Towards Long Context Hallucination Detection

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable performance across various tasks. However, they are prone to hallucination, generating information that is either unsubstantiated or contradictory to the given context. Although many studies have investigated hallucinations in LLMs, addressing hallucinations in long-context inputs remains an open problem. In this work, we take an initial step toward solving this problem by constructing a dataset specifically designed for long-context hallucination detection. Furthermore, we propose a novel architecture that enables pre-trained encoder models, such as BERT, to process long contexts and effectively detect contextual hallucinations through a decomposition and aggregation mechanism. Our experimental results show that the proposed architecture significantly outperforms previous models of similar size and performs on par with LLM-based models while providing substantially faster inference.

1 Introduction

Large language models (LLMs) have demonstrated potential in generative and knowledge-intensive tasks, such as question-answering (QA) and summarization. Despite these advancements, their practical deployment presents notable challenges, particularly due to the issue of "hallucination," wherein models generate content that appears plausible but is factually incorrect or nonsensical.

Previous research has studied hallucination detection mainly through the lens of Natural Language Inference (NLI): given a pair of input texts context and response, a generated response is considered faithful and free of hallucinations only when it is logically entailed by the context (Maynez et al., 2020; Kryscinski et al., 2020; Fabbri et al., 2021; Zha et al., 2023). Some studies explore hallucination detection by training small, encoder models like BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019) on NLI datasets (Kryscinski et al., 2020; Zha et al., 2023); some other studies take a LLM-based approach and prompt LLMs to assess whether hallucinations are present (Chang et al., 2024; Hu et al., 2024). However, both lines of work encounter challenges when addressing longer contexts. For instance, BERT-based models for hallucination detection are constrained by a maximum input length of 512 tokens, while LLM-based prompting for evaluating the faithfulness of responses to long contexts is not only expensive but also empirically suboptimal (Kim et al., 2024).

In this work, we introduce a novel architecture that enables pre-trained encoder models, such as BERT, to process long contexts and effectively detect contextual hallucinations through a decomposition and aggregation mechanism. Our model begins by decomposing the long input contexts and responses into smaller chunks. It then generates deep representations for each chunk using a backbone encoder model. Finally, it aggregates these chunk-level representations through a learned attention and pooling layer to create a holistic representation of both the context and response chunks to evaluate hallucination. Due to the scarcity of available datasets in long-context hallucination detection, we develop a prompting workflow that introduces hallucinations into an existing long document summarization dataset, BookSum (Kryściński et al., 2022), to empirically evaluate our proposed architecture. Our experimental results demonstrate that the proposed architecture significantly outperforms prior models of similar size and achieves performance comparable to LLM-based models while offering substantially faster inference.

2 Problem Definition

In this work, we investigate the problem of long-context hallucination detection. Our objective is to develop a model that can effectively and efficiently detect hallucinations given a pair of input texts: a context and a corresponding response. Specifically, we focus on cases where the context is long-form, which presents additional challenges for models in terms of processing and making inferences within a short time frame.

We define the hallucinations under study as follows: given a document, a response is considered to contain hallucinations if and only if (a) it introduces unsubstantiated information that is not grounded in the context, or (b) it presents information that contradicts the context. The models are expected to perform a binary classification to determine whether the response hallucinates relative to the context, regardless of the specific type of hallucination.

To empirically evaluate our models within this problem setting, we conduct experiments on the task of long-document summarization, where the context consists of a long document about a book and the response is a corresponding summary. However, we posit that our hallucination injection framework and model design can also generalize to other domains involving long-context hallucination detection such as dialogue systems.

3 Dataset Collection

We consider the task of book summarization to support our experiments and construct our dataset from BookSum (Kryściński et al., 2022). This dataset includes varying levels of document-summary pairs, including book-level, chapter-level, and paragraph-level pairs. In our study, we focus on chapter-level document-summary pairs, as they align more closely with our research interests. Chapter-level documents have on average 5,101 tokens, and summaries have on average 505 tokens. The dataset only provides expert written, ground-truth summaries for the different levels of documents. We synthesize a hallucinatory subset by injecting some hallucination for certain pairs in the dataset. To create a balanced dataset, we introduce hallucinations with a 50% probability while iterating through the dataset. Each time we introduce a hallucination, we randomly select one type of hallucination from the two categories introduced in Section 3.1. The statistics of our dataset is shown in Table 1

Split	# of Examples	% of hallucinations
Train	5,653	51%
Dev	854	48%
Test	950	52%

Table 1: The statistics of our constructed dataset.

