An Automatic Method to Estimate Correctness of RAG

Chi Zhang [†1], Vivek Datla[†‡2], Aditya Shrivastava[2], Alfy Samuel[2],

Zhiqi Huang[2], Anoop Kumar[2], Daben Liu[2]

[1]Carnegie Mellon University

[2]Capital One

chiz5[†]@andrew.cmu.edu

{vivek.datla[†‡], aditya.shrivastava2, alfy.samuel, zhiqi.huang, anoop.kumar, daben.liu}@capitalone.com

[†]Equal Contribution, [‡]Corresponding Author

Abstract

In sectors in where data quality is critical, like finance and healthcare, it is crucial to have confidence in not only the outputs generated by retrieval-augmented generation (RAG) models but also the process followed by the model while arriving at the output. Existing methods, such as hallucination detection and input-output entailment measurements, fail to capture the model's internal state during answer generation. This paper introduces a novel approach to predict the correctness of the generated answer by modeling the model's uncertainty on quantified perturbations of input. Extensive experiments across multiple large language models (LLMs) demonstrate that our approach quantifies RAG robustness by aligning predictions with ground truth with a Avg.Mean Square Error (MSE) ≤ 0.002 while offering flexibility for diverse qualitative metrics.

1 Introduction

The advent of LLMs has improved many natural language processing (NLP) tasks, with one of the most significant applications being Retrieval-Augmented Generation (RAG). By combining information retrieval with powerful generative capabilities, RAG enables models to generate more accurate, contextually relevant responses to complex queries(Lewis et al., 2020).

The use of RAG systems in domains such as finance, healthcare, and legal applications presents significant risks. In these high-stake domains, providing a wrong or inappropriate answer can lead to severe consequences, particularly when the response has legal or ethical implications (Benjamin and Schweber, 2023). A key challenge in deploying RAG systems in such contexts is the potential for the underlying LLM to generate answers from its pre-trained knowledge rather than relying on the specific context provided. This behavior is especially dangerous as the current guardrails fail, i.e., current evaluation methodologies primarily assess the correctness of the answer without scrutinizing the internal state of LLM when generating the answer (Cao et al., 2022).

This problem becomes particularly pronounced with popular large scale LLMs trained with huge amount of training data, such as Llama and Mistral/Mixtral families. The sheer volume of training data enables these models to recall information from memory, creating the illusion of correctness without context alignment. As these models are increasingly used in real-world applications, where the inputs are often unseen or unfamiliar, the risk of incorrect or contextually irrelevant answers grows. This highlights the critical need to distinguish between genuine context-driven responses and those generated from the model's pre-existing knowledge.

An illustrative example of this issue is the *third variable problem*, such as the statistical correlation between ice-cream sales and drowning accidents. While the correlation may be true on a population level, the underlying cause is not ice-cream consumption but rather the fact that people are more likely to swim during the summer months when ice-cream consumption is also higher. Similarly, LLMs trained on extensive corpora like RedPajama (Computer, 2023) often perform well on test sets derived from these corpora (Xu et al., 2024). However, in real-world applications with novel inputs, the models are more likely to generate answers from memory rather than context, increasing the risk of unreliable outputs (Xu et al., 2023).

1.1 Looking deeper into RAG

At its core, RAG involves two key processes: retrieving pertinent information in response to a query and generating answers based on that retrieved content. The retrieval component ensures that the model has access to documents or passages most relevant to the input query, while the genera-

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 603–611
January 19–24, 2025.

Symbol	Meaning
BM	Base Model
PM	Perturbation Model
MSE	Mean Square Error
PTB	Perturbation
SS	Single Shot
IT	Iterative
RD_P	Random perturbation
EN_P	Entropy based perturbation
LL	Log-likelihood
B_c	Baseline Condition
N_c	Normal Condition
P_c	Perturbed Condition
t	Threshold
i	Instruction
c	Context
pt_c	Perturbed Context
q	Question
p	Prompt $(i + c + q)$
pt_p	Perturbed Prompt $(i + pt_c + q)$

Table 1: Notations and Abbrevations

tion component synthesizes this information into coherent, informative responses.

