–	–	76.6	–	–	57.1
SoTA NLI	–	QA	–	–	58.6	–	–	61.8
NLI (our impl.)	QA	Sum.	–	–	55.1	–	–	53.0
NLI (our impl.)	Sum.	QA	–	–	71.0	–	–	64.9
Hidden states based								
32nd Layer	QA	Sum.	100.0	89.6	79.4	99.0	97.1	56.1
32nd Layer	Sum.	QA	100.0	82.5	81.8	97.0	94.8	59.4
28th Layer	QA	Sum.	100.0	91.4	83.6	99.2	97.3	57.7
28th Layer	Sum.	QA	100.0	83.3	84.7	97.2	95.2	58.8
24th Layer	QA	Sum.	100.0	92.0	81.3	99.2	97.4	58.3
24th Layer	Sum.	QA	100.0	83.1	83.0	99.2	97.4	58.3
Attention maps based (Ours)								
Lookback Lens	QA	Sum.	98.3	91.2	85.3	88.3	87.1	66.1
Lookback Lens	Sum.	QA	97.7	88.8	82.0	86.2	85.3	66.0

Table 2: AUROC of the classification tasks using predefined span segmentation and sliding window (size = 8) on NQ (QA) and CNN/DM (Sum.). The source task scores (Train/Test) are averaged over two-fold validation.

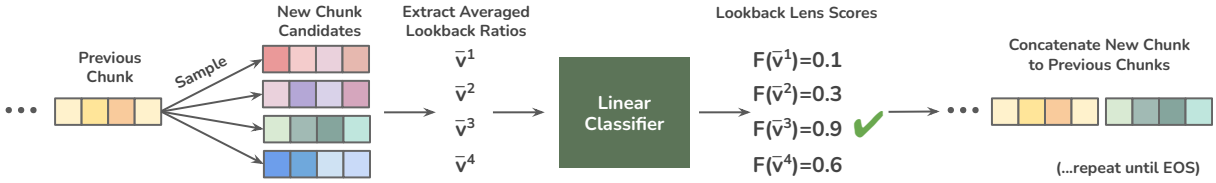

Figure 2: *Lookback Lens Guided Decoding*: sample multiple chunk candidates, compute lookback ratios from attention maps to be scored by *Lookback Lens*, and select the best candidate that is less likely to be hallucinations.

3 Contextual Hallucinations Mitigation

3.1 Lookback Lens Guided Decoding

To mitigate the impact of contextual hallucinations identified by the *Lookback Lens*, we introduce a classifier-guided decoding strategy to guide the generation toward more contextually accurate outputs. This approach serves as a robustness test of the *Lookback Lens*' ability to handle various text generation scenarios. While prior studies on controllable text generation adjust the output probabilities using classifiers based on the output tokens (Yang and Klein, 2021), our method fundamentally differs by not using the tokens themselves but rather their attention maps during generation.

We propose *Lookback Lens Guided Decoding*, which incorporates *Lookback Lens* ($\mathcal{F}$) into the decoding process. Since all tokens in the vocabulary share the same attention pattern during one decoding step, $\mathcal{F}$ cannot directly influence one-step token choice. Instead, $\mathcal{F}$ can evaluate multiple-token chunks, as each chunk causes different attention patterns in multiple decoding steps.

Given the context and partially generated text, we independently sample a set of k candidate chunks $\{C_1, C_2, \ldots, C_k\}$ at the same decoding step t. For each chunk C_j, the associated lookback ratios are averaged to form a feature vector $\bar{\mathbf{v}}^j$. As shown in Figure 2, we select the best candidate C^* predicted by $\mathcal{F}$ and append to the generation,

$$C^* = \underset{C_j \in \{C_1, C_2, \ldots, C_k\}}{\arg\max} \mathcal{F}(\bar{\mathbf{v}}^j).$$

We repeat this process until it generates the EOS token or reaches the maximum length.

3.2 Experimental Setup

We evaluate *Lookback Lens Guided Decoding* on three tasks that involve generating texts conditioned on given contexts, including summarization with XSum (Narayan et al., 2018), QA with NQ (Kwiatkowski et al., 2019), and multi-turn conversations with MT-bench (Zheng et al., 2024).

For testing the generalization ability of the *Lookback Lens*, we only train it with the CNN/DM sum-

marization dataset from the detection task in Section 2.2. Thus, only the XSum dataset will be the same-task transfer setting, while NQ and MT-bench will be cross-task transfer setting.

XSum To test the *Lookback Lens*'s effectiveness at transferring across data distributions for the same task (summarization), we use 1,000 examples sampled from the testing set of XSum. Prior studies (Maynez et al., 2020) indicate that traditional evaluation metrics such as ROUGE (Lin, 2004) or BERTScore (Zhang et al., 2019a) correlated poorly with human evaluation on faithfulness and factuality. Recent studies (Chiang and Lee, 2023; Liu et al., 2023) also show a strong correlation between GPT-4 (OpenAI, 2023) evaluation and human evaluation. Thus, we report the averaged accuracy from the binary judgments of GPT-4o, with the prompts in Appendix B.1. We also conduct a pilot study for human evaluation on GPT-4o's judgment in Appendix B.2, finding that 97% of the GPT-4o judgments are consistent with human judgment.

Natural Questions We use the NQ data from the setup of Liu et al. (2024) we describe in Appendix C.2 and evaluate the best span exact match following Kandpal et al. (2023); Mallen et al. (2023).

MT-Bench We consider a multi-turn conversations setup where the model needs to follow previous chat history. We use MT-bench (Zheng et al., 2024), a multi-turn instruction-following benchmark covering eight categories. We focus exclusively on generating responses for the second turn and use GPT-3.5's responses as the default for the first turn. We use GPT-4 to score the model's answers on a scale of 1 to 10 based on various factors, including *helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response*.

