InterpDetect: Interpretable Signals for Detecting Hallucinations in Retrieval-Augmented Generation

Likun Tan, Kuan-Wei Huang, Joy Shi, Kevin Wu*
Pegasi AI, NYC
likun,kuan-wei,joy,kevin@usepegasi.com

Abstract

Retrieval-Augmented Generation (RAG) integrates external knowledge to mitigate hallucinations, yet models often generate outputs inconsistent with retrieved content. Accurate hallucination detection requires disentangling the contributions of external context and parametric knowledge, which prior methods typically conflate. We investigate the mechanisms underlying RAG hallucinations and find they arise when later-layer FFN modules disproportionately inject parametric knowledge into the residual stream. To address this, we explore a mechanistic detection approach based on *external context scores* and *parametric knowledge scores*. Using Qwen3-0.6b, we compute these scores across layers and attention heads and train regression-based classifiers to predict hallucinations. Our method is evaluated against state-of-the-art LLMs (GPT-5, GPT-4.1) and detection baselines (RAGAS, TruLens, RefChecker). Furthermore, classifiers trained on Qwen3-0.6b signals generalize to GPT-4.1-mini responses, demonstrating the potential of `proxy-model evaluation`. Our results highlight mechanistic signals as efficient, generalizable predictors for hallucination detection in RAG systems. [2]

1 Introduction

Large language models (LLMs) achieve strong performance across tasks such as question answering, summarization, and code generation [1, 2, 3], yet they frequently generate *hallucinations*—outputs that are factually incorrect or unsupported by evidence [4]. Retrieval-Augmented Generation (RAG) aims to mitigate hallucinations by grounding outputs in external knowledge [5], but models may still produce responses that contradict retrieved content [6, 7, 8, 9]. Detecting these hallucinations is crucial in high-stakes domains such as healthcare, finance, and education.

Recent work has advanced RAG-specific hallucination detection along two axes. First, token- and span-level corpora and toolkits, such as RAGTruth and LettuceDetect, enable fine-grained supervision and efficient classifiers [10, 11], while multilingual shared tasks (Mu-SHROOM/SemEval-2025) and pipelines (HalluSearch) foster robust, cross-lingual detection [12, 13]. Second, mechanistic and attribution-based approaches, including ReDeEP and LRP4RAG, analyze internal activations to disentangle parametric (internal) knowledge from retrieved (external) context, revealing that overactive FFN pathways and attention mis-weighting often correlate with hallucinations [8, 14, 15]. Surveys further synthesize best practices and open challenges in RAG hallucination detection [16, 17].

Building on this literature, we explore the use of **External Context Score (ECS)** and **Parametric Knowledge Score (PKS)**, computed across layers and attention heads following the ReDeep framework, as predictive features for hallucination detection in the domain of financial question

*Corresponding Author

[2] Our code and data are available at `https://github.com/pegasi-ai/InterpDetect`.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Workshop on Mechanistic Interpretability.

and answering. Mechanistically, ECS quantifies how much a model's output relies on retrieved information, computed via attention weights and semantic similarity between attended context and generated tokens. High ECS indicates strong grounding in external knowledge, reducing hallucination risk. PKS reflects contributions from feed-forward networks (FFNs) and attention pathways that inject parametric knowledge into the residual stream; over-reliance can drive hallucinations. Together, ECS and PKS decompose generation into external versus internal sources, enabling fine-grained analysis of hallucination origins in RAG models.

Using Qwen3-0.6b as the base model, we compute ECS and PKS across layers and attention heads, and use them as input features to train regression-based classifiers for hallucination detection. We further demonstrate the effectiveness of `proxy-model evaluation` by applying classifiers trained on Qwen3-0.6b to outputs from larger models, such as GPT-4-mini, achieving comparable performance with substantially lower computational resource.

Our contributions are threefold:

1. We extend the ReDeEP framework for computing *external context scores* (ECS) and *parametric knowledge scores* (PKS) with a fully open-sourced implementation built on TransformerLens [18], enabling compatibility with any TransformerLens-supported model without modifying the underlying model library.
2. We conduct a systematic evaluation of multiple regression-based classifiers, identifying the optimal model for hallucination prediction.
3. We demonstrate that leveraging a 0.6b-parameter model as a proxy allows effective and economical computation, facilitating practical application to large-scale, production-level models.

This work highlights the utility of mechanistic signals as reliable, low-cost indicators for hallucination detection, advancing the safe and trustworthy deployment of LLMs.