With the RAG framework, it is usually unclear whether the output is generated based on the given context or from the LLM's preexisting memory. Specifically, we are interested in understanding the impact on the generated output when the model becomes confused. We are interested in answering the following research questions:

Can we predict :

- When LLMs generate wrong answers?
- When LLMs exhibit pre-learned behavior, ignore the input and instead generate a response from memory?
- When LLMs generate sub-par output?

Understanding confusion within an LLM during text generation provides a valuable insight into the LLM's internal state when predicting the next token. It is important to note that multiple models within the same family, such as Llama3-70B and Llama3-8b, often use the same tokenizer but differ in terms of size, with variations in the number of parameters. Interestingly, the generation of the next token may pose different levels of difficulty, or *"confusion"*, across models of the same family.

This paper presents a method for predicting the correctness of output generated by LLM by analyzing the model's confusion in response to quantified perturbations. We capture the LLM's patterns of confusion when generating responses with the original context, a perturbed context, and with no

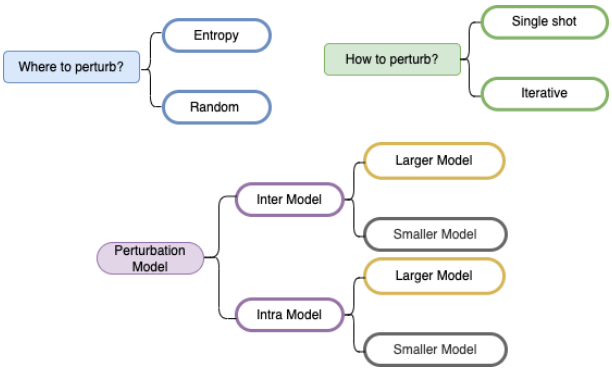

Figure 1: Various conditions considered while studying the impact on BM's confusion

context. These patterns are then used to train a regression model that predicts how closely the output aligns with the ground truth answer. We conducted several experiments with LLMs of varying sizes within the same family and different model families to assess how each LLM handles these perturbations. Figure 1 shows various conditions considered while studying the impact of LLM's confusion while generating an output. More details of these selections are in Section 3.

2 Related Work

Perturbation of text input to an LLM refers to the intentional alteration or modification of the input data provided to an LLM in order to evaluate its robustness, sensitivity, and generalization capabilities. These perturbations can be small but significant changes, such as:

- Word substitutions: Replacing specific words with synonyms, misspellings, or out-of-vocabulary terms (Wang et al., 2023).
- Sentence restructuring: Changing the order of words or sentences while maintaining the overall meaning (Hu et al., 2024).
- Noise injection: Introducing random errors, typos, or irrelevant data into the input (Le et al., 2022).
- Contextual modifications: Removing or altering certain contextual clues to see how the model reacts to incomplete or ambiguous information (Li et al., 2020).

Perturbations have been extensively used to study the robustness of models by creating synthetic adversarial examples. Contextualized adversarial generation model (CLARE) (Li et al., 2020) generates perturbations on the input text using various combinations of replacement, insertion, and deletion of the text. In their modeling, the goal was

to test the minimum or efficient edits to achieve a successful adversarial behavior of the LLMs.

In similar lines of work, the TextFooler (Jin et al., 2020) method demonstrates that by exploiting entailment features, even when assuming the target model is a black box, it is possible to make classifiers change their predictions. Specifically, the method identifies keywords in the target model and prioritizes replacing them with semantically similar and grammatically correct alternatives, causing models trained on BERT embeddings to alter their responses. The inspiration from this work is the evidence that pretrained LLMs are sensitive to underlying memory representations, and assumptions of consistency/robustness of models built on them might be risky.

Our perturbation algorithm follows in the same lines as CLARE (Li et al., 2020), their contextual perturbations using an LLM showed promising results for creating confusing contexts to the models. We improve on this idea by identifying "where" to perturb based on the model's inherent confusion. When using an LLM to generate text, we observe inflection points where the model exhibits low confidence and masks those tokens.