3.1 Hallucination Injection

We develop a prompting workflow that supports us to introduce hallucination to our dataset of long document summarization. We consider two following types of hallucination as introduced in Section 2. The exact prompts we use for this process are shown in Appendix C.

Baseless Information Hallucination We prompt GPT-4o to *"add a complete sentence that is related to the topic but introduces some new information you make up ..."*.

Contradictory Information Hallucination We prompt GPT-4o to *"rewrite one sentence completely so that it utterly contradicts from its original sentence ..."*.

3.2 Dataset Verification

To assess the quality of the annotations, we randomly sample 20 examples from our dataset and evaluate whether hallucinations are present in the summaries. We then compare our annotations with those in the generated dataset, resulting in a Cohen's kappa agreement of 0.9, indicating a high level of alignment between our generated data and human judgments.

We also employ Perplexity score as an estimate to automatically measure the coherence and fluency of the summary after our introduction of hallucination. Perplexity is defined as the exponentiated average negative log-likelihood of a sequence and is popularly used as a measure to evaluate the performance of a language model as well as the quality of generations. It quantifies how well a probabilistic model predicts a sequence of words. A lower perplexity score indicates that the language model assesses the sequence of text as being more aligned with its predicted probabilities, reflecting better coherence and fluency. We calculate the perplexity score of a summary as follows: $\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i)\right)$.

We utilize Llama-3.2-1B to compute the average perplexity scores for both the original summaries and the summaries after the introduction of hallucination. Interestingly, we observe that the average perplexity score decreases from 18.52 to 18.26 after the injection of hallucinations, indicating a high

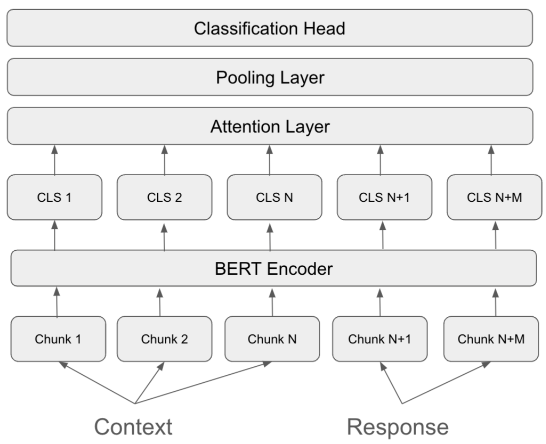

Figure 1: The structure of our proposed architecture. In the attention layer, we add a new token of [CLS] at the beginning of all chunk-level CLS representations to be used as a pooled representation for the whole input, and a [SEP] between the context chunk representations and the response chunk representations to distinguish them.

quality of our data augmentation process.

4 Our Method

The primary obstacle preventing BERT-based models from effectively processing long documents is the computation of the full quadratic attention matrix, which incurs $O(n^2)$ time and memory complexity, where n represents the input sequence length. Intuitively, each token must attend to all other tokens to develop robust representations of the input texts. To tackle this challenge, we propose an architecture that employs a decomposition and aggregation strategy. The structure of our model is shown in Figure 1. Given a pair of input texts—context and response—we first decompose them into fixed length chunks for both the context and response. Each chunk is then processed through a pre-trained BERT encoder to obtain their corresponding CLS representations. Subsequently, we employ an attention layer to learn which chunks are most prominent for assessing the presence of hallucinations in the response with respect to the context. Finally, we utilize a pooling layer to obtain a holistic representation of all chunks for the purpose of classification. We provide further experimental details regarding chunk sizes, the number of chunks, and various other hyperparameters and architectural design choices in Section 5 and Appendix A.

Our proposed architecture offers several advantages: 1. Our framework does not necessitate any further pretraining and can be implemented on top of existing encoder models. In contrast, previous approaches for long-context processing, such as Hierarchical Attention Transformer (HAT) (Chalkidis et al., 2022) or Longformer (Beltagy et al., 2020) require pretraining on long-form texts, which can be computationally expensive. Our model circumvents this requirement, enabling the use of any encoder model as the backbone for fine-tuning on domain-specific tasks, such as long-context hallucination detection. 2. Theoretically, our model can accommodate very long contexts by continually adding layers of decomposition and aggregation (one layer can process up to 512 chunks $\times 512$ chunk size of tokens). Given a fixed chunk length c (e.g. 512), the computation complexity of our model is $O(k^2)$, where k denotes the number of chunks and $k = \frac{n}{c}$. This represents a significant improvement over the $O(n^2)$ complexity of BERT.