2 Related Work

Hallucination Detection in RAG: Hallucination detection methods span black-box classifiers, attribution, uncertainty modeling, and fine-tuning pipelines. Token-level approaches such as LettuceDetect[11] achieve efficient span-level detection but generalize poorly across models. Attributional methods like LRP4RAG[15] provide interpretability via relevance propagation, yet incur high computational cost. Uncertainty-based methods (e.g., FRANQ[19]) quantify response faithfulness but do not explain causal mechanisms, while fine-tuning pipelines such as RAG-HAT[20] require costly data and training. Our work builds on ReDeEP [8], the first mechanistic framework for RAG hallucinations, which disentangles parametric (internal) and contextual (retrieved) contributions through *external context* and *parametric knowledge* scores. We extend this line by providing a TransformerLens-based implementation, systematically benchmarking regression-based classifiers, and showing that signals extracted from a 0.6b model transfer effectively to larger models. This retains ReDeEP's interpretability while improving scalability and practicality for deployment.

Parametric vs. External Knowledge: Recent mechanistic interpretability studies have elucidated how LLMs balance *parametric knowledge* (FFN-stored internal memory) and *external context* (retrieved information via attention heads). Hallucinations often arise when later-layer FFNs over-inject parametric knowledge into the residual stream while attention underweights retrieved content, causing misalignment between internal and external sources [8]. Models also exhibit a "shortcut" bias, over-relying on retrieved context even when parametric knowledge is complementary [21]. Knowledge is structured across neurons, with semantically related information clustered in FFNs and attention heads [22, 23]. While our work focuses on RAG hallucination detection, these findings support broader applications such as targeted knowledge editing and controlled grounding, enhancing interpretability and reliability in high-stakes LLM deployments [24].

3 Methodology

The main stages of our methodology are illustrated in Figure 1. We first follow a standard RAG pipeline to generate a response given a query and retrieved document. The model's *parametric*

knowledge is treated as internal knowledge, elicited by prompting the LLM with the query. For each context-response span pair, we compute the *External Context Score* (ECS) for attention heads and the *Parametric Knowledge Score* (PKS) for feed-forward network layers, capturing contributions from external and internal knowledge, respectively. After confirming alignment with hallucination labels via correlation analysis, these scores are used as features for binary classifiers to detect hallucinated spans, which are then aggregated to yield response-level predictions. The following sections provide a more detailed discussion of each step.

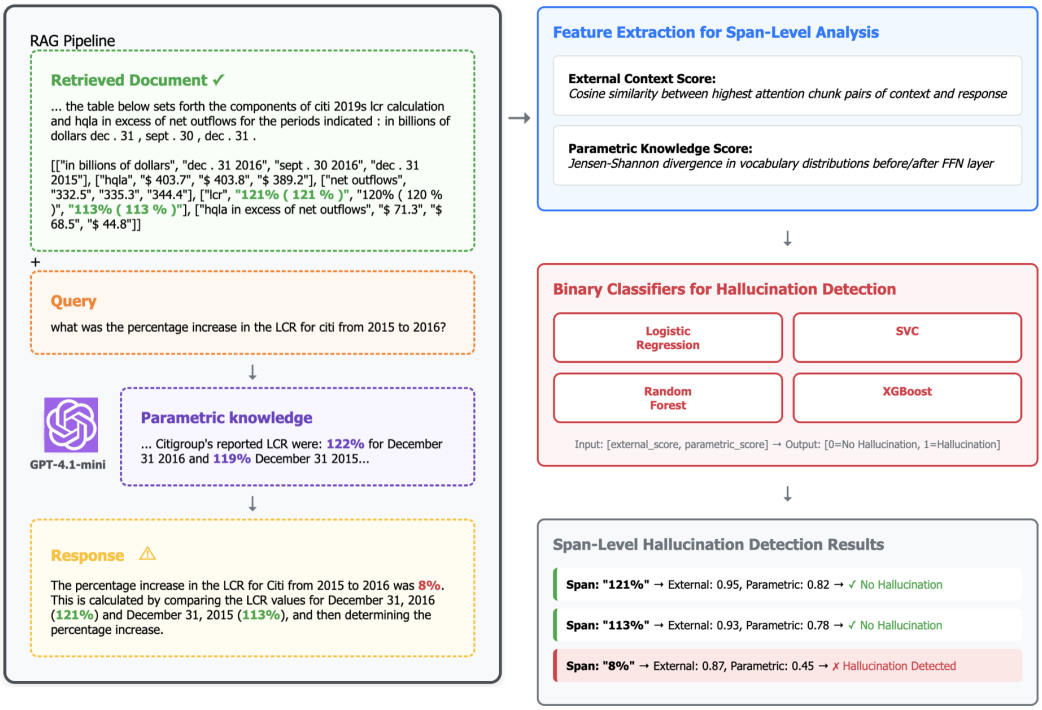

Figure 1: Pipeline for Span-Level Hallucination Detection

3.1 Data Curation

We utilize the FinQA dataset from RagBench [25] as our primary source. The dataset comprises 12.5k training instances and 2.29k test instances. Each instance includes a document drawn from the financial reports of S&P 500 companies, a question whose answer is grounded in the document, and a response originally generated by gpt-3.5-turbo and claude-3-haiku. For the purposes of our study, we regenerate responses using Qwen3-0.6b—the model consistently employed for computing interpretability scores—as well as GPT-4.1 mini, in order to examine whether the proposed method extends to hallucination detection in outputs from alternative models, thereby reflecting a more practical setting. Our data generation is divided into three steps as follows:

