Visualizing token importance for black-box language models

Paulius Rauba*
University of Cambridge

Qiyao Wei*
University of Cambridge

Mihaela van der Schaar
University of Cambridge

Abstract

We consider the problem of auditing *black-box* large language models (LLMs) to ensure they behave reliably when deployed in production settings, particularly in high-stakes domains such as legal, medical, and regulatory compliance. Existing approaches for LLM auditing often focus on isolated aspects of model behavior, such as detecting specific biases or evaluating fairness. We are interested in a more general question—can we understand how the outputs of black-box LLMs depend on *each input token*? There is a critical need to have such tools in real-world applications that rely on inaccessible API endpoints to language models. However, this is a highly non-trivial problem, as LLMs are stochastic functions (i.e. two outputs will be different by chance), while computing prompt-level gradients to approximate input sensitivity is infeasible. To address this, we propose Distribution-Based Sensitivity Analysis (DBSA), a lightweight model-agnostic procedure to evaluate the sensitivity of the output of a language model for each input token, without making any distributional assumptions about the LLM. DBSA is developed as a *practical tool* for practitioners, enabling quick, plug-and-play visual exploration of LLMs reliance on specific input tokens. Through illustrative examples, we demonstrate how DBSA can enable users to inspect LLM inputs and find sensitivities that may be overlooked by existing LLM interpretability methods.

1 Introduction

Auditing black-box language models is a fundamental requirement to ensuring they behave reliably when deployed in the real-world. Society implicitly expects LLMs to operate ethically, comply with regulations, and deliver technically sound responses without causing harm. This has resulted in the development of broad audit frameworks focused on governance and regulatory oversight (Mökander et al., 2023; Meskó and Topol, 2023; Raji et al., 2022). Despite these high-level frameworks, *practical algorithms* that practitioners could employ to audit LLMs are scarce.

Why is there a need for practical LLM audit algorithms? While high-level frameworks are necessary, there is an increasing demand for *practical* algorithms that provide detailed, interpretable insights into LLM behavior. The urgency for such tools arises from two factors: *mitigating real-world harm* and *complying with regulatory demands*. Without precise auditing tools, LLMs may propagate biases or deliver inconsistent outputs, leading to societal harm (Gehman et al., 2020; Nozza, Bianchi, Hovy, et al., 2021) or diverge from their intended purpose over time, causing unforseen negative impacts (Lauer, 2021; Rahwan, 2018; Dafoe, 2018). Regulatory frameworks like the EU AI Act are considering classifying LLMs as "high-risk AI systems" that would require mandatory third-party audits before deployment (Helberger, Diakopoulos, et al., 2023; Mökander et al., 2023). Existing tools do not meet these needs because they either focus narrowly on specific biases or require complex, benchmark-oriented setups that are not adaptable to diverse real-world scenarios.

What is missing from current LLM auditing methods? The auditing landscape for LLMs is dominated by methods that target isolated dimensions, such as bias detection (Borkan et al., 2019; Dixon et al., 2018; Park, Shin, and Fung, 2018), fairness evaluation (Garg et al., 2019), or adversarial robustness (Geisler et al., 2024). While these approaches are valuable for targeted analysis, they lack the flexibility and general applicability required by practitioners who need quick, interpretable feedback on LLM behavior across varied domains. What is missing is a practical, easy-to-deploy, statistically-grounded tool that provides actionable, visual insights into the behavior of any black-box LLM.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

Table 1: **Why sensitivity analysis is important for model auditing**. Two examples show sensitivity scores for each word when an LLM is asked for legal advice. Darker shades indicate higher importance. In Example 1, the model focuses on the alleged crime, while in Example 2, its sensitivity to the person's name indicates an undesired reliance when the name is irrelevant to the LLM response. Example 2 could prompt further investigation in practical scenarios.
Legend: 🟥 Most important ⬜ Least important

Example 1 (Desired behavior):
John Doe is accused of murder.

Example 2 (Undesired behavior):
John Doe is accused of murder.