5 Experiment

We conduct experiments using our constructed dataset and compare the performance of our proposed model with that of previous approaches.

5.1 Models

Longformer Longformer is a modified Transformer architecture with a self-attention operation that scales linearly with the sequence length, making it versatile for processing long documents (Beltagy et al., 2020). We finetune a pre-trained Longformer model using our dataset for model comparison.

Hierarchical Attention Transformer (HAT) Hierarchical Attention Transformers (HATs) employ a multilevel attention mechanism consists of segment-wise attention followed by cross-segment attention to effectively handle long documents (Chalkidis et al., 2022). We finetune a pre-trained HAT model using our dataset for our experiments.

Alignscore Alignscore is a RoBERTa model trained on a general function that assesses the information alignment between two arbitrary text pieces. Its training incorporates a wide range of data sources, resulting in 4.7 million training examples derived from seven well-established tasks: Natural Language Inference (NLI), Question Answering (QA), paraphrasing, fact verification, information retrieval, semantic similarity, and summarization. (Zha et al., 2023). The model can infer

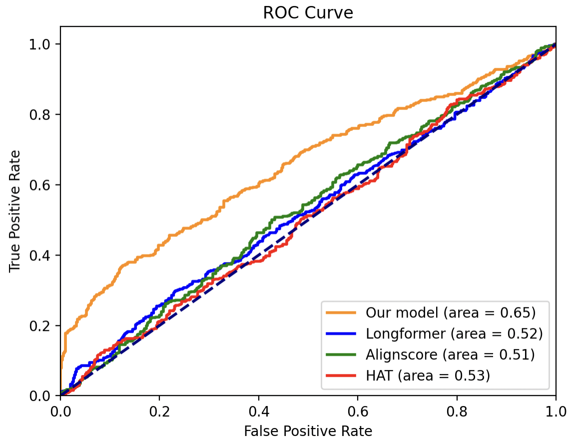

Figure 2: ROC AUC Results

Model	PRECISION	RECALL	LATENCY
HAT	48.42	70.55	**41.01**
Longformer	47.89	**87.47**	18.15
Alignscore	50.09	60.00	1.44
Refchecker	52.13	51.21	0.15
GPT-4o	<u>53.11</u>	<u>78.68</u>	0.79
Our Model	**54.50**	73.19	<u>18.62</u>

Table 2: Results of all of the models we tested. Latency is computed as the number of samples processed per second at inference time, the higher the faster. The **bolded** numbers represent the best performance across all models and the <u>underlined</u> numbers represent the second best. See more details about hyperparameter choices as well as how latency is computed in Appendix A.

with arbitrarily long texts; however, it cannot be trained on texts longer than 512 tokens. The authors also present it as an off-the-shelf metric, given that it has been trained on a substantial amount of factual consistency data. Therefore, we evaluate the model off-the-shelf without any additional training in this study.

RefChecker RefChecker introduces claim-triplets to represent claims in LLM responses, aiming to detect fine-grained hallucinations (Hu et al., 2024). This framework first prompts an LLM to extract claims from the response, and then prompt an LLM another time to compare each of the claim to the context to predict hallucination. We use GPT-4o-mini as the LLM backbone for both the extractor and checker in their framework.

GPT-4o We zero-shot prompt GPT-4o-mini with specific instructions and definitions of our task to predict hallucinations as a strong baseline. The exact prompt that we use is shown in Appendix C.

Our Model The structure of our model is described in Section 4. More experimental details about our model are discussed in Appendix A.

5.2 Results

We present the Receiver Operating Characteristic (ROC) Curve and the ROC Area Under the Curve (AUC) score in Figure 2. Due to the black-box nature of LLM-based models, we are unable to obtain their predicted scores, so only the results from encoder models are displayed. We see that all baseline models lack discriminative ability in terms of detecting hallucination with long context: state-of-the-art metrics in factual consistency evaluation like AlignScore fail to adapt to long-form texts; Longformer and HAT also exhibit insufficient expressive capacity to distinguish hallucinations, despite being pre-trained on long-form texts and then finetuned on the same training set as our model utill converged. In contrast, our model demonstrates strong performance on this task, without any pre-training on long-form or factual consistency data.