- **Response Generation:** We subsample 3,000 training instances from the original dataset. Responses are generated using Qwen3-0.6b and GPT-4.1 mini, with each model producing outputs separately.
- **Labeling:** To train the hallucination detection model, we need to identify spans in responses that contain hallucinated content relative to the source document. We first use LettuceDetect [11] for span-level labeling. To address potential errors, we add two LLM-based judges: llama-4-maverick-17b-128e-instruct and gpt-oss-120b, which evaluate responses generated by Qwen3-0.6b. For responses from the GPT family, we replace gpt-oss-120b with claude-sonnet-4 to reduce family-specific bias. We then apply majority voting at the response level. A sample is kept if at least one additional judge agrees with the LettuceDetect label. After filtering, 1,852 samples remain for downstream analysis.

- **Chunking:** To improve the accuracy of external context and parametric knowledge scores, we compute them at the span level. Therefore, we chunk both the retrieved document and the response. For the retrieved document, we use the existing `documents_sentences` from the original dataset. For responses, we split them into individual sentences.

An example from data preprocessing is provided in Appendix A.

3.2 Mechanistic Metrics

Our work draws on concepts from mechanistic interpretability research. Specifically, we leverage TransformerLens, an open-source library that provides access to internal model parameters—such as attention heads, feed-forward network layers (FFNs), and residual streams—in GPT-like models. In these models, residual connections allow each layer to incrementally update the hidden state using information from attention heads and FFNs. For a detailed description of the architecture, we refer the reader to the original TransformerLens work [18]. We primarily use TransformerLens to compute two metrics: the *External Context Score* and the *Parametric Knowledge Score*, following the definitions in ReDeEP [8]. While the original ReDeEP framework supports both token-level and chunk-level hallucination detection, computing scores at the token level is computationally expensive and does not fully capture context. Therefore, we restrict our calculations to the chunk level for both metrics.

External Context Score: The External Context Score (ECS) quantifies the extent to which a language model leverages external context when generating a response. In mechanistic interpretability, attention heads are responsible for retrieving relevant information from the context. To measure this utilization, ECS captures the semantic alignment between response segments and the context chunks most strongly attended to by the model.

Let the external context be partitioned into chunks $\{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_M\}$, and the generated output into response chunks $\{\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_N\}$, corresponding to `prompt_spans` and `response_spans` in Appendix A. For each attention head h at layer l, the most relevant context chunk for each response chunk $\tilde{r}_j$ is identified as

$$\tilde{c}_j^{\ell,h} = \arg\max_{\tilde{c}_i} A(\tilde{r}_j^{\ell,h}, \tilde{c}_i^{\ell,h}),$$

where A denotes the token-level attention weight matrix, and l and h indicate the layer and head indices. The chunk-level ECS is defined as the cosine similarity between the embeddings of $\tilde{r}_j$ and its corresponding context chunk $\tilde{c}_j$:

$$\text{ECS}_{\tilde{r}_j}^{\ell,h} = \cos\left(e(\tilde{r}_j^{\ell,h}),\, e(\tilde{c}_j^{\ell,h})\right),$$

where $e(\cdot)$ represents the embedding function, and $\cos(\cdot,\cdot)$ denotes cosine similarity.

Parametric Knowledge Score: The Parametric Knowledge Score (PKS) quantifies the extent to which the FFN contributes to *parametric knowledge*, i.e., knowledge stored in the model's weights, as opposed to information coming from external context. This is done by measuring the difference between residual stream states **before** the FFN layer and **after** the FFN layer.

Since the residual stream itself does not directly indicate "which token is being suggested"—it is a latent vector—we map it through the unembedding/projection matrix to the vocabulary distribution. This allows us to observe how the residual change after the FFN influences predicted token probabilities.