Inspired by DetectGPT's (Mitchell et al., 2023) approach to using perturbed texts to quantify changes in NLL (as a predictor of a model's behavior when deviating from its memory representation), we combine both techniques. Specifically, we observe NLL changes as the model generates text, then perturb the text strategically by determining "when," "where," and "how" to modify it. This allows us to evaluate the model's behavior during output generation.

The SAPLMA (Azaria and Mitchell, 2023) model helps in identifying the truthfulness of the generated answers by finding patterns in the hidden layers when the model is generating the answer. The authors propose that modeling variations of activation across hidden-layers as a solution to handle hallucinations.

3 Method

To estimate the correctness of output generated by LLM in a RAG setting, we track changes in the model's log-likelihood (LL) at critical inflection points. This method allows us to assess the model's confidence in generating responses under varying conditions. Specifically, we capture the negative log-likelihood (NLL) loss across three experimental settings. Fig. 2 shows these three settings:

- Baseline Condition (B_c): Model is given a prompt (p) which has $\{i, q\}$, without any context (c). This setup serves to evaluate the model's performance by relying solely on its pre-trained knowledge. The goal is to quantify how the model's uncertainty or confusion manifests when it generates answers without the guidance of relevant context.
- Normal Condition (N_c) : Model is given p which has $\{i, c, q\}$. This configuration reflects a standard RAG setting, where the model is expected to leverage both the provided context and the question to produce an accurate response. The aim is to measure how the model's confusion varies in a typical RAG scenario where context plays a crucial role.
- Perturbed Condition (P_c): In this setting, the model receives perturbed prompt(pt_p) consisting of $\{i, pt_c, q\}$ (details of the PTB process are outlined in the next section). The i and q remain consistent across all conditions, ensuring that any observed variations in the model's performance can be attributed to pt_c. This setting enables us to measure how the model's confusion fluctuates with a perturbed context (pt_c).

In each of these conditions, we control pt_p, maintaining uniformity in both the i and q. By comparing the NLL across these scenarios, we seek to identify key variations in the model's behavior (its internal confusion) and its ability to generate accurate responses under different contextual influences.

3.1 Perturbations(PTBs)

As illustrated in Fig. 3, the input to the LLM within the RAG framework is structured as a prompt (p) comprising three components: instruction (i), context(c), and, question (q). In this study, we focus exclusively on perturbing the context, applying controlled modifications to examine their impact. Specifically, we manipulate the context along three dimensions: what is perturbed, where the perturbation occurs, and how the perturbation is applied. The strategies employed for systematic perturbation are outlined in Fig. 1, detailing the methods used to ensure consistent and measurable alterations to the input context.

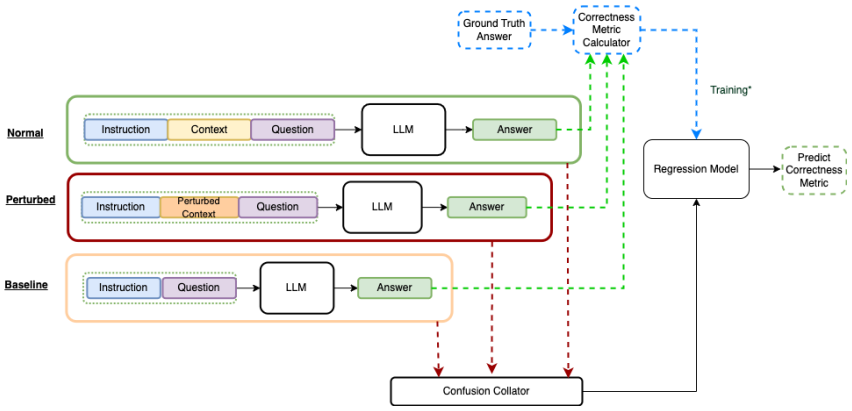

Figure 2: Identifying confusion across various settings of input to RAG to predict the output correctness.