We show the precision, recall score and inference latency of our model and all baseline models in Table 2. Notably, Longformer exhibits high recall but low precision, indicating that it tends to overpredict the positive class, leading to a high number of false positives. Additionally, while Refchecker takes considerably more time for inference by extracting and verifying individual claims, it performs worse than GPT-4o, despite using the same backbone LLM. This suggests that traditional approaches to hallucination detection, which rely on splitting inputs into claims and verifying each claim to produce an aggregated score, may not be as effective when applied to long-context inputs. This observation aligns with the suboptimal performance of AlignScore on our dataset, as its approach mirrors this method. Our model, on the other hand, matches GPT-4o in precision and recall but achieves 20x faster inference times, making it more applicable for real-world deployment. More details of how we measure the inference latency are discussed in Appendix A.

6 Conclusion

We construct a dataset and propose a new architecture to study long context hallucination detection. We will release our code and data for further research.

Limitations One limitation of our work is that our proposed model requires in-domain training for a specific domain. This is different from prompting with LLMs. However, our proposed prompting workflow of hallucination injection makes it easy to obtain high-quality training data for other domains (e.g. dialogue) as well to support the training of our model in these areas, and then our model will have faster inference time in deployment with on par performance with strong LLMs.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Ilias Chalkidis, Xiang Dai, Manos Fergadiotis, Prodromos Malakasiotis, and Desmond Elliott. 2022. An exploration of hierarchical attention transformers for efficient long document classification. *Preprint*, arXiv:2210.05529.

Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Booookscore: A systematic exploration of book-length summarization in the era of llms. *Preprint*, arXiv:2310.00785.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models. *Preprint*, arXiv:2405.14486.

Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Fables: Evaluating faithfulness and content selection in book-length summarization. *Preprint*, arXiv:2404.01261.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. *Preprint*, arXiv:2105.08209.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

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

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. Alignscore: Evaluating factual consistency with a unified alignment function. *Preprint*, arXiv:2305.16739.

A Experiment Details

A.1 Training Details

We train our model with the Huggingface Transformers and Accelerate package. We use Amazon Elastic Compute Cloud (Amazon EC2) for our training experiments. We use one p4d.24xlarge instance for the training. It has 8 NVIDIA A100 GPUs with 40.0 GB GPU memory each. The optimal hyperparamters we find for our model is 40 chunks in total, 32 for context and 8 for response, each with a chunk size of 256. We train our model with 2e-6 learning rate, 0.1 weight decay, 1000 warm up steps, and 100 epochs. We train with only the first 1,000 examples for our model as it already shows good performance in the validation set. We use pre-trained Roberta-large as our backbone encoder model and a randomly initialized Roberta Attention layer. All parameters in the architecture are being optimized. In the attention layer, we add a new token of [CLS] at the beginning of all chunk-level CLS representations to be used as a pooled representation for the whole input, and a [SEP] between the context chunk representations and the response chunk representations to distinguish them.

A.2 Inference Latency

HAT, Longformer, and our model inference with 8 GPUs (data parallel) with a batch size of 4. However, the codebase provided by the authors of Alignscore doesn't support multi-gpu inference with longer texts and also doesn't support batching. So the inference latency of AlignScore is computed as their inference time with one gpu and batch size of one multiplied by 32 as an estimate. Inference time of GPT-4o and Refchecker depends on API calls to OpenAI and may differ from time to time due to network, API availability, and some other reasons.

B Dataset Examples

The whole chapter is too long to present, so here we show examples of original summary from the BookSum dataset, as well as summary after our hallucination injection. We highlight the specific sentence that was rewritten or added in different colors.

Original Summary Any state–old, new, whatever–needs good laws and good armed forces. Since you can't have good armed forces without good law, let's just say you need a good army. There are four types of armies you could have: a local army, mercenaries, auxiliaries , or some kind of mixture. First things first: mercenaries and auxiliary armies are useless. Just don't do it. Mercenaries are only interested in the money and are not reliable. That's how Italy got into trouble–occupation by France and Spain–in the first place. Plus, if a mercenary leader is good then you have to be afraid that he will turn against you, and if he is bad he will make you lose anyway. So, no good. Good armies? Citizen armies. Look at Rome. Look at Sparta! Carthage used mercenaries, and guess what happened to them? They got owned by Philip of Macedonia, Alexander the Great's dad. Don't be like Carthage. Machiavelli gives us a bunch of examples, but the basic takeaway here is mercenaries = bad. They are lazy. They are expensive. They kill during wartime. And they don't even defend their camps. Machiavelli has a little bit of an axe to grind about this problem, since he has wanted Italy to stop relying on mercenaries forever, but no one would listen to him.