We then apply the Jensen-Shannon divergence (JSD) to compute the distance between the two vocabulary distributions, which defines the token-level PKS. The chunk-level PKS is computed by averaging the token-level PKS over all tokens in the chunk. Mathematically, this is expressed as

$$\text{PKS}_{t_n}^{\ell} = \text{JSD}\left(p(x_n^{\text{mid},\ell}),\, p(x_n^{\ell})\right), \quad \text{PKS}_{\tilde{r}}^{\ell} = \frac{1}{|\tilde{r}|} \sum_{t_n \in \tilde{r}} \text{PKS}_{t_n}^{\ell}.$$

where $p(\cdot)$ denotes the mapping from residual stream states to vocabulary distributions, and $x_n^{\text{mid},\ell}$ and x_n^{ℓ} refer to the residual stream states before and after the FFN layer, respectively. $\text{PKS}_{t_n}^{\ell}$ and $\text{PKS}_{\tilde{r}}^{\ell}$ stand for token-level and chunk-level PKS, respectively.

More details about the computation is given in Appendix B.

3.3 Classifier for Hallucination Detection

Hallucination detection is formulated as a binary classification task. As input features, we use the **External Context Score** (ECS) computed for each attention head and layer, together with the **Parametric Knowledge Score** (PKS) computed for each layer. Prior to classification, features are standardized using `StandardScaler` and refined via `SmartCorrelatedSelection` to remove redundant or highly correlated features. We evaluate four classifiers—Logistic Regression, Support Vector Classification (SVC), Random Forest, and XGBoost—and select the model achieving the best performance for inference. Predictions are generated at the span level and can subsequently be aggregated to obtain response-level hallucination detection results.

4 Experiments

4.1 Correlation Analysis

The primary objective of this work is to leverage mechanistic signals, i.e., the **External Context Score** (ECS) and the **Parametric Knowledge Score** (PKS), for hallucination detection, under the assumption that both are correlated with hallucination occurrence in generated responses.

We begin by examining the relationship between ECS and RAG hallucinations. Specifically, we compare ECS values between truthful and hallucinated responses. Figure 2(a) reports per-layer, per-head scores, all of which are positive. Since ECS reflects an LLM's reliance on retrieved context through attention heads, these results indicate that hallucinated responses utilize less external context than truthful ones. To further test this hypothesis, we computed the Pearson Correlation Coefficient (PCC) between hallucination labels and ECS. Because a negative correlation was expected, we use the inverse hallucination label instead. As shown in Figure 2(b), all attention heads exhibit negative correlations, confirming that higher ECS values are associated with lower hallucination likelihood. Taken together, Figures 2(a) and 2(b) suggest that RAG hallucinations emerge when the model fails to adequately exploit external context. We also examine the role of copying heads by computing `OV_copying_score` as a proxy of `full_OV_copying_score` (Figure 2(c)). This proxy is based on the findings in [26], which shows that `OV_copying_score` and `full_OV_copying_score` exhibit a strong positive correlation across heads and layers, suggesting that `OV_copying_score` can provide interpretable insights at lower computational cost. However, unlike the strong correlation observed between ECS and copying head scores in LLaMA models [8], we did not find such correlation in Qwen3-0.6b. Consequently, we use attention scores from all layers and heads for ECS calculation and rely on feature-selection techniques during the classification stage.

We next investigate how PKS contributes to RAG hallucinations. Figure 3(a) shows that PKS values are positive across nearly all layers, except the final one. Notably, FFN modules in later layers exhibit substantially higher scores for hallucinated responses than for truthful ones, yielding elevated layer-averaged scores for hallucinations. Consistently, Figure 3(b) presents the Pearson correlation between hallucination labels and PKS, revealing that later-layer FFNs are positively correlated with hallucinations.

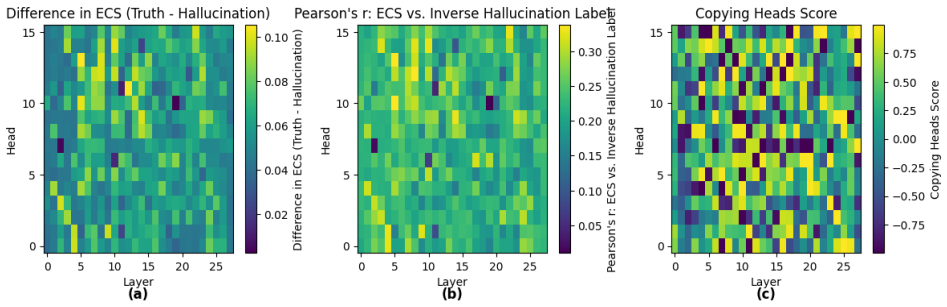

Figure 2: Relationship Between LLM Utilization of External Context and Hallucination