Additionally, since we are particularly interested in mitigating contextual hallucinations, we further exclude math questions and evaluate the remaining 50 general questions. We specifically instruct GPT-4o to focus on whether the responses are faithful to the chat history (see prompt in Appendix B.1). We refer to this setup as MT-Bench (hallu.).

Baselines To evaluate the performance of our proposed method, we compared it against the following baselines: **1) Greedy Decoding:** generating responses using the LLaMA-2-7B-Chat model (Touvron et al., 2023) through greedy decoding. **2) Other Classifier-Guided Decoding:** using exactly the same setting but with different classifiers introduced in Section 2.2, including text-based entailment classifiers and hidden states-based classifiers.

Method	XSum	NQ	MT-Bench	
			Hallu.	Ori.
Greedy Decoding	49.0	71.2	6.08	5.10
Text-based classifier guided decoding				
SoTA NLI†	59.0	74.2	6.12	5.03
NLI (our impl.)	44.1	72.5	5.72	4.99
Hidden states based classifier guided decoding				
32nd layer	48.3	73.9	5.49	4.91
28th layer	48.9	73.0	5.71	5.06
24th layer	47.5	73.9	5.65	5.16
Lookback Lens guided decoding				
Ours	58.6	74.2	6.27	5.10

Table 3: Decoding results using 8 candidates per chunk in a chunk size of 8. We compare our methods with greedy decoding and classifier-guided decoding using the NLI models, and hidden state representations of different layers. †The SoTA NLI is trained on 731k examples so it may not be directly comparable.

3.3 Main Results

We show our results using eight candidates per chunk in a chunk size of eight in Table 3, and the ablation with different chunk sizes is shown in Table 6. *Lookback Lens Guided Decoding* can improve the performance on both in-domain task (XSum, by 9.6%) and out-of-domain tasks (NQ, by 3%). The original greedy decoding results on XSum achieved 49.0% correct which means 510 examples were hallucinated. Our decoding method significantly reduced the number of hallucinated examples from 510 to 414, resulting in an 18.8% reduction in the hallucinated examples. This result is on par with using SoTA NLI to guide the decoding, where SoTA NLI is trained on roughly 731k annotated summarization examples, which is 700× larger compared to our 1k training set. (See Appendix C.1.) In contrast, decoding guided by hidden states-based or the NLI (our implementation) classifiers, both trained on the same data of our method, can only slightly improve the performance on NQ, but not for XSum, probably due to the issue of distribution shift, highlighting the advantages of *Lookback Lens* in generalization ability.

For MT-bench, we evaluate both settings: the original setting (ori.) and the setting that is specifically for judging contextual hallucinations (hallu.).

Source	Target	Predefined Span	Sliding Window
Lookback Lens: Train 13B → Test 13B			
QA	Sum.	84.0	60.4
Sum.	QA	84.3	60.8
QA-train	QA	93.3	63.7
Lookback Lens: Train 7B → Test 13B			
QA	Sum.	73.5	58.8
Sum.	QA	78.2	60.5
QA-train	QA	80.6	62.4

Table 4: Cross model transfer results on detection tasks.

Method	XSum	NQ	
Greedy	52.9	74.0	
Text-based classifier guided decoding			
SoTA NLI†	59.6	74.4	
Method	**CNN/DM →XSum**	**NQ-train →NQ**	**CNN/DM →NQ**
Lookback Lens guided decoding			
13B → 13B	57.9	75.6	74.8
7B → 13B	56.1	76.4	73.7

Table 5: Cross model transfer from LLaMA-2-7B-chat to LLaMA-2-13B-chat using greedy decoding and classifier guided sampling methods with chunk size 8.

We do not expect our method can improve on the original setting, because it evaluates many factors such as helpfulness, relevance, etc. But we expect to see an improvement on the hallucination setting. The results shown in Table 3 suggest that our decoding method can boost the performance on the hallucination setting while maintaining the same performance in the original setting, which shows that our decoding method is effective in reducing hallucinations without compromising the overall generation quality.

4 Cross-model Transfer

One benefit of using the lookback ratio to capture higher-level model patterns for hallucination detection is its potential to better transfer across models. A classifier trained with one model's lookback ratio could potentially be applied to another model *without* retraining, provided correlation between the target model's attention pattern and that of the original model. Here, we show that we can transfer a *Lookback Lens* trained on attention maps from LLaMA-2-7B-Chat to LLaMA-2-13B-Chat without any retraining.

Since the total numbers of attention heads are different in 7B and 13B models, and there is no obvious one-to-one mapping between the heads, we use a linear regression model to map the heads from the 13B model to the heads in 7B model. Concretely, we have 1024 heads in 7B and 1600 heads in 13B. We extract the averaged lookback ratio per head for all the $|D|$ training examples, resulting in a $1024 \times |D|$ matrix and a $1600 \times |D|$ matrix.[2] We then fit a linear regression model to map the heads to reconstruct the 7B heads from 13B heads. After applying the linear transformation to the lookback ratio from 13B, the transformed heads can be directly used by 7B's classifiers. See details in Appendix C.1.

The detection results are shown in Table 4. We first show the same-model (13B→13B) + cross-task transfer result, and the cross-model (7B→13B) + cross-task transfer result. Although cross-model transfer yields slightly worse results compared to same-model transfer, the AUROC scores are still non-trivially high. Consider that doing cross-model + cross-task transfer at the same time may be tough to *Lookback Lens*, we also include one more setting that does training on 2.5K examples of the NQ training set[3] and then transfer to the NQ testing set. We see the cross-model same-task transfer results are even closer to the same-model transfer results.