How can sensitivity analysis serve as a practical and interpretable tool for auditing LLMs? We diverge from existing auditing mechanisms and define an entirely new task—developing a model-agnostic framework for LLM sensitivity analysis. Sensitivity analysis offers a practical and interpretable way to audit LLMs by showing how model outputs change with variations in input. In a nutshell, the primary question that sensitivity analysis for language models answers is the following:

> *"How do the answers produced by a language model depend on each input token?"*

Such an approach allows practitioners to visualize and interpret the influence of each token, providing a model-agnostic way of auditing LLM behavior. For example, Table 1 shows how sensitivity analysis can reveal unwanted model behavior when evaluating legal prompts, providing clear, actionable insights that standard benchmarks might overlook. *The ability to identify such context-dependent sensitivities allows practitioners to explore if model behavior is appropriate for the given scenario.*

However, implementing such a tool poses unique challenges. Repurposing existing gradient-based methods for black-box LLMs (Zeiler and Fergus, 2014) is impossible because we assume no access to LLM internals. Another alternative could be to manually change the input to a desired perturbation and inspect the changes. However, this is problematic because LLMs are stochastic generators, i.e. two outputs will vary by chance. Therefore, it is *insufficient* to look at two individual answers due to the stochastic nature of the LLMs; one must consider the *entire distribution of possible answers* rather than individual outputs.

Our solution. To provide a practical solution for these challenges, we introduce *Distribution-Based Sensitivity Analysis* (DBSA). DBSA is a lightweight, exploratory tool that finds the most important tokens in an input prompt. It is designed to be immediately usable by practitioners without requiring access to proprietary model internals or specialized knowledge. To achieve this, we approximate sensitivity analysis by considering each token's nearest neighbors in embedding space, performing Monte Carlo sampling for each neighbor and comparing Monte Carlo estimates on a reduced similarity space. With this formulation, we (i) capture the stochastic nature of LLM outputs more faithfully than point estimates; (ii) connect LLM outputs to distributional testing; (iii) quantify effect size; (iv) maintain model and input agnosticism, and (v) allow to directly visualize and highlight tokens based on their importance.

What practical value does DBSA offer? DBSA is designed for immediate application across various domains such as legal, medical, and customer service, where understanding token-level behavior is important. Existing methods, such as gradient-based sensitivity and bias evaluation techniques, are limited to specific use cases or require access to model internals. DBSA addresses *the entirely new task of generalized sensitivity analysis for black-box LLMs*, applicable across diverse domains. Because this is a new interpretability setup, it does not lend itself naturally to benchmark comparisons—a reason why we focus on illustrative examples and ablation studies in Sec. 5 - 6. We see DBSA as having direct applicability in many applications, such as visualizing all tokens in a prompt (Table 2), top n tokens and computing effect sizes (Table 3), and others.

Contributions. ① *Conceptually*, we introduce an entirely new task—auditing how the output distribution of black-box LLMs depend on specific input tokens (Sec. 2). ② *Technically*, we develop DBSA, a lightweight, easily deployable model-agnostic algorithm that uses finite-sample approximations and statistical testing to measure the impact of token-level changes (Sec. 3 - 4). ③ *Empirically*, we show how DBSA offers black-box interpretability (Sec. 5) and build trust in the method by performing relevant quantitative ablations (Sec. 6).

2 Unique challenges of obtaining token importance scores in black-box language models

We start by defining our problem (Sec. 2.1) and explaining why evaluating LLM responses requires looking at *distributions* of answers (Sec. 2.2). We then explore the unique challenges this poses (Sec. 2.3), proposing a procedure to address them (Sec. 3).

2.1 Problem formulation

Let $\mathcal{X}$ denote the input space, where each input $x \in \mathcal{X}$ is a sequence of words or tokens $x = (t_1, t_2, \ldots, t_n)$. Let $\mathcal{Y}$ denote the output space of the machine learning system. We define the system as a stochastic mapping $\mathcal{S} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes the space of probability distributions over $\mathcal{Y}$.

Objective. Our objective is to address the following research question: Given a black-box ML system $\mathcal{S}$, an input $x \in \mathcal{X}$, and an individual token(s) t_i at position i in x, how can we systematically measure and interpret the sensitivity of the output distribution $\mathcal{S}(x)$ with respect to small perturbations of t_i? We measure such *minimal* changes in token-level sensitivity as follows.

Definition 1 (Token-Level Sensitivity)**.** *The token-level sensitivity of the machine learning system $\mathcal{S}$ at position i in input $x \in \mathcal{X}$ is defined as the expected change in the output distribution $\mathcal{S}(x)$ resulting from a small perturbation δt_i to the word t_i, normalized by the magnitude of the perturbation:*

$$\textit{Sensitivity}(t_i) = \lim_{\|\delta t_i\| \to 0} \frac{\omega\left(\mathcal{S}(x),\ \mathcal{S}\left(x_{[t_i \leftarrow t_i + \delta t_i]}\right)\right)}{\|\delta t_i\|}$$

$x_{[t_i \leftarrow t_i + \delta t_i]}$ *denotes the input sequence where t_i has been perturbed by δt_i and ω is a suitable distance measure.*

2.2 Assessing sensitivity through comparison of output distributions

How can we evaluate token-level sensitivity? This requires (a) analyzing how the entire distribution of responses would change under the defined sensitivity metric; (b) understanding how to approximate such small perturbations in a discrete token space and a black-box setting; (c) evaluate how to take into account the inherent stochasticity of language model responses. We address these in turn.