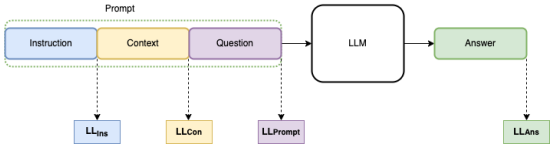

Figure 3: The various places where we capture the NLL across the input and generated output of the RAG system.

3.2 Where to perturb?

To determine the locations for PTBs, we experimented with two distinct approaches:

- Entropy-based perturbation(EN_P): Perturbing the context based on the most confusing token as identified by a perturbation model, which is another LLM.
- Random perturbation(RD_P): Perturbing the context at random.

The PM used in either case may belong to the same or a different model family as BM and can vary in size, either smaller or larger than the base model. PTBs can be applied iteratively (IT), degrading the context one token at a time or in a single-shot (SS) approach, where multiple PTBs are introduced simultaneously. This allows for a controlled examination of how different PTB strategies impact model behavior.

3.3 Entropy Based Perturbation (EN_P):

This process ensures a controlled modification of the input context based on the most confusing tokens if the text were to be generated by the perturbation model (PM). The PM is another LLM having a variety of possible combinations such as type, size, and others shown in Fig. 1. Below are the steps for EN_P.

1. For a given input text consisting of N tokens and based on a predefined hyper-parameter K (the number of perturbations), we first calculate the negative log-likelihood (NLL) of generating each token using a PM.
2. The token with the lowest log-likelihood is identified as the most confusing token.
3. This confusing token is replaced with a masked token specific to the PM, and PM is used to predict and fill in the replaced token.
4. In the iterative perturbation (IT-EN_P) setting, the process is repeated in K steps, with each step identifying and replacing the next most confusing token.
5. In the single-shot perturbation (SS-RN_P) setting, all K confusing tokens are identified, masked, and filled by PM in one step.

Fig. 4 shows an example of how we perform IT-EN_P and use it as context inside the RAG.

3.4 Predict Model *Correctness*

After having systematically perturbed input and generating the output for the three cases, N_c (no perturbation of context), B_c (no context provided), and P_c (with perturbed context), we collect the model's internal confusion on the input and output.

3.4.1 Collecting Internal Model Confusion

The following are the places for collecting LL from BM under all three conditions:

- N_c: prompt, instruction, context, question, and output
- B_c: prompt, instruction, question, and output
- P_c: prompt, instruction, perturbed context, question, and output

3.4.2 Estimating *Correctness* of Output

All the LL numbers calculated above are provided to a regression model as the input. We train a random forest regression model (Pedregosa et al., 2011) with 2100 estimators to predict the similarity

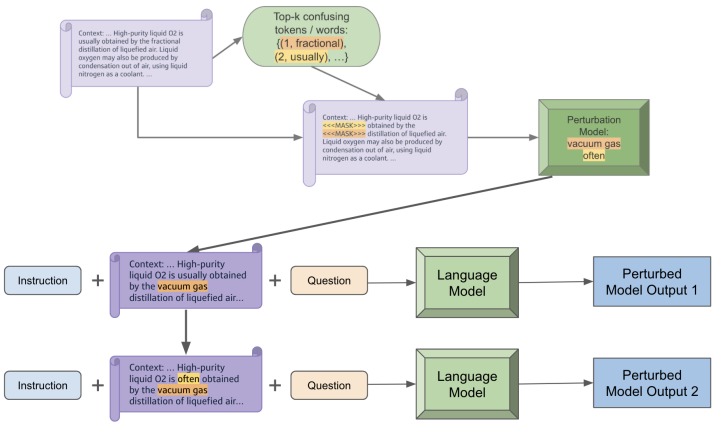

Figure 4: An example of IT-EN_P

score between the ground truth and the generated output.

During training time, we used a correctness metric (such as, cosine similarity) of the ground truth and generated an answer under the original condition as a signal for correctness and as the expected output of the regression model. In this way, the regression model learns to estimate the *Correctness* according to the log-likelihood results on model inputs and outputs generated in different cases.