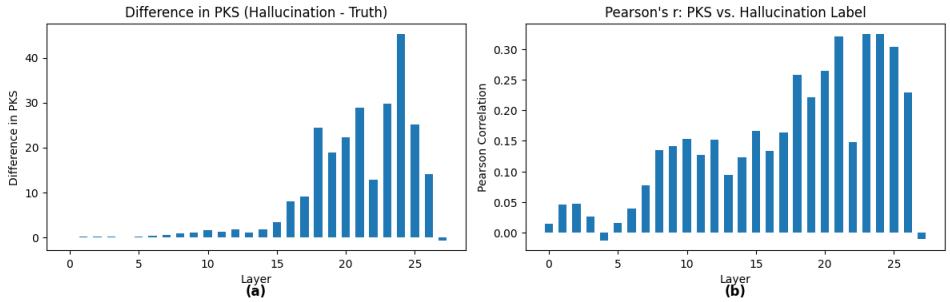

Figure 3: Relationship Between LLM Utilization of Parametric Knowledge and Hallucination

4.2 Hallucination Detection

Training of Classifiers: Training was conducted at the span level. Following the data preparation procedure in Section 3.1, we obtained 1,852 instances, corresponding to 7,799 span-level samples (4,406 negative and 3,393 positive). The input features consist of the **External Context Score** (ECS) and **Parametric Knowledge Score** (PKS) defined in Section 3.2. For Qwen3-0.6b, which has 28 layers and 16 attention heads per layer, this yields 476 features in total. To mitigate redundancy, feature selection reduced the dimensionality from 476 to 341. Results are reported in Table 1. Among the four models, SVC achieved the highest validation F1 score and was selected as the final prediction model. By contrast, XGBoost achieved strong training performance but exhibited severe overfitting, highlighting the risk of high-capacity models with limited data and the need for stronger regularization or substantially more training samples to generalize effectively. In comparison, SVC offered a more stable balance between model complexity and generalization. Further training details are provided in Appendix C.

Prediction: We obtain response-level labels using the trained SVC model by aggregating span-level predictions: a response is labeled as hallucinated if any span is predicted as such. We evaluate two settings: **self-evaluation**, where responses are generated by the same model used to compute ECS/PKS (Qwen3-0.6b), and **proxy-based evaluation**, where responses are produced by a different, typically production-level model (e.g., GPT-4.1 mini). We assume that, for responses grounded in retrieved context, the model should rely more heavily on external context than on its parametric knowledge. This assumption is model-agnostic. Consequently, given a pair consisting of a response and its corresponding retrieved context, we can leverage internal signals from any model to assess the relative utilization of external context versus parametric knowledge. This assumption forms the foundation of our **proxy-based** approach.

Our method is compared against a diverse set of baselines, including proprietary models (GPT-5, GPT-4.1), open-source models (GPT-OSS-20b, LLaMA-3.3-70b-versatile, LLaMA-3.1-8b-instant, Qwen3-32b, Qwen3-0.6b), and commercial detection systems (RAGAS [27], TruLens [28], RefChecker [29]). For language models, we apply the prompt in Appendix D to assess the faithfulness of responses relative to retrieved documents. For commercial detection systems, implementation details are provided in Appendix E. Detection results are summarized in Table 2, and key observations are discussed below:

- Overall, our method achieves moderate performance. In the self-evaluation setting, we obtain higher F1 scores than TruLens and llama-3.1-8b-instant, and perform comparably to RefChecker. In the proxy-based evaluation, our method performs even better, outperforming nearly all models except GPT-5 and RAGAS in F1 score. Notably, the models with superior performance are either proprietary or significantly larger, whereas we employ an efficient classifier leveraging signals from a 0.6b parameter model.
- We also include Qwen3-0.6b as a detection model to demonstrate that, by itself, the 0.6b model is insufficient for hallucination detection. However, when combined with a strong classifier that leverages its internal signals, performance is substantially improved.
- Our model exhibits higher recall than precision in both settings. One possible explanation is that the training dataset may contain some false-positive samples, which could bias the classifier toward

predicting more hallucinations. During data curation (see Section 3.1), we retained instances where LettuceDetect won the majority vote but did not exclude low-confidence span-level labels. These low-confidence spans are potential false positives but warrant further investigation. Nonetheless, we argue that higher recall is generally desirable, as false-positive cases can be further verified in downstream tasks.

- In proxy-based evaluation, we observe that all language models (from tiny 0.6b to GPT-5) exhibit higher precision than recall. This trend may stem from models such as GPT-4.1-mini producing fluent, reasonable-sounding responses, where subtle inaccuracies are less likely to be flagged by the detection models. In contrast, during self-evaluation, errors or unsupported content generated by Qwen3-0.6b are more readily detected by stronger models.