First, we need a framework to evaluate how the *entire distribution of responses* would change under a particular change in the input. To do this, we take the approach of using Monte Carlo estimates of perturbed inputs as a measure of constructing such a distribution (Rauba, Wei, and Schaar, 2024). Suppose we swap an input token with its nearest neighbor in embedding space. A naive approach might involve comparing individual outputs from $\mathcal{S}$ for the original input x and its perturbed version $x_{[t_i \leftarrow t_i + \delta t_i]}$:

$$y = \mathcal{S}(x), \quad y' = \mathcal{S}\left(x_{[t_i \leftarrow t_i + \delta t_i]}\right)$$

However, since $\mathcal{S}$ produces random outputs even for the same input, such a comparison between single samples y and y' is insufficient. Any observed difference could merely be a consequence of the system's stochasticity rather than the effect of the perturbation δt_i.

To address this challenge, we employ a statistical approach that compares the entire output distributions of $\mathcal{S}$ for the original and perturbed inputs (Rauba, Wei, and Schaar, 2024). Specifically, we consider the distributions $\mathcal{S}(x)$ and $\mathcal{S}\left(x_{[t_i \leftarrow t_i + \delta t_i]}\right)$ over the output space $\mathcal{Y}$. Therefore, our objective is to determine whether the swapped nearest neighbor token δt_i leads to a significant change in the output distribution. This can be formalized as a hypothesis testing problem:

$$\begin{aligned} H_0 &: \mathcal{S}(x) = \mathcal{S}\left(x_{[t_i \leftarrow t_i + \delta t_i]}\right) \quad \text{(No effect)} \\ H_1 &: \mathcal{S}(x) \neq \mathcal{S}\left(x_{[t_i \leftarrow t_i + \delta t_i]}\right) \quad \text{(Has an effect)} \end{aligned}$$

The primary benefit of such a distributional formulation is that it captures the full stochastic behavior of $\mathcal{S}$ instead of just a single realization. This means we could perform statistical inference by directly comparing these distributions and understanding how much the outputs have shifted across the whole output space, even detecting subtle shifts that might not be apparent from individual samples. However, while theoretically grounded, the usage of such a hypothesis testing framework in the context of LLMs poses unique practical challenges that do not appear in regular settings.

2.3 Challenges with analyzing token-level output distributions

Building on the foundational work of Rauba, Wei, and Schaar (2024), we identify three key challenges in analyzing output distributions for token-level sensitivity, with particular emphasis on the discrete nature of nearest-neighbor perturbations.

▶ **Challenge 1: Computational intractability**. As established in prior work, the fundamental computational barrier lies in the exponential size of the output space $\mathcal{Y}$. For language models operating on vocabulary V with sequence length L, we have $|\mathcal{Y}| = |V|^L$ possible outputs. The probability of any output sequence $y = (y_1, y_2, \ldots, y_L) \in \mathcal{Y}$ under input x is given by:

$$\mathcal{S}(x)(y) = P(y \mid x) = \prod_{t=1}^{L} P(y_t \mid y_{<t}, x)$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ represents the token history. Comparing distributions $\mathcal{S}(x)$ and $\mathcal{S}(x_{[t_i \leftarrow t_i + \delta t_i]})$ requires evaluating these probabilities across the entire output space - a computation that grows exponentially with both vocabulary size and sequence length.

▶ **Challenge 2: Semantic interpretability**. Furthermore, statistical differences between output distributions may not reflect meaningful semantic changes. That is, while the outputs might be different token-wise, they might convey the same semantic meaning. Consider the outputs:

y_1: "*Targeted radiation therapy is suggested*", y_2: "*We suggest targeted radiation therapy*"

While $\mathcal{S}(x)(y_1) \neq \mathcal{S}(x)(y_2)$ in general, both convey identical medical recommendations. Therefore, we require a way to evaluate outputs such that we take into account whether the *semantic* meaning has changed.