Given promising results on detection tasks, we apply cross-model transfer to *Lookback Lens Guided Decoding*. We conduct the same-task transfer setting: NQ-train (7B) to NQ (13B), and CNN/DM (7B) to XSum (13B). In Table 5, we observe a performance improvement similar to same-model transfer using 13B itself, or using the SoTA NLI model applied on the 13B decoding. However, on cross-task + cross-model transfer settings: CNN/DM (7B) to NQ (13B), we do not observe significant improvements where we attribute to the larger distribution shift. We leave this challenging setting for future work.

5 Discussions and Ablations

In this section, we further conduct various experiments and ablation studies on the *Lookback Lens* and its corresponding classifier guided decoding.

Effect of Chunk Size In Section 3.3 (Table 3), we experiment with chunk size = 8. Here, we study

[2]To ensure that two models are generating the same content when extracting lookback ratio, we decode from 7B and run the 13B model on the 7B outputs.

[3]The NQ-train 2.5K data is annotated in the same method to annotate NQ testing set, as described in Section 2.2.

the effect of varying chunk sizes, from 4, 8, to 16. We see that there is a slight trend that *Lookback Lens* guided decoding prefers shorter chunk size for NQ and longer chunk size for XSum. However, in general the improvements are consistent across different chunk sizes, thus reducing the need to optimize for chunk sizes.

Method	**NQ**			**XSum**		
Chunk size=	4	8	16	4	8	16
Greedy		71.2			49.0	
Text-based classifier guided decoding						
SoTA NLI†	73.7	74.2	74.4	57.3	59.0	62.1
Hidden states based classifier guided decoding						
32nd layer	72.6	73.9	72.7	48.9	48.3	48.3
28th layer	72.9	73.0	74.1	47.2	48.9	47.1
24th layer	75.0	73.9	72.5	47.6	47.5	51.2
Lookback Lens guided decoding						
Ours	75.4	74.2	74.3	53.2	58.6	57.7

Table 6: Performance comparison on various datasets using different methods and chunk sizes.

Method	**Predefined Span**					
	QA → Sum.			**Sum. → QA**		
All heads		85.3			82.0	
Top-k heads only						
with k =	10	50	100	10	50	100
Largest mag.	71.2	82.3	82.8	79.2	80.3	81.1
Most positive	65.1	74.9	75.4	66.3	70.3	74.4
Most negative	59.5	67.5	74.4	66.4	70.2	73.0

Table 7: Cross-task transfer AUROC using top-k attention heads selected according to: coefficients with the largest magnitude (largest mag.), most positive, and most negative. We consider k = 10, 50, and 100.

Predictive Power of Different Heads In the aforementioned experiments, we utilize all attention heads to train the *Lookback Lens*. We are thus interested in how the predictive power is distributed among different heads in making predictions. That is, how much performance can we recover if we only utilize a subset of heads? To answer this, we use the coefficients in the linear classifier of the *Lookback Lens* (in Section 2) to estimate the importance of each head in detecting hallucinations.

In Table 7, we show the results on detection tasks achieved by different detectors trained using only a subset of top-k heads with the largest magnitude of coefficients in the original *Lookback Lens* trained will all heads. The results show that the predictive power is not concentrated only on a subset of heads. Using only top-10 heads is worse than using all heads, and increasing k consistently improves performance and top-100 heads largely recover the model's performance using all heads.

Layers	**Predefined Span**	
	QA → Sum.	**Sum. → QA**
Layer 1-4	69.6	64.0
Layer 5-8	75.6	70.1
Layer 9-12	75.4	68.3
Layer 13-16	81.2	78.2
Layer 17-20	80.8	78.2
Layer 21-24	64.4	73.1
Layer 25-28	66.0	74.4
Layer 29-32	66.4	71.4
Layer 1-32	85.3	82.0

Table 8: Cross-task transfer AUROC among layers.

More interestingly, we also include the results that only select the top-k heads among the heads with most positive/negative coefficients, which are positive/negatively correlated to factuality. On the heads with positive coefficients, higher lookback ratio (i.e., when the heads attend at the context more) indicates higher factuality and less hallucination; conversely, heads with negative coefficients suggest a lower lookback ratio (i.e., attending to generated tokens more) is more likely to be truthful. Table 7 shows that none of positive or negative heads alone can be on par with using the top-k largest magnitude heads. This result implies that both positive and negative heads are critical for a model to generate factual responses. We conjecture that the positive heads may specialize at context grounding, and thus higher lookback ratio on these heads leads to more factual response. On the other hand, the negative heads may be critical at ensuring consistency in its own generation, and thus should attend to the generated tokens more. We leave further investigation on this interesting balance for future work. Meanwhile, we visualize the lookback ratio of positive/negative heads in Appendix D.1.

Reducing Number of Layers We experiment with using only a subset of layers for *Lookback Lens*, as shown in Table 8. We can see that the predictive power is not concentrated in any subset of layers, as none of them can recover the performance of the full model that uses all layers. However, we observe that the middle layers (13-16, 17-20) are slightly more useful than other layers.