This number is usually a continuous number between 0 and 1, representing the similarity between the model output and the ground truth result. During evaluation, we chose a threshold (t) for that correctness to decide whether the output is good or bad. The t is a hyper-parameter that needs to be adjusted based on BM.

4 Experiments

4.1 Dataset and Model

We evaluated our method to predict the correctness of the output generation of RAG on the SQuAD dataset (Rajpurkar et al., 2016). We randomly sampled 3000 data points total from the SQuAD dataset, among which 2100 were from the training dataset to train the regression model, and 900 were from the evaluation dataset to evaluate the result. Here we assumed that the retrieval portion of RAG is already performed and we have the right context to generate the answer.

We experiment with our method on a variety of large language models, including T5-3b (Raffel et al., 2019), Llama3-8b (Touvron et al., 2023), Mistral-7b (Jiang et al., 2023), and Llama3-70b (Touvron et al., 2023).

4.2 Metrics

During the evaluation, we first selected a Difference Metric to compute the similarity between the model output and the ground truth answer. This gives us a *Correctness* score, which we aim to estimate with our method.

The *Correctness* score ranges from [0,1] for Cosine Similarity and ROUGE (Lin, 2004). To evaluate estimation accuracy, we first report the Mean Square Error (MSE) between the estimated *Correctness* score and the predicted score. We then set a threshold(t) for the ground truth to produce a binary result: if the model's *Correctness* score exceeds t, we classify the output as correct (positive), and if it falls below, we classify it as incorrect (negative). By comparing our estimated *Correctness* score to the binary result, we calculate the AUROC.

Finally, we apply the same t to our estimated *Correctness* score to classify the model output as correct (positive) or incorrect (negative). We then compare this classification with the binary result from the ground truth to evaluate the Accuracy and F1 score of our method.

5 Results

As mentioned above we experimented with several BMs and PMs under various settings. To ascertain the effect of PTBs on predicting the correctness score, we used Llama3-8b as our BM and PM. Table 2 shows that at 15 PTBs, we are able to predict the correctness score with the highest Accuracy, F1, and AUROC.

For the experiments below on Llama3-8b we use 15 PTBs as our standard. Table 3 shows the effect when perturbating iteratively (PTB-IT) vs

#PTBs	Acc	F1	AUROC
5	0.710	**0.791**	0.735
10	0.697	0.737	**0.773**
15	**0.712**	0.753	**0.773**
20	0.709	0.748	0.765
25	0.679	0.769	0.698

Table 2: Effect of #PTBs on Llama 3-8b as BM and PM.($t = 0.95$ and $0.002 \leq MSE \leq 0.004$)

single-shot (SS) using the EN_P method. This is a very conservative approach where the BM and PM are the same. Results show that we have a qualitatively better chance of predicting the closeness to the correctness score when using SS-PTBs in the EN_P setting. Table 4 shows that iterative EN_P setting (see Table 3) has more predictability on the correctness of the output compared to RD_P setting.

Pert. type	Acc	F1	AUROC
Single-shot	0.740	0.782	0.797
Iterative	0.718	0.754	0.785

Table 3: SS VS IT PTB using EN_P on Llama3-8b model as BM and PM.(t=0.95 and MSE $\leq$ 0.004)

Pert. type	Acc.	F1	AUROC
Single-shot	0.731	0.779	0.789
Iterative	0.729	0.762	0.806

Table 4: SS VS IT PTB using RD_P on Llama3-8b model as BM and PM.(t=0.95 and MSE $\leq$ 0.003)

When using different PMs (see Figure 1), we experimented with inter and intra model-family PTBs. We used a smaller model google-t5-3b, a similar sized model mistral-7B and a large model llama3-70B as PMs. When experimenting with the IT-EN_P setting, we observe that (see Table 5) PTBs with a smaller model of a different model family help to infer the correctness of the output better than using the same model or a similar model from a different family.