Summary After Injection of Contradictory Information Any state–old, new, whatever–needs good laws and good armed forces. Since you can't have good armed forces without good law, let's just say you need a good army. There are four types of armies you could have: a local army, mercenaries, auxiliaries, or some kind of mixture. First things first: mercenaries and auxiliary armies are essential. Always rely on them. Mercenaries are only interested in the money and are not reliable. That's how Italy got into trouble–occupation by France and Spain–in the first place. Plus, if a mercenary leader is good then you have to be afraid that he will turn against you, and if he is bad he will make you lose anyway. So, no good. Good armies? Citizen armies. Look at Rome. Look at Sparta! Carthage used mercenaries, and guess what happened to them? They got owned by Philip of Macedonia, Alexander the Great's dad. Don't be like Carthage. Machiavelli gives us a bunch of examples, but the basic takeaway here is mercenaries = bad. They are lazy. They are expensive. They kill during wartime. And they don't even defend their camps. Machiavelli has a little bit of an axe to grind about this problem, since he has wanted Italy to stop relying on mercenaries forever, but no one would listen to him.

Original Summary Chapter X is entitled "How to Measure the Strength of Any Prince's State. Here Machiavelli adopts a decidedly militaristic tone. Princes, he writes, are better off when they can assemble an army and stand up against attackers; once again, Cesare Borgia is cited as a perfect example. Machiavelli addresses the majority of this chapter to the other class of princes: "those who can't take the field against their foes, but have to hide behind their walls and defend themselves there. What should these more vulnerable princes do. They should keep their cities well-fortified; they should ignore the rural areas and focus their defense efforts on the urban centers; and they should be careful not to earn the people's hatred. A prudent prince is able to keep his subjects loyal to him and in good spirits during a siege. The burden during a siege is often on the besieger; he can almost never afford to wage a siege and do nothing else for a year. Defense, therefore, can consist of slowing the attacker down, wearing him out. Machiavelli cites the cities in Germany as examples of good fortification. These cities have moats, walls, artillery, public warehouses of food, drink, and fuel, and large supplies of raw materials in reserve to keep workers busy and economies going during a siege

Summary After Injection of Baseless New Information Chapter X is entitled "How to Measure the Strength of Any Prince's State." Here Machiavelli adopts a decidedly militaristic tone. Princes, he writes, are better off when they can assemble an army and stand up against attackers; once again, Cesare Borgia is cited as a perfect example. Machiavelli addresses the majority of this chapter to the other class of princes: "those who can't take the field against their foes, but have to hide behind their walls and defend themselves there." What should these more vulnerable princes do? They should keep their cities well-fortified; they should ignore the rural areas and focus their defense efforts on the urban centers; and they should be careful not to earn the people's hatred. He notes that a well-designed urban area can serve as a formidable defense mechanism, with strategically placed fortifications and supply depots. A prudent prince is able to keep his subjects loyal to him and in good spirits during a siege. The burden during a siege is often on the besieger; he can almost never afford to wage a siege and do nothing else for a year. Defense, therefore, can consist of slowing the attacker down, wearing him out. Machiavelli cites the cities in Germany as examples of good fortification. These cities have moats, walls, artillery, public warehouses of food, drink, and fuel, and large supplies of raw materials in reserve to keep workers busy and economies going during a siege.

C GPT-4o Prompts

Prompts Used to Introduce Baseless Information Hallucination "Add a complete sentence that is related to the topic but introduces some new information you make up. You can add the sentence anywhere in the paragraph but make sure it is a complete sentence and the paragraph is coherent. Reply with the whole paragraph that includes the sentence you added."

Prompts Used to Introduce Contradictory Information Hallucination "Given the paragraph, rewrite one sentence completely so that it utterly contradicts from its original sentence. You can choose any sentence in the paragraph but make sure the paragraph is still coherent and now has a claim that contradicts the original paragraph. Reply with the whole paragraph after the change."

Prompts Used to Run GPT-4o-mini Experiments "You will be given a document and a summary. Your task is to determine whether the summary is faithful or unfaithful to the information provided in the document. If the summary contains any statements that contradict the information given in the document, or if it includes information not present or implied by the document, reply 'unfaithful'. Otherwise, reply 'faithful'."