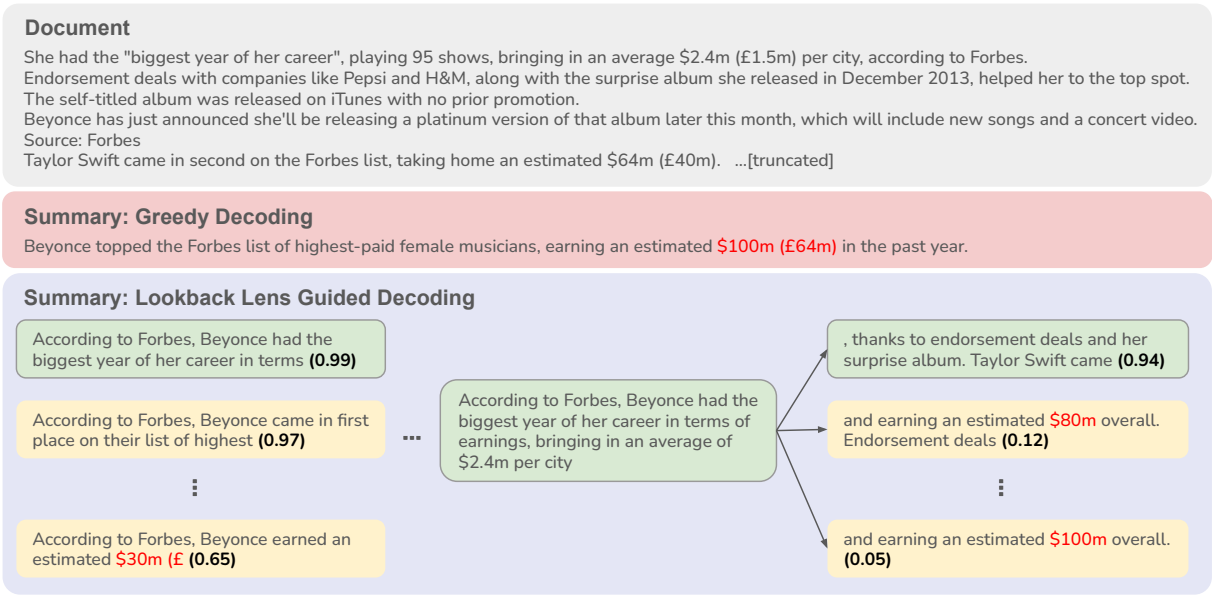

Figure 3: Qualitative example on XSum using the LLaMA-2-7B-Chat model with greedy decoding and *Lookback Lens Guided Decoding*. The numbers in the parenthesis show the predicted scores from the *Lookback Lens*.

Qualitative Study We show qualitative examples from XSum in Figure 3 to illustrate how *Lookback Lens* guided decoding improves performance. Greedy decoding from LLaMA-2-7B-Chat results in a hallucination, i.e. *$100m (£64m)*, that does not exist in the input document. However, the *Lookback Lens* is able to assign low scores for the chunk candidates that have contextual hallucinations (as marked in red). Therefore, *Lookback Lens Guided Decoding* is able to help the model generate a summary that is factual to the given context.

6 Related Work

Hallucinations in LLMs Simhi et al. (2024) defined *close-book hallucination* vs *open-book hallucination* for settings of relying on parametric knowledge vs knowledge in context. We term *open-book hallucination* as *contextual hallucination* for better clarity. Previous studies in hallucinations primarily focus on close-book hallucinations (Chen et al., 2023; Min et al., 2023; Chern et al., 2023) and their detection (Azaria and Mitchell, 2023; Simhi et al., 2024) and mitigation (Li et al., 2024; Chuang et al., 2024; Chen et al., 2024a; Zhang et al., 2024). Most of the studies focus on leveraging LLM's internal representations, such as hidden states (Burns et al., 2023; Azaria and Mitchell, 2023), MLP outputs (Zhang et al., 2024; Simhi et al., 2024), attention block outputs (Zhang et al., 2024; Simhi et al., 2024) and attention head outputs (Li et al., 2024; Chen et al., 2024b; Simhi et al., 2024). Our work, however, focuses on contextual hallucinations, where models produce content inconsistent with the provided context (Maynez et al., 2020; Fabbri et al., 2021; Shi et al., 2023). Thus, different from prior studies, we focus on the attention maps instead of internal representations, as we believe that the attention maps patterns record how the LLM process the given contextual information. Most of the prior studies treat detection and mitigation as two separate tasks, expect for Simhi et al. (2024); Chen et al. (2024a). Our work focuses not only on detection, but also tries to incorporate the detector into the decoding process to further mitigate the contextual hallucinations. Recently, Simhi et al. (2024) also explored detecting and mitigating both close-book and open-book hallucinations. However, their open-book hallucination setting is limited to DisentQA (Neeman et al., 2023), which creates knowledge conflicts between parametric knowledge and given context. In contrast, we focus on LLaMA-2's naturally generated responses to capture general cases where LLMs fail to follow the context, not just due to knowledge conflicts.

Classifier Guided Generation Classifier guided generation aims to control attributes like topic or sentiment in text generation. PPLM (Dathathri et al., 2019) uses gradient ascent to adjust LM probabilities via attribute classifiers. FUDGE (Yang and Klein, 2021) uses an attribute predictor on partial sequences to modify LM probabilities. Our method uniquely guides generation using classifiers on attention maps, setting it apart from prior approaches.

Self-attention and Model Behavior The attention mechanism, initially introduced in RNN-based encoder-decoder for neural machine translation (Bahdanau et al., 2015; Luong et al., 2015), was later adopted in the Transformer model's self-attention module (Vaswani et al., 2017), enabling greater parallelization. Self-attention's interpretability has led researchers to use it for understanding model behaviors (Clark et al., 2019; Hao et al., 2021; Vashishth et al., 2019). Our work demonstrates that attention maps in LLMs are effective for detecting contextual hallucinations, providing a lightweight and interpretable solution compared to complex hidden representation methods (Zhang et al., 2024; Chen et al., 2024b).