▶ **Challenge 3: Discrete token space constraints**. Third, we require to perform swaps in discrete token space. The definition of token-level sensitivity in Definition 1 assumes the existence of infinitesimal perturbations δt_i. However, in a discrete token space without access to embeddings, such continuous perturbations are undefined. Instead, we must work with discrete jumps between tokens, where the smallest possible perturbation is a substitution with the nearest neighbor:

$$\min_{\delta t_i} \|\delta t_i\| = \min_{t_j \in \mathcal{N}(t_i)} \operatorname{dist}(t_i, t_j)$$

where $\mathcal{N}(t_i)$ denotes the set of nearest neighbors of token t_i.

In order to develop a method to visualize tokens, we must make it computationally tractable, evaluate the effect of sensitivity on a semantic level and take into account the discrete nature of the token space. The following sections introduce an approach that addresses this.

Takeaway. The discrete nature of nearest-neighbor token perturbations introduces fundamental constraints on sensitivity analysis that require extending existing distributional approaches to handle non-infinitesimal changes in the black-box LLM setting.

3 Token-level sensitivity with finite sample approximations

How can we address the challenges outlined in Section 2.3? We propose a light-weight finite sample approximation procedure that enables practical estimation of token-level sensitivity through hypothesis testing on output distributions.

3.1 Addressing outlined challenges

▶ **Addressing challenge 1: Computational intractability**. We approximate the intractable distributions $\mathcal{S}(x)$ and $\mathcal{S}(x_{[t_i \leftarrow t'_i]})$ using finite samples obtained via Monte Carlo sampling of the LLM. We generate n

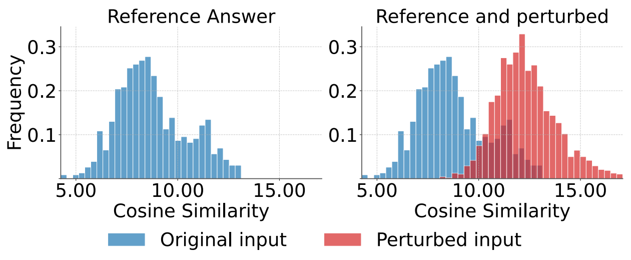


Figure 1: **Example of null and alternative distributions.** The null distribution P_0 (left, blue) is constructed based on the intrinsic variability of responses. The alternative distribution with a perturbed input of a *single* nearest neighbor P_1 (right, red) is quantified with respect to the original distributions. This figure shows the output distribution change in the cosine similarity space.

independent samples from each distribution, which is used to calculate the token-level sensitivities.

$$\hat{\mathcal{S}}(x) = \{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}, \quad y^{(j)} \sim \mathcal{S}(x),$$
$$\hat{\mathcal{S}}(x_{[t_i \leftarrow t'_i]}) = \{y'^{(1)}, y'^{(2)}, \dots, y'^{(n)}\}, \quad y'^{(j)} \sim \mathcal{S}(x_{[t_i \leftarrow t'_i]}).$$

▶ **Addressing challenge 2: Semantic interpretability**. Recall that statistical differences between output distributions may not reflect meaningful semantic changes. To ensure that we are indeed measuring semantic distances, we define a semantic similarity function $s : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ that quantifies the semantic similarity between outputs. We use a similarity metric in Sec. 4 to evaluate distributional differences.

▶. **Addressing challenge 3: Discrete token space constraints)**. We approximate the infinitesimal perturbation δt_i by substituting t_i with one of its k nearest neighbors in the embedding space. Let $\phi : \mathcal{W} \to \mathbb{R}^d$ be a token embedding function, and let $\mathcal{N}_k(t_i)$ denote the set of k nearest neighbors of t_i based on the similarity metric s. We select a small perturbation $t'_i \in \mathcal{N}_k(t_i)$ and define the perturbation magnitude as $\|\delta t_i\| = \|\phi(t'_i) - \phi(t_i)\|$.

3.2 Quantifying token-level sensitivity

Let P_0 be the distribution of similarities between pairs within $\hat{\mathcal{S}}(x)$, and P_1 be the distribution of similarities between pairs from $\hat{\mathcal{S}}(x)$ and $\hat{\mathcal{S}}(x_{[t_i \leftarrow t'_i]})$:

$$P_0 = \left\{ s\left(y^{(i)}, y^{(j)}\right) \mid 1 \leq i, j \leq n \right\},$$
$$P_1 = \left\{ s\left(y^{(i)}, y'^{(j)}\right) \mid 1 \leq i, j \leq n \right\},$$

where $y^{(i)}, y^{(j)} \overset{\text{i.i.d.}}{\sim} \mathcal{S}(x)$ and $y'^{(j)} \overset{\text{i.i.d.}}{\sim} \mathcal{S}(x_{[t_i \leftarrow t'_i]})$.Here, P_0 captures the intrinsic variability within the original output (equivalent to the *null distribution*), whereas P_1 captures the cross-distribution similarities between the original and swapped token outputs (equivalent to the *alternative distribution*), c.f. Figure 1, where each swapped token is a nearest neighbor of the original token. We have therefore constructed two distributions which represent the variability in answer similarities as a proxy for sensitivity (Def. 1). In Sec. 4, we show how such distributions can be used to obtain token-level sensitivities and associated p-values.