Table 1: Span-level Detection performance (%)

Classifier	Train Prec.	Val Prec.	Train Rec.	Val Rec.	Train F1	Val F1
LR	79.71	74.84	77.05	71.09	78.36	72.92
SVC	84.44	79.00	79.24	74.34	81.76	76.60
RandomForest	79.87	74.92	76.13	72.27	77.95	73.57
XGBoost	99.74	76.45	99.77	73.75	99.75	75.08

Table 2: Response-level Detection Performance (%)

	Self-Evaluation			Proxy-based Evaluation		
Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
GPT-5	77.27	**92.97**	**84.40**	91.67	66.27	**76.92**
GPT-4.1	76.39	85.94	80.88	**94.29**	39.76	55.93
GPT-OSS-20b	82.79	78.91	80.80	92.31	43.37	59.02
llama-3.3-70b-versatile	81.03	73.44	77.05	93.75	18.07	30.30
llama-3.1-8b-instant	69.23	49.22	57.53	70.37	22.89	34.55
Qwen3-32b	79.55	82.03	80.77	86.11	37.35	52.10
Qwen3-0.6b	70.27	20.31	31.52	78.79	31.33	44.83
RAGAS	68.45	89.84	77.70	75.29	77.11	76.19
TruLens	**89.61**	53.91	67.32	49.08	**96.39**	65.04
RefChecker	84.62	68.75	75.86	71.43	12.05	20.62
Ours	63.89	89.84	74.68	62.90	93.98	75.36

Note: RAGAS, TruLens and RefChecker use GPT-4.1 under the hood. In all the experiments, we obtain mechanistic metrics from Qwen3-0.6b. Model response is from Qwen3-0.6b under Self-Evaluation while from GPT-4.1 mini under Proxy-based Evaluation.

5 Conclusion

In this work, we developed a detection method for RAG hallucinations by decoupling the attributions from parametric knowledge and external context. Our correlation study shows that hallucinations arise from insufficient utilization of external context and over-reliance on parametric knowledge. Guided by these insights, we experimented with a number of classification methods to predict span-level hallucination and aggregate the results for response-level detection. We demonstrate the comparable ability of this cost-free, low-memory model with the commercial counterparts. Moreover, we show that our model can be used as a proxy on evaluation of large-scale, production-level models.

6 Limitations

While our approach demonstrates the effectiveness of mechanistic signals for hallucination detection in RAG, several limitations remain. First, our ECS and PKS computations rely on information from all layers and heads, which is computationally intensive, particularly during the projection from hidden states to the vocabulary distribution. Given the weak and unclear correlation between

copying-head scores in Qwen3-0.6b and hallucination, future work should aim to identify the layers most critical for efficient hallucination detection. For instance, our correlation analysis suggests that retaining only late-layer signals may suffice for downstream classification, reducing computation while preserving predictive power. Second, we only utilize ECS and PKS as input features for classification. Although these mechanistic features provide valuable signals, the overall performance remains below that of large-scale models. Expanding the feature set to include uncertainty-based measures and representation-level features could enrich the representation and improve detection accuracy. Finally, a more ambitious application of mechanistic signals is in response intervention, where the goal is to produce more reliable and truthful outputs by steering the model during generation. In such a setting, the current `proxy-based evaluation` framework is insufficient, as effective intervention requires access to and manipulation of layer-level activations in real time. Implementing such interventions would necessitate substantial computational resources and more sophisticated tooling to interface with the model's internal states.

Acknowledgments and Disclosure of Funding

We thank Neo for supporting this research through their startup accelerator program. Their contribution played a crucial role in enabling the development and evaluation of our models.

References

[1] Siwei Wang et al. Infibench: Evaluating the question-answering capabilities of code large language models. *OpenReview*, 2024.

[2] Balázs Szalontai et al. Large language models for code summarization. *arXiv preprint arXiv:2405.19032*, 2024.

[3] Juyong Jiang et al. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[4] Zhiwei Ji, Yiming Zhang, Yicheng Liu, et al. A survey on hallucination in large language models. *arXiv preprint arXiv:2311.05232*, 2023.

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

[6] Juntong Song, Xingguang Wang, Juno Zhu, Yuanhao Wu, Xuxin Cheng, Randy Zhong, and Cheng Niu. Rag-hat: A hallucination-aware tuning pipeline for llm in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1548–1558, 2024.

[7] Haichuan Hu, Congqing He, Xiaochen Xie, and Quanjun Zhang. Lrp4rag: Detecting hallucinations in retrieval-augmented generation via layer-wise relevance propagation. *arXiv preprint arXiv:2408.15533*, 2024.

[8] Zhongxiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, and Han Li. Redeep: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. *arXiv preprint arXiv:2410.11414*, 2024.