7 Conclusion

We introduce the *Lookback Lens*, a lightweight classifier designed to detect contextual hallucinations by utilizing the *lookback ratio*, which is computed solely from attention weights. This classifier not only effectively identifies contextual hallucinations but also mitigates them through *Lookback Lens Guided Decoding* from the LLM. Remarkably, the method is transferable across various tasks, and even across models after mapping their attention heads. This research opens up new possibilities for leveraging attention map information to combat hallucinations in large language models.

Limitations

Despite the effectiveness of the *Lookback Lens* and its decoding, there are several limitations to consider.

- First, the performance upper bound of *Lookback Lens Guided Decoding* is limited by the sampling capabilities of the LLM itself. If the LLM fails to sample the correct chunk among the eight candidates, the *Lookback Lens* cannot correct the error.

- Second, although the *Lookback Lens* is a lightweight classifier with negligible inference time, the requirement to sample multiple candidates from the LLM increases the total inference time. We argue that *Lookback Lens Guided Decoding* is a preliminary approach that demonstrates the feasibility of integrating the *Lookback Lens* into the decoding process, as well as a robustness test for the *Lookback Lens* to handle various text generation scenarios. However, other options, such as intervening in the attention map mechanism based on *Lookback Lens* signals, could potentially achieve faster inference, and we leave this for future work.

- Lastly, the *Lookback Lens* relies on annotated examples of around 1k-2k to train the classifier. While other end-to-end methods (Chuang et al., 2024) can mitigate close-book hallucinations without training data, they lack interpretability due to the absence of a detection step. Nevertheless, we believe that requiring 1,000 annotated examples is a feasible setting.

Acknowledgement

We sincerely thank Philip Schroeder, Huirong Wen, Andrew Rouditchenko, Nishad Gothoskar, Ani Nrusimha, Howard Chen, Weijia Shi, and Nour Jedidi for their discussion and help in this project. This research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. Linlu and Yoon were supported in part by MIT-IBM Watson AI Lab.

Ethics Statement

In this research, we used publicly available datasets and we did not collect any personal information. All datasets and models are used in accordance with their intended use and licenses. Our method is designed to improve the factuality of large language models (LLMs), which can have a positive impact on various applications, such as question-answering systems, summarization systems, and other applications that rely on LLMs. When deployed, however, our approach still carries the issues stemming from LLMs, which means that there is a risk that the LLM can produce biased, harmful, or offensive output. Therefore, caution should be exercised before implementing similar approaches in real-world applications.

References

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

James Campbell, Richard Ren, and Phillip Guo. 2023. Localizing lying in llama: Understanding instructed dishonesty on true-false questions through prompting, probing, and patching. *arXiv preprint arXiv:2311.15131*.

Jifan Chen, Grace Kim, Aniruddh Sriram, Greg Durrett, and Eunsol Choi. 2023. Complex claim verification with evidence retrieved in the wild. *arXiv preprint arXiv:2305.11859*.

Shiqi Chen, Miao Xiong, Junteng Liu, Zhengxuan Wu, Teng Xiao, Siyang Gao, and Junxian He. 2024a. In-context sharpness as alerts: An inner representation perspective for hallucination mitigation. *arXiv preprint arXiv:2403.01548*.

Zhongzhi Chen, Xingwu Sun, Xianfeng Jiao, Fengzong Lian, Zhanhui Kang, Di Wang, and Chengzhong Xu. 2024b. Truth forest: Toward multi-scale truthfulness in large language models through intervention without tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20967–20974.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Cheng-Han Chiang and Hung-Yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. 2024. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Alexander R Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2021. Qafacteval: Improved qa-based factual consistency evaluation for summarization. *arXiv preprint arXiv:2112.08542*.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. True: Re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3905–3920.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Philippe Laban, Tobias Schnabel, Paul N Bennett, and Marti A Hearst. 2022. Summac: Re-visiting nli-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

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

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. 2023. Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10056–10070.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report.

OpenAI. 2024. Hello gpt-4o.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643, Online. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen-tau Yih. 2023. Trusting your evidence: Hallucinate less with context-aware decoding. *arXiv preprint arXiv:2305.14739*.

Adi Simhi, Jonathan Herzig, Idan Szpektor, and Yonatan Belinkov. 2024. Constructing benchmarks and interventions for combating hallucinations in llms. *arXiv preprint arXiv:2404.09971*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across nlp tasks. *arXiv preprint arXiv:1909.11218*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vectara. 2023. vectarahallucination_valuation_model. `https://huggingface.co/vectara/hallucination_evaluation_model`. Accessed: 2024-06-12.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535.

Shaolei Zhang, Tian Yu, and Yang Feng. 2024. Truthx: Alleviating hallucinations by editing large language models in truthful space. *arXiv preprint arXiv:2402.17811*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019a. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019b. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

A Data Creation for *Lookback Lens*

Our experimental setup aims to evaluate the ability of *Lookback Lens* to detect hallucinations in large language models with attention maps. We consider the summarization task and question-answering (QA) task in data creation.

For the summarization task, we sampled 1,000 examples from the CNN/DM dataset (See et al., 2017). For QA, we use 2,655 examples from the Natural Questions (Kwiatkowski et al., 2019) from the setup of Liu et al. (2024) to mix the gold document with irrelevant documents. To keep our focus more on LLM hallucinations rather than being distracted by assessing LLMs' long-context utilization ability, we limited context to three documents per question where the gold document containing the answer was placed in the middle, surrounded by two irrelevant documents.

We prompt LLaMA-2-7B-Chat (Touvron et al., 2023) to generate correct responses by greedy decoding for both tasks to ensure that both hallucinated and non-hallucinated examples derive from the same source distribution. The max length of generation is set to 256 tokens, or until the EOS token is generated.