Our hypothesis test is then formulated as:

$$\begin{aligned} H_0 &: \mathbb{E}[P_0] = \mathbb{E}[P_1], && \text{(No significant effect)}, \\ H_1 &: \mathbb{E}[P_0] \neq \mathbb{E}[P_1], && \text{(Perturbation shifts similarity)}. \end{aligned}$$

We estimate the token-level sensitivity $\mathcal{S}$ at position i by computing the average difference between the null distribution and alternative distribution.

Advantages. With this formulation, we (i) capture the stochastic nature of LLM outputs more faithfully than point estimates; (ii) connect LLM outputs to frequentist hypothesis testing; (iii) quantify effect size; and (iv) maintain model and input agnosticism.

Takeaway. Using finite-sample approximations can address the three primary challenges associated with evaluating token-level sensitivity.

4 Distribution-based sensitivity analysis

Now, we present a light model-agnostic methodology for assessing the token-level sensitivity of LLMs. Our approach avoids restrictive distributional assumptions about LLM outputs. We enable frequentist statistical hypothesis testing using effect size and p-values through the construction of null and alternative distributions. Importantly, our framework is applicable to *any perturbation* and *any language model*, with the minimal requirement of being able to sample from the language model's output distribution and construct embeddings. We outline the general procedure below and provide other implementation details in Appendix B.

DBSA in a nutshell. DBSA computes token-level sensitivity by measuring the effect of token perturbations on LLM output distributions. Given an input sequence x and its unique tokens T, indices $\mathcal{I}_w$ for each token w are identified, and for each $i \in \mathcal{I}_w$, we find the k nearest neighbors $\mathcal{N}_k(t_i)$ of t_i in the embedding space using a distance function D. For each neighbor $t'_i \in \mathcal{N}_k(t_i)$, a perturbed sequence $x^{(t'_i)}$ is generated by replacing t_i with t'_i. The LLM produces samples $Y = \{y^{(j)}\}_{j=1}^n \sim \mathcal{S}(x)$ and $Y' = \{y'^{(j)}\}_{j=1}^m \sim \mathcal{S}(x^{(t'_i)})$, and their embeddings Φ and Φ' are used to compute pairwise similarities with a function s.

Distance measure. There are many possible distances metrics which might offer different properties. We seek for a distance which would (i) directly compute pairwise similarities; (ii) avoid intermediate density estimation; (iii) capture different moments of the distribution; (iv) would directly compute p-values, and (v) scale well with the number of samples. A natural choice for this is the *energy distance*. The energy distance between $\hat{\mathcal{S}}(x)$ and $\hat{\mathcal{S}}(x^{(t'_i)})$ is calculated as $E(\hat{\mathcal{S}}(x), \hat{\mathcal{S}}(x^{(t'_i)})) = 2A - B - C$, where $A = \frac{1}{nm}\sum_{a=1}^n \sum_{b=1}^m s(\phi(y^{(a)}), \phi(y'^{(b)}))$, $B = \frac{1}{n^2}\sum_{a=1}^n \sum_{b=1}^n s(\phi(y^{(a)}), \phi(y^{(b)}))$, and $C = \frac{1}{m^2}\sum_{a=1}^m \sum_{b=1}^m s(\phi(y'^{(a)}), \phi(y'^{(b)}))$. With this distance, sensitivity scores $\omega_{wi}^{(w'_i)}$ and p-values $p_{wi}^{(w'_i)}$ are computed for each token position and neighbor, then averaged across neighbors and instances to yield final scores ω_w and p_w for each token w. This model-agnostic approach only requires the ability to sample outputs and compute embeddings, ensuring broad applicability.

The chosen distance measure is a design choice and not a necessary component of the sensitivity analysis more broadly. One can choose a different ω, a different number of nearest neighbors or embedding components to suit each use case separately.

We provide more implementation details in Appendix B, including an algorithm for DBSA and a discussion on different design choices. We perform some major ablations of such choices in Sec. 6 and Appendix C.

Takeaway. A distribution-based approach provides a lightweight solution that captures the full variability of LLM outputs, enabling practitioners to audit model behavior without requiring access to internals.