Similar experiment when performed under the IT-RD_P setting using the same PMs on Llama3-8b showed a lesser predictability of the correctness score compared to the IT-EN_P setting (see Table 6). The result shows that EN_P using different PMs still affects BM significantly compared to RD_P, where the perturbed token might not be confusing to BM. Studying the predictability of the model correctness under SS-EN_P setting (see Table 7) shows that IT-EN_P is more effective than SS-EN_P as systematic perturbation captures the

PM	Acc	F1	AUROC
T5-3b	0.726	**0.801**	0.736
Mistral-7b	0.704	0.783	0.729
Llama3-70b	**0.752**	0.785	**0.824**

Table 5: Effect of PMs on Llama3-8b under IT-EN_P. (t=0.92 and MSE$\leq$ 0.004)

changes in input and always the most confusing token at that step of iteration is considered. Where as SS-PTBs replace all tokens at the same time. The simplest setting for PTBs is SS-RD_P where what tokens are replaced are chosen at random and they are perturbed in a single-shot. Table 8 shows that even in this simplistic setting, we can quantify the effectiveness of our approach.

PM	Acc	F1	AUROC
T5-3b	0.737	0.773	0.807
Mistral-7b	0.749	0.775	0.831
Llama3-70b	**0.754**	**0.800**	**0.832**

Table 6: Effect of PMs on Llama3-8b under IT-RD_P. (t=0.95 and MSE$\leq$ 0.002)

PM	Acc	F1	AUROC
T5-3b	0.715	0.751	0.779
Mistral-7b	0.678	0.725	0.748
Llama3-70b	**0.754**	**0.785**	**0.809**

Table 7: Effect of PMs on Llama3-8b under SS-EN_P. (t=0.95, MSE$\leq$ 0.004)

This approach of quantifying the model correctness based on the internal confusion of BM in the process of generating the output can be extended to multiple correctness scores. The final step of fitting the regression function to predict the correctness score needs to be performed. Table 9 shows the flexibility in our modeling technique to predict various correctness scores. The result shows that it is easier to predict cosine-similarity with bert-embeddings of output and ground-truth compared to ROUGE-like metrics, which are sensitive to the length of output.

We conducted several experiments using a variety of BMs and PMs. Table 10 shows experiments with T5-3b, Mistral-7b and Llama3-70b as BM, and T5-3b, Llama3-8b, Mistral-7b, and Llama3-70b as PM. The results show how the selection of PM plays an important role in predicting the correctness of the answer generated by BM. For T5-3B the better PM is itself. For Mistral-7B surprisingly, T5-3B is a better PM for predicting its correctness. Further investigation is needed to

PM	Acc	F1	AUROC
T5-3b	0.724	**0.759**	0.800
Mistral-7b	0.678	0.725	0.748
Llama3-70b	**0.731**	0.744	**0.823**

Table 8: Effect of PMs on Llama3-8b under SS-RD_P. (t=0.95, MSE$\leq$ 0.003)

Metric	MSE	Acc.	F1	AUROC
Cos Sim.	**0.004**	0.710	**0.791**	**0.735**
ROUGE-f1	0.016	**0.778**	0.715	0.719
ROUGE-p	0.008	0.621	0.574	0.640

Table 9: Predictability of different correctness scores using Llama3-8b as both PM and BM.

understand the impact of PM with different tokenizer family compared to BM. For Llama3-70B, Llama3-8B as PM shows better predictability of its correctness.