After the annotation was collected, we extract hallucinated and non-hallucinated spans, as well as the corresponding attention map lookback ratio, from the LLaMA-2-7B-Chat model, to train the *Lookback Lens* classifiers.

In the predefined span setting, three types of spans are considered as non-hallucinated spans: 1) the text segment before the first hallucinated span in the response 2) the text segment after the last hallucinated span in the response 3) the response annotated as non-hallucinated. All the annotated hallucinated spans are used as negative data to train the *Lookback Lens*.

In the sliding window setting, we consider all the possible fixed sized chunk with size = 8. If a chunk is overlapping with any of the annotated hallucinated spans, then it is considered as hallucinated, otherwise it is non-hallucinated.

Why not use existing data? Initially, we considered using the HaluEval dataset (Li et al., 2023), which was created by prompting GPT-3.5 (OpenAI, 2022) to generate "hallucinated examples" against human-annotated non-hallucinated responses, on summarization, QA, and dialogue tasks. However, we have concerns that their method introduces a bias by creating fundamentally different data distri-

butions between hallucinated and non-hallucinated examples. This discrepancy could potentially lead the classifier to learn to distinguish the sources of responses rather than accurately detecting hallucinations.

Additionally, we argue that the LLM's attention weight will be more meaningful if the text is generated by the same LLM itself, not from external sources and teacher forcing to obtain the attention weights. To ensure an unbiased and controlled evaluation environment, we generated our own dataset on summarization and QA tasks.

B Evaluation Details

B.1 Evaluation Prompt for GPT-4o

We show the templates used to prompt GPT-4o (`gpt-4o-2024-05-13`) in annotating the truthfulness of a response and the span-level hallucination segment prediction in Table 9 and Table 10, respectively for CNN/DM and Natural Questions.

This prompt is used for 1) collecting the data to train the *Lookback Lens* in Table 1, and 2) evaluating the XSum summarization task in Sections 3, 4, and 5. We also provide the approximate cost of GPT-4o calls (in USD):

- 1000 examples from XSum is around $8.
- 1000 examples from CNN/DM is around $12.
- 2655 examples from NQ is around $16.

Figure 4: Screenshot of human annotation interface.

B.2 Human Evaluation on GPT-4o Evaluation

Summarization To assess the quality of GPT-4o's evaluations, we initially conducted a pilot study using 70 XSum dataset examples, with native English-speaking authors and colleagues as evaluators. Evaluators received the document, ground truth summary, LLaMA-2-7B-Chat's summary, and GPT-4o's judgment to provide a binary judgment on GPT-4o's accuracy. Our interface is depicted in Appendix B.1 (see Figure 4). This initial evaluation affirmed the correctness of GPT-4o's judgments in 68 out of 70 cases. To further verify these results, we expanded our evaluation through Amazon MTurk, adding two additional annotations per example. Across all 210 evaluations (70 initial + 140 MTurk), only 9 annotations were marked incorrect, and in only 2 cases did a majority of annotators deem the judgment incorrect (marked incorrect by at least two annotators). With a final accuracy of 97.1%, and high intra-annotator agreement, the comprehensive evaluation supports GPT-4o's use as an automatic evaluator for the entire dataset.

Question Answering We expand the human evaluation to Natural Questions dataset using Amazon MTurk. The evaluation interface is copied from the summarization setup, but changing "summary" to "answer", as well as adding the "question" field.

We take 50 examples and assign each example to three different annotators. There are 7 annotations marked incorrect out of the 150 annotations. In total, 3 of the examples are marked incorrect by at least two annotators. If applying a majority vote, 47 out of 50 examples are correct, resulting in a 94.0% accuracy. This suggests that it is generally sufficient to use GPT-4o to verify the generated responses on the question-answering task.

```
You will be provided with a document and a proposed summary. Your task is to determine if the
proposed summary can be directly inferred from the document. If the summary contains any information
not found in the document, it is considered false. Even if the summary is different from a ground
truth summary, it might still be true, as long as it doesn't contain false information.
For each proposed summary, explain why it is true or false based on the information from the
document. Focus only on the original document's content, disregarding any external context.
After your explanation, give your final conclusion as Conclusion: True if the proposed summary is
completely accurate based on the document, or Conclusion: False if it contains any incorrect or
unsupported information. If your conclusion is 'False', identify the exact phrases or name entities
from the summary that is incorrect by stating Problematic Spans: [the inaccurate text spans from
the summary, in Python list of strings format].

#Document#: {document}

#Ground Truth Summary#: {ground_truth_summary}

#Proposed Summary#: {response}

Write your explanation first, and then give your final conclusion as Conclusion: True if
the proposed summary is completely accurate based on the document, or Conclusion: False if it
contains any incorrect or unsupported information. Add Problematic Spans: [the exact inaccurate
text spans from the summary, in a list of strings] if your conclusion is 'False'.
```

Table 9: Prompt template for GPT-4o in annotating the truthfulness and predicting span-level hallucinations on summarization tasks. Used for CNN/DM and XSum.

```
You will be provided with a document and a proposed answer to a question. Your task is to determine
if the proposed answer can be directly inferred from the document. If the answer contains any
information not found in the document, it is considered false. Even if the answer is different from
a ground truth answer, it might still be true, as long as it doesn't contain false information.
For each proposed answer, explain why it is true or false based on the information from the document.
Focus only on the original document's content, disregarding any external context.
After your explanation, give your final conclusion as Conclusion: True if the proposed answer is
completely accurate based on the document, or Conclusion: False if it contains any incorrect or
unsupported information. If your conclusion is 'False', identify the exact phrases or name entities
from the answer that is incorrect by stating Problematic Spans: [the inaccurate text spans from the
answer, in Python list of strings format].

#Document#: {document}

#Ground Truth Answers (a list of valid answers)#: {ground_truth_answers}

#Proposed Answer#: {response}

Write your explanation first, and then give your final conclusion as Conclusion: True if
the proposed answer is completely accurate based on the document, or Conclusion: False if it
contains any incorrect or unsupported information. Add Problematic Spans: [the exact inaccurate
text spans from the answer, in a list of strings] if your conclusion is 'False'.
```

Table 10: Prompt template for GPT-4o in annotating the truthfulness and predicting span-level hallucinations on question-answering tasks. Used for Natural Questions.