5 Illustrative examples

Purpose and scope. In this section, we present two illustrative examples that demonstrate how DBSA can be used as a decision-support tool in real-world domains. Importantly, DBSA can be used *in addition to* other audit mechanisms to aid decision support. Our examples are based on text-based prompts because, unlike many common LLM interpretability methods, we do not assume any ground-truth labels. All experimental details can be found in Appendix D[1].

[1]Associated code for DBSA can be found here: https://github.com/vanderschaarlab/visualizing-token-

Comparison to other auditing algorithms. Existing sensitivity methods cannot be directly repurposed for these use cases due to the unique problem formulation (Sec. 2) that requires understanding *token-level* impact on LLMs.

Illustrative Example I: Legal Systems

Example I. Using DBSA in legal audit

The Problem. A tech company uses an AI-powered system to analyze legal contracts for potential risks by querying a language model for advice. Recently, the system missed a critical clause that resulted in massive legal damage.
The Need. The company cannot rely on black-box LLMs for contract evaluation. It needs more transparency and currently has no way of auditing models. They want to implement an additional monitoring system which can help understand *how* the answers might change if the information were slightly different.

Context. The consequences of misinterpreting key contractual elements can be severe. In this example, we illustrate how DBSA can be integrated into the auditing process to provide *token-level insights* that directly inform legal teams. Table 2 provides an example to visualize DBSA.

Table 2: **Example of using DBSA to audit legal documents for which words impact answers the most**. Each word is highlighted based on its normalized impact on the output distribution ω. Darker color highlights greater impact on answers.
Legend: 🟥 Most important ⬜ Least important

Company A agrees to pay Company B $10 million for developing a revolutionary AI software within 12 months . If Company B fails to deliver a fully functional product by the deadline , they must refund 50 % of the payment and provide an additional 3 months of development at no extra cost . However , if the delay is due to circumstances beyond Company B ' s reasonable control , these penalties shall not apply . This agreement is governed by California law and any disputes shall be resolved through binding arbitration .

Discussion I. DBSA ensures the LLM focuses on critical contractual terms like "agreement", "California", and "AI software" It also identifies irrelevant terms, such as "product" or "governed", confirming their minimal impact. This validates the model's behavior, giving legal teams confidence in its outputs. Therefore, DBSA can be used to highlight how LLM answers depend on each word to corroborate expert knowledge and evaluate LLM safety.

importance

Illustrative Example II: Clinical Support

Example II. Using DBSA in clinical support

The Problem. A hospital has implemented an LLM-powered clinical decision support system to assist doctors. The system analyzes patient data (e.g. medical history) to provide treatment recommendations. Recently, there was a near-miss incident where the system suggested an incorrect diagnosis that could have led to potentially harmful treatment. Upon review, it was discovered that *a single word* in the patient's symptom description significantly altered the system output.
The Need. The hospital cannot use an LLM without having an additional mechanism to understand the impact of each word on the output of the model.

Context. Decision-support systems must be interpretable and reliable, as their outputs directly impact patient safety. We can employ DBSA to identify the top k words which change the output the most with minor variability and compute their effect sizes (ω) as well as p-values. We show such an example in Table 3.

Table 3: **Example of using DBSA to identify top words affecting the output and identify effects of minor variations**. The top $k = 5$ words are highlighted, and their effect size, p-value, and statistical significance (p $< \alpha$, where $\alpha = 0.05$) are computed. We see that none of the words affect the output significantly due to the high p-value.
Legend: 🟥 Most important ⬜ Least important

Patient is a 45-year-old male presenting with progressive dyspnea on exertion over the past two weeks. On examination, patient appears mildly distressed. Lower extremities show 2+ pitting edema to mid -shin bilaterally. Chest X-ray shows pulmonary vascular congestion. Clinical presentation is consistent with new-onset congestive heart failure, likely due to hypertensive heart disease.

Word	Effect size	p-value	p $< \alpha$
congestive	0.08	0.11	✗
examination	0.07	0.16	✗
Lower	0.07	0.28	✗
mid	0.07	0.39	✗
hypertensive	0.06	0.31	✗

Discussion II. DBSA confirms the LLM's stability, as the highlighted words are appropriate and expected, demonstrating consistency in recommendations. In this way, DBSA can be used to highlight the top k words and evaluate their effect size.

Takeaway. Existing sensitivity methods are inadequate for highlighting the importance of each token. DBSA offers a model-agnostic solution to evaluate model sensitivity to each token.

6 Insights Experiments

Purpose and scope. To supplement the illustrative examples above, we aim to provide a preliminary analysis into the effectiveness and sensitivity of DBSA across various parameters and conditions. We do this by performing comprehensive empirical testing, ablating most of our design components and evaluating how they affect the results of DBSA. All experimental details can be found in Appendix .