[9] Likun Tan, Kuan-Wei Huang, and Kevin Wu. Fred: Financial retrieval-enhanced detection and editing of hallucinations in language models. *arXiv preprint arXiv:2507.20930*, 2025.

[10] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of ACL 2024*, 2024.

[11] Ádám Kovács and Gábor Recski. Lettucedetect: A hallucination detection framework for rag applications. *arXiv preprint arXiv:2501.00232*, 2025.

[12] Raúl Vázquez, Timothée Mickus, Elaine Zosa, Teemu Vahtola, Jörg Tiedemann, et al. Semeval-2025 task 3: Mu-shroom, the multilingual shared task on hallucinations and related observable overgeneration mistakes. *arXiv preprint arXiv:2504.11975*, 2025.

[13] Marcely Boito, Damien Haziza, Khalid Choukri, et al. A search-enhanced rag pipeline for hallucination detection. In *Proceedings of SemEval 2025*, 2025.

[14] Zhongxiang Sun, Xiaoxue Zang, Kai Zheng, et al. Can we mitigate rag hallucinations without regeneration or modifying llms? (aarf: Add attention, reduce ffn). In *Proceedings of The Web Conference 2025*, 2025. Slides: ICLR 2025 workshop presentation.

[15] Yuchen Liu, Yiming Zhang, Hao Chen, et al. Lrp4rag: Detecting hallucinations in retrieval-augmented generation via layer-wise relevance propagation. *arXiv preprint arXiv:2408.15533*, 2024.

[16] Shuo Wang, Xin Li, Wayne Xin Zhao, et al. Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey. *arXiv preprint arXiv:2504.14891*, 2025.

[17] Wayne Xin Zhao, Kun Liu, Junfeng Huang, et al. Retrieval-augmented generation: A comprehensive survey of techniques and evaluation. *arXiv preprint arXiv:2506.00054*, 2025.

[18] Neel Nanda and Joseph Bloom. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`, 2022.

[19] Ekaterina Fadeeva, Luca Calatroni, Giorgio Chiaselotti, Stefano Morosi, Tiziana Spadea, and Simone Vantini. Franq: Faithfulness-aware uncertainty quantification for fact-checking the output of retrieval augmented generation. *arXiv preprint arXiv:2503.12345*, 2025.

[20] Juntong Song, Yuxiang Wang, Lei Zhang, Tianyu Zhao, Shizhe He, Dongyan Zhang, and Kai Xu. Rag-hat: A hallucination-aware tuning pipeline for llm in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[21] Reshmi Ghosh, Rahul Seetharaman, Hitesh Wadhwa, Somyaa Aggarwal, Samyadeep Basu, Soundararajan Srinivasan, Wenlong Zhao, Shreyas Chaudhari, and Ehsan Aghazadeh. Quantifying reliance on external information over parametric knowledge during retrieval augmented generation (rag) using mechanistic analysis. *arXiv preprint arXiv:2410.00857*, 2024.

[22] Zeping Yu and Sophia Ananiadou. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[23] Sitao Cheng, Liangming Pan, Xunjian Yin, Xinyi Wang, and William Yang Wang. Understanding the interplay between parametric and contextual knowledge for large language models. *arXiv preprint arXiv:2410.08414*, 2024.

[24] Andrew Parry, Catherine Chen, Carsten Eickhoff, and Sean MacAvaney. Mechir: A mechanistic interpretability framework for information retrieval. In *Proceedings of the 47th European Conference on Information Retrieval*, 2025.

[25] Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

[26] Anonymous (TransformerLens Documentation). Transformerlens main demo notebook – eigenvalue copying score analysis. `https://transformerlensorg.github.io/TransformerLens/generated/demos/Main_Demo.html`, 2025. Shows empirical finding that OV_copying_score and full_OV_copying_score are "highly (but not perfectly!) correlated".

[27] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, 2024.

[28] Anupam Datta, Matt Fredrikson, Klas Leino, Kaiji Lu, Shayak Sen, Ricardo Shih, and Zifan Wang. Exploring conceptual soundness with trulens. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176, pages 302–307. PMLR, 2022.

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

A Preprocessing Example

A sample entry from our preprocessed dataset is shown below:

```
question: "What is the growth rate in CCG operating income in 2015?"

documents: [
  "Management 2019's discussion and analysis of financial condition
     and results of operations (continued). "
  "The following results drove changes in CCG operating income by
     approximately the amounts indicated:"
]

prompt: "Given the context, please answer the question based on the
    provided information from the context. Include any reasoning with
    the answer.\n\nContext: Management 2019's discussion and analysis
    of financial condition and results of operations (continued). The
    following results drove changes in CCG operating income by
    approximately the amounts indicated:\nQuestion: What is the growth
     rate in CCG operating income in 2015?\nAnswer:"

response: "The exact growth rate is derived from the differences in
    the values provided. Thus, the answer is that the growth rate in
    CCG operating income in 2015 is approximately -20.7%."

labels: [
  {start: 673, end: 722, confidence: 0.8037, text: "Growth Rate} = \\
     frac{-2161}{10327} \\times 100\\%"},
  {start: 730, end: 734, confidence: 0.5833, text: "20.7"}
]

labels_llama: 1
labels_gpt: 1

prompt_spans: [[0,135],[136,144],[145,348],[349,1177],[1178,3465]]
response_spans: [[0,383],[384,597],[598,663]]
```

Notes:

- `labels` provide span-level hallucination annotations with start and end indices in the response text.
- `labels_llama` and `labels_gpt` indicate whether the respective LLM judges marked the response as hallucinated (1) or not (0).
- `prompt_spans` and `response_spans` segment the prompt and response into sentence or phrase-level chunks for span-level scoring.

B Computation Details of Mechanistic Metrics

The computation was performed using the TransformerLens library on the Qwen3-0.6B model. Inference was executed on a Google Colab L4 GPU with 24-GB memory, using `torch.float16` precision to reduce activation storage costs. Sentence-level semantic similarity was computed using the BAAI/bge-base-en-v1.5 encoder, also hosted on GPU to avoid transfer overhead.

The average execution time for an end-to-end ECS/PKS computation was 42 seconds per example. GPU allocation remained within 1.9–2.1 GB, with reserved memory peaking at 2.2 GB across iterations. After each iteration, tensors were explicitly released, and memory was reclaimed using `torch.cuda.empty_cache()` and `torch.cuda.ipc_collect()` to prevent fragmentation.

C Training Details of Classifiers

We split the dataset of 1,852 instances (7,799 span-level samples in total, with 4,406 negative and 3,393 positive labels) into training and validation subsets using a 90/10 stratified split based on the hallucination label. As features, we combined external context scores (per attention head and layer) with parametric knowledge scores (per layer). Feature preprocessing was performed through a pipeline consisting of standardization (`StandardScaler`), removal of near-constant features (`DropConstantFeatures`), elimination of duplicate features (`DropDuplicateFeatures`), and correlation-based selection (`SmartCorrelatedSelection`) with a Pearson correlation threshold of 0.9. The correlation filter employed a `RandomForestClassifier` with maximum depth 5 as the estimator. After preprocessing, the feature dimensionality was reduced from the full set to a filtered subset. We then evaluated four classifiers—Logistic Regression, Support Vector Classification (SVC), Random Forest, and XGBoost—using pipelines composed of the preprocessing module followed by the respective classifier. For Random Forest and XGBoost, we set the maximum tree depth to 5 while leaving other hyperparameters at their default values.

D Prompt for Baselines

Below is the prompt used for hallucination detection when using models GPT-5, GPT-4.1, GPT-OSS-20b, LLaMA-3.3-70b-versatile, LLaMA-3.1-8b-instant, Qwen3-32b and Qwen3-0.6b.

User Prompt

```
You are an expert fact-checker. Given a context, a question, and a response,
your task is to determine if the response is faithful to the context.
Context: context
Question: question
Response: response
Is the response supported and grounded in the context above? Answer "Yes"
or "No", and provide a short reason if the answer is "No". Be concise and
objective.
```

E Implementation of Commercial Detection Systems

We describe our implementation of three commercial tools—RAGAS, TruLens, and RefChecker—for hallucination detection below.

RAGAS: We use RAGAS's `faithfulness` metric to evaluate how well a model's responses align with the provided context documents. GPT-4.1 is used as the evaluator model. For each data point, a faithfulness score between 0 and 1 is computed. To determine the optimal threshold that maximizes F1 score, we evaluate F1 across candidate thresholds in [0,1]. Predictions are binarized at each threshold, and the threshold that maximizes F1—balancing precision and recall—is selected.

TruLens: TruLens provides a framework for evaluating hallucination via its groundedness feedback mechanism. We use the `groundedness_measure_with_cot_reasons` function to compute groundedness scores in [0,1]. The F1-optimal threshold is determined using the same procedure as in RAGAS.

RefChecker: RefChecker extracts factual claims from model responses and verifies them against reference documents. It consists of two components: an `LLMExtractor` that extracts claims, and an `LLMChecker` that classifies claims as `Entailment`, `Neutral`, or `Contradictory`. A response is labeled hallucinated if any claim is `Contradictory`, and non-hallucinated if all claims are either `Entailment` or `Neutral`.