B.3 Evaluation Prompt for MT-Bench

We show the evaluation prompt for MT-Bench (hallucination) in Table 11. We follow standard practice for MT-Bench (original) evaluation[4] and show evaluation prompts in Table 12. We evaluate MT-bench (original) with their default GPT-4 model `gpt-4-0613` and our proposed MT-Bench (hallucination) with the latest GPT-4o model (`gpt-4o-2024-05-13`).

C Experiment Details

C.1 Model Details

State-of-the-art NLI Model We give further detail on the pretrained SoTA NLI model [5] used as our topline hallucination detector. Specifically, the model is based on DeBERTa-V3-base (He et al., 2021) and further finetuned on a range of NLI and summarization datasets with exam-

[4] https://github.com/lm-sys/FastChat/tree/main/fastchat/llm_judge.

[5] https://huggingface.co/vectara/hallucination_evaluation_model

```
Please act as an impartial judge and evaluate the faithfulness and consistency of the response
provided by an AI assistant to the user question displayed below. Your evaluation should consider
whether the assistant's answer to the second user question is faithful and consistent to the chat
history. If the answer contains any misinformation not found or not supported by the chat history,
it is considered a hallucination. You evaluation should focus on the assistant's answer to the
second user question. Begin your evaluation by providing a short explanation. Be as objective as
possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by
strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

<|The Start of Assistant A's Conversation with User|>

### User:
{question_1}

### Assistant A:
{answer_1}

### User:
{question_2}

### Assistant A:
{answer_2}

<|The End of Assistant A's Conversation with User|>
```

Table 11: GPT-4o evaluation prompt for MT-bench (hallucination).

ples annotated with factual consistency, including FEVER (Thorne et al., 2018), Vitamin C (Schuster et al., 2021) and PAWS (Zhang et al., 2019b). Roughly 731k data examples can be collected from the training set of the above three datasets. The model is reported to have superior performance when evaluated on TRUE (Honovich et al., 2022) SummaC Benchmark (Laban et al., 2022) and AnyScale Ranking Test for Hallucinations [6].

Other Model Details and License

- `Llama-2-7B-Chat`: A 7B parameter model that is instruction fine-tuned. HuggingFace ID: `meta-llama/Llama-2-7b-chat-hf`.
- `Llama-2-13B-Chat`: A 13B parameter model that is instruction fine-tuned. HuggingFace ID: `meta-llama/Llama-2-13b-chat-hf`.
- `hallucination_evaluation_model`: Based on `microsoft/deberta-v3-base` which has 86M parameters. HuggingFace ID: `vectara/hallucination_evaluation_model`.
- `DeBERTa-V3-Base`: a 86M parameters encoder based model. HuggingFace ID: `microsoft/deberta-v3-base`.

The above models have the following licenses.

- `Llama-2-7B-Chat` is under the Llama 2 Community License Agreement.
- `Llama-2-13B-Chat` is under the Llama 2 Community License Agreement.
- `vectara/hallucination_evaluation_model` is under the Apache 2.0 License.
- `DeBERTa-V3-Base` is under MIT License.

Inference Details We run all the models on NVIDIA A6000 (48GB) and V100 (32GB) GPUs. We do not train the model, but only run the inference part. Each of the examples takes around 20-30 seconds for 7B model, 40-60 seconds for 13B model to generate responses using our *Lookback Lens Guided Decoding*. Please check Appendix C.2 to estimate the total running time on each of the datasets, as it depends on number of examples.

All the inferences are run with either greedy decoding or sampling using temperature 0.9 and top-p sampling with $p = 0.95$. The implementation is based on Huggingface Transformers packages.[7] All the scores in the paper are from a single run due to the limited computation for the large models.

Classifier Training Details We use Scikit-Learn `sklearn.linear_model.LogisticRegression`[8]

[6] https://www.anyscale.com/blog/llama-2-is-about-as-factually-accurate-as-gpt-4-for-summaries-and-is-30x-cheaper

[7] https://github.com/huggingface/transformers

[8] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

```
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. You evaluation should focus on the assistant's answer to the second user question. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

<|The Start of Assistant A's Conversation with User|>

### User:
{question_1}

### Assistant A:
{answer_1}

### User:
{question_2}

### Assistant A:
{answer_2}

<|The End of Assistant A's Conversation with User|>
```

```
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question. Your evaluation should consider correctness and helpfulness. You will be given a reference answer and the assistant's answer. You evaluation should focus on the assistant's answer to the second question. Begin your evaluation by comparing the assistant's answer with the reference answer. Identify and correct any mistakes. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

<|The Start of Reference Answer|>

### User:
{question_1}

### Reference answer:
{ref_answer_1}

### User:
{question_2}

### Reference answer:
{ref_answer_2}

<|The End of Reference Answer|>


<|The Start of Assistant A's Conversation with User|>

### User:
{question_1}

### Assistant A:
{answer_1}

### User:
{question_2}

### Assistant A:
{answer_2}

<|The End of Assistant A's Conversation with User|>
```

Table 12: GPT-4 evaluation prompt for general questions (top) and math questions (bottom) on MT-bench (original).

to train the classifiers of *Lookback Lens* on CPU machine. We use all the default hyperparameters, such as L2 penalty, etc, but we change the `max_iter` to 1000 to ensure it is converged.