6.1 Experiment I: Effects across models

Setup. We compare DBSA sensitivity analysis results across language models to assess consistency in identified sensitivities. Inputs are perturbed using nearest neighbors, and effect size distributions are evaluated across LLMs (see Figure 2) using the prompt from our clinical support example.

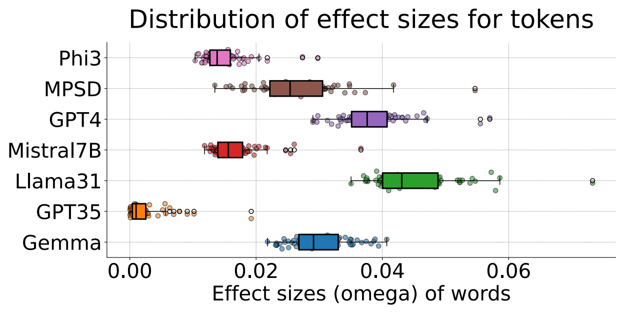


Figure 2: **Comparison of effect sizes across different language models**. The chart describes ω values for each word for seven large language models. The y-axis lists the models analyzed (full names in the Appendix, abbreviated for clarity). This comparison illustrates how the models vary in their sensitivity to input perturbations.

Takeaway I. DBSA shows variation in sensitivity patterns across models, indicating that model architecture influences token importance. This validates DBSA's capability to identify these variations.

6.2 Experiment II: Sensitivity across language models

Setup. We compute Spearman rank correlation coefficients between models to quantify similarity in sensitivity rankings, assessing if models exhibit consistent behavior in token sensitivity. Results are in Table 4.

Table 4: **Spearman Rank Coefficients between LLMs**. Higher values indicate stronger correlation in word rankings between models.

Model	GPT-4	GPT-3.5	SmolLM	MagicPrompt	Mistral-7B	Phi-3-mini	Llama-3-8B
GPT-4	1.00	0.14	0.03	-0.07	-0.02	0.04	-0.07
GPT-3.5	0.14	1.00	0.03	0.09	0.38	-0.07	0.17
SmolLM	0.03	0.03	1.00	0.16	0.25	0.05	0.23
MagicPrompt	-0.07	0.09	0.16	1.00	0.44	0.43	0.38
Mistral-7B	-0.02	0.38	0.25	0.44	1.00	0.16	0.25
Phi-3-mini	0.04	-0.07	0.05	0.43	0.16	1.00	0.11
Llama-3-8B	-0.07	0.17	0.23	0.38	0.25	0.11	1.00

Takeaway II. The analysis reveals substantial variability in sensitivity patterns across models, with some showing higher alignment (e.g., models from similar architectures like GPT-4 and GPT-3.5) while others demonstrate divergent sensitivity behaviors. DBSA effectively identifies these discrepancies, offering a reliable tool for practitioners to evaluate model robustness and make informed decisions when considering model replacements or optimizations.

6.3 Experiment III: Evaluating similarity functions

Setup. We test three similarity metrics (Cosine, L1, L2) to verify if DBSA's sensitivity analysis is consistent across metrics. We measure the Spearman correlation of token ω scores across LLMs to ensure comparable results (see Fig, 3).

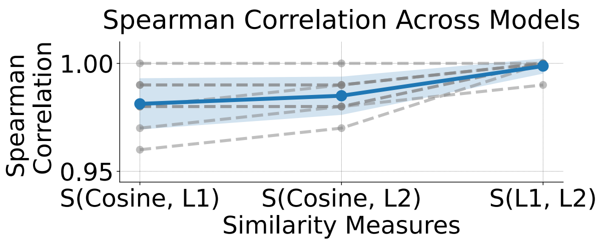


Figure 3: **Comparison of similarity functions (Cosine, L1, L2) in DBSA**. We show the average Spearman correlation across models (blue) and individual models (grey). Results indicate all three measures yield consistent outcomes, demonstrating their interchangeability within the DBSA framework.

Takeaway III. All three similarity metrics produce similar results, demonstrating DBSA's robustness. This confirms that practitioners can choose any standard metric without compromising the quality of analysis, offering flexibility in deploying DBSA.

6.4 Experiment IV: Evaluating the effect of Monte Carlo samples

Setup. We assess how varying Monte Carlo (MC) sample sizes affects DBSA's stability and reliability. We measure Spearman correlation between model outputs

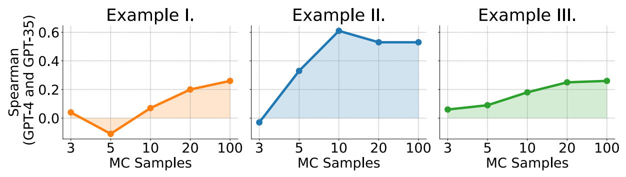


Figure 4: **Stabilization of Spearman correlation between GPT-3.5 and GPT-4 with increasing Monte Carlo (MC) samples for three prompts.** The figure shows that with fewer MC samples, the Spearman correlation fluctuates, indicating variability. As MC samples increase, the correlation stabilizes, showing greater reliability.

(e.g., GPT-4, GPT-3.5) to determine if larger sample sizes stabilize results across three prompts (Fig. 4).

Takeaway IV. As MC samples increase, DBSA's outputs stabilize across larger models, confirming the importance of sample size for accuracy. This shows that DBSA's reliability can be tuned based on sample quantity, offering guidelines for practitioners. This experiment establishes that DBSA provides consistent

6.5 Other empirical insights

We perform evaluation on other components, such as performing ablations by evaluating different distance functions (Energy, Earth Mover's Distance, or mean distance of cosine similarities) and the effect of choosing a different number of nearest neighbors (Appendix C).

7 Related work

Existing methods for evaluating the sensitivity and behavior of large language models (LLMs) primarily include gradient-based approaches, bias measurement, counterfactual fairness, and text summarization metrics. **Gradient-based methods** (Geisler et al., 2024) require access to model internals, making them impractical for black-box models, and focus on performance rather than interpretability. **Bias easurement techniques** (Borkan et al., 2019; Dixon et al., 2018; Park, Shin, and Fung, 2018) quantify fairness but are limited to specific subgroups and require human annotation, which restricts their flexibility and applicability to arbitrary perturbations. **Counterfactual fairness** approaches (Garg et al., 2019) commonly rely on specific assumptions, labeled data, and fundamentally answer a different question, i..e. that of fairaness. **Text summarization metrics** such as ROUGE and BERTScore (Bhandari et al., 2020; Zhang et al., 2019; Zhao et al., 2019; Lin, 2004) assess text generation quality but are not designed for inspecting output variability for input perturbations.

DBSA is a tangential auditing framework to the above-mentioned frameworks which answers a *fundamentally different question*. Because this paper defines an entirely new task—that of developing a model-agnostic framework for token-level sensitivity — we encourage practitioners to use DBSA *in addition to* any existing techniques that are used to audit LLM models. This is useful, as DBSA supports statistical inference, computes effect sizes, and allows exploration of any input perturbation, offering a human-interpretable approach that prioritizes qualitative understanding over benchmark performance. This is also important, as there have been direct calls for practical mechanisms to audit language models in addition to existing methods Mökander et al., 2023; Meskó and Topol, 2023. Table 5 summarizes these methods across five criteria. Extended related work can be found in Appendix A.

8 Discussion

The need for practical algorithms to audit LLMs. There is a growing need for *practical* algorithms that can be used to audit LLMs. DBSA is a versatile, model-agnostic framework designed for practitioners who wish to understand how sensitive the model is to information in the prompt. DBSA addresses a new task within the LLM auditing literature that other methods have so far overlooked. We see DBSA functioning as a diagnostic instrument that *complements* existing frameworks, especially in high-stakes settings. DBSA has immediate practical relevance to many practitioners who lack practical tools to audit LLMs. With this, we hope our work lays the foundation for a new class of statistics-based auditing tools.

Limitations. DBSA requires access to an embedding function and the ability to sample from the LLM multiple times which could limit its accessibility to some practitioners. We outline token-level sensitivity challenges in Sec. 2. While our solution in Sec. 3 addresses these, it has limitations, such as finding reliable nearest neighbors (which may vary in magnitude in embedding space) or selecting appropriate distance metrics/embedding functions. Finally, we wish to acknowledge the importance of ensuring that DBSA is deployed responsibly and does not inadvertently enable malicious actors to exploit vulnerabilities in LLMs.

Directions for Future Work (1) DBSA could be extended to have a multi-level approach to interpretability. We think this is a non-trivial (but extremely fruitful) extension of our current work. Such an extension would have to address at least five different non-trivial challenges: (i) which levels are worth testing (multiple tokens? semantic meaning? sentence-level?); (ii) the mutual information between levels and their interactions (we cannot assume independence; otherwise, this exten-

Table 5: **Comparison to related work**. DBSA addresses the entirely new task of generalized sensitivity analysis