BM	PM	Acc.	F1	AUROC
T5-3b	**T5-3b**	**0.787**	**0.728**	**0.859**
	Llama3-8b	0.770	0.702	0.851
	Mistral-7b	0.767	0.7	0.856
	Llama3-70b	0.769	0.707	0.846
Mistral-7b	**T5-3b**	**0.658**	**0.737**	**0.656**
	Llama3-8b	0.631	0.717	0.634
	Mistral-7b	0.588	0.689	0.573
	Llama3-70b	0.592	0.691	0.584
Llama3-70b	T5-3b	0.700	0.783	0.715
	Llama3-8b	**0.756**	**0.823**	**0.784**
	Mistral-7b	0.590	0.469	0.706

Table 10: Effect of PMs on T5-3b (t=0.7, MSE$\leq$ 0.07), Mistral-7b (t=0.90, MSE$\leq$ 0.003) and Llama3-70b (t=0.95, MSE$\leq$ 0.004)

6 Discussion

Understanding the LLM's internal processes is essential to assess whether LLM relies on pre-training knowledge or follows the input prompt, which includes an instruction, context, and question. Even when the model adheres to the prompt, the variation in its internal confusion, especially in response to changes in the context, plays a significant role in the quality of the answer.

Another key finding from our experiments is that as models increase in parameter size, variation in context has less of an impact on model confusion. In many cases, larger models generate correct answers even when no context is provided despite instructions to use it. This suggests that large LLMs sometimes rely on pre-training knowledge rather than the given context, likely due to encountering similar data during training (such as Wikipedia), though not necessarily the specific dataset. So, having PTBs on the input context showed lesser impact on the internal confusion of the model. Even though counter intuitive, having a healthy amount of confusion while generating the output on variations of input (PTBs) is a positive signal that the model is using the context while generating the output.

For smaller models, our approach reliably predicts when the model produces sub-optimal responses. When perturbing the input using the same model, we observe consistent improvements in prediction accuracy, particularly as the number of perturbations increases from 5 to 25. This supports our hypothesis that quantifying LLM's internal confusion while handling extensive perturbations is a valuable signal to predict when the LLM is likely to generate a sub-optimal result.

7 Future Work

In this work, we investigated how the variation in a model's internal confusion, triggered by controlled input perturbations, can serve as a signal for assessing the accuracy of LLM-generated output. As suggested by SAPLMA (Azaria and Mitchell, 2023), leveraging hidden layer activations offers additional benefits in identifying inflection points where a model's output shifts. Future work will focus on studying the effect of different tokenizers and hidden layer activations on the model's confusion. We aim to further explore how perturbation techniques affect specific configurations and families of LLMs, deepening our understanding of their behavior and robustness.

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, Singapore. Association for Computational Linguistics.

Weiser Benjamin and Nate Schweber. 2023. The chatgpt lawyer explains himself. *The New York Times*.

Meng Cao, Yue Dong, and Jackie Cheung. 2022. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3340–3354, Dublin, Ireland. Association for Computational Linguistics.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Xinyu Hu, Mingqi Gao, Sen Hu, Yang Zhang, Yicheng Chen, Teng Xu, and Xiaojun Wan. 2024. Are llm-based evaluators confusing nlg quality criteria? *arXiv preprint arXiv:2402.12055*.

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

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2953–2965, Dublin, Ireland. Association for Computational Linguistics.

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.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. 2023. Are large language models really robust to word-level perturbations? *arXiv preprint arXiv:2309.11166*.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2023. An llm can fool itself: A prompt-based adversarial attack. *arXiv preprint arXiv:2310.13345*.

Appendix

A More results for different LLMs

A.1 Models accuracy

Pert. Model	Acc	F1 w/Buf	F1 no/Buf
T5-3b	**0.759**	**0.834**	**0.812**
Llama3-8b	0.756	0.831	0.798
Mistral-7b	0.755	0.831	0.771

Table 11: PTB of Llama3-8b and having a buffer where we do not predict the accuracy of the generated output. ($t = 0.92$)

A.2 More Correctness Estimation results

t	Acc	F1	AUROC
0.86	**0.919**	**0.958**	0.697
0.89	0.853	0.920	0.743
0.92	0.791	0.874	0.764
0.95	0.712	0.753	0.773
0.98	0.611	0.143	**0.781**

Table 12: Effect of t on Llama3-8b as both BM and PM under IT-EN_P setting. (MSE≤ 0.003)

Table 12 shows the effect of various t on the regression model predicting the correctness of the generated output. We selected the t that gave the F1 and AUROC scores.