Heads Mapping Details We use Scikit-Learn `sklearn.linear_model.LinearRegression`[9] in Section 4, to fit a linear transformation from LLaMA-2-13B-Chat's attention heads to LLaMA-2-7B-Chat's attention heads. It is computed to solve the close-form Ordinary Least Squares optimization problem, without gradient descent. We use all the default hyperparameters and run it on our CPU machine.

C.2 Dataset Details

The datasets we used in the paper have the following details:

- CNN/DM: sampled 1000 examples from the testing set. Apache-2.0 license. https://huggingface.co/datasets/abisee/cnn_dailymail
- Natural Questions: Apache-2.0 license. Testing set: 2655 examples from https://github.com/nelson-liu/lost-in-the-middle. NQ-train: sampled 2499 examples from its training set, using the positive document provided by https://github.com/facebookresearch/DPR
- XSum: 1000 examples sampled from the testing set. MIT license. https://github.com/EdinburghNLP/XSum
- MT-bench: 80 examples. Apache-2.0 license. https://github.com/lm-sys/FastChat/tree/main/fastchat/llm_judge

D Additional Results

D.1 Visualization

We visualize the lookback ratio of the top-10 most positive/negative heads when LLaMA-2-7B-Chat decodes the answer for an NQ example. The top-10 most positive/negative heads are selected with the most positive/negative coefficients from the classifier. The green rectangle frames the part that contains the hallucinations, i.e. *and in Germany in the 14th century*. We can see that during the generation of the hallucinated span, the positive heads, especially for the top-1 heads (topmost), show a lower lookback ratio (in blue), while the negative heads show a slightly higher lookback ratio (in red). However, the behavior of *Lookback Lens* still needs to be determined by the collective behavior of all heads and the weight and bias of the classifier.

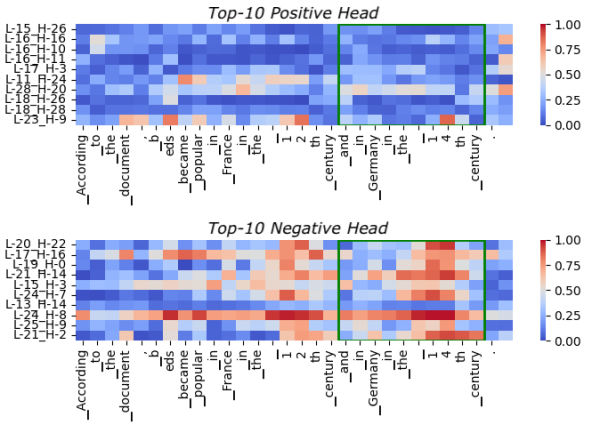

Figure 5: Top-10 positive/negative heads ranked from top to the bottom by the magnitude of their coefficients in the *Lookback Lens* classifier.

D.2 Using Multiple or All Layers for Hidden States

Multiple Layer We follow the prior study (Azaria and Mitchell, 2023) to use the layers with the best predictive power in hallucination detection: 32nd/28th/24th/20th layers. We concatenate the 4 layer features into a huge feature. Please note that the hidden dimension of LLaMA-7B is 4096, so combining 4 layers would result in a 16384-dim feature vector. In contrast, our *Lookback Lens* feature for the 7B model is only 1024-dim. Thus, the big classifier using 16384 input features is supposed to be more effective given that it uses 10x more features.

However, the result shown in Table 13 indicates that concatenating 4 layers is still less effective compared to our Lookback Lens.

All Layers We also try to use the hidden states from all layers, but concatenating them all will result in a huge feature vector with dimensions of more than 100k and make the classifier extremely slow in training. Thus, we perform max/average pooling for the features across different layers, resulting in 4096-dim feature vectors as the classifier inputs. The results shown in the table below are still worse than our *Lookback Lens* results.

The two experiments above indicate that using

[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Method	AUROC (sliding window = 8)	
	NQ → Sum.	Sum. → NQ
Residual outputs (hidden states)		
Layer 32	56.1	59.4
Layer 28	57.7	58.8
Layer 24	58.3	58.3
Layer 20	57.6	59.5
Concatenate above 4 layers	58.8	59.2
Max pooling all 32 layers	56.7	59.2
Average pooling all 32 layers	57.3	59.2
Ours: Lookback Lens	66.1	66.0

Table 13: AUROC results for different methods of utilizing hidden states.

Method	AUROC (sliding window = 8)	
	NQ → Sum.	Sum. → NQ
Attention block outputs		
Layer 32	57.6	60.7
Layer 28	58.5	57.2
Layer 24	56.3	57.2
Residual outputs (hidden states)		
Layer 32	56.1	59.4
Layer 28	57.7	58.8
Layer 24	58.3	58.3
Ours: Lookback Lens	66.1	66.0

Table 14: AUROC results for different layers and outputs.

multiple or all layers may not be the key to making the classifier accurate. Instead, by designing good features like lookback ratio, the compact 1024-dim feature can be even more effective compared to the 10x bigger high-dimensional hidden state features.

D.3 Comparing Attention Outputs with Hidden States

Some papers mention that attention block outputs could be more useful for detecting hallucinations (Campbell et al., 2023; Li et al., 2024), while our main experiments only consider the hidden states as input features for detecting contextual hallucinations. Here we include additional experiment results that use attention block outputs instead. In Table 14, we show that there is no significant difference when switching to attention block outputs, and our *Lookback Lens* still outperforms these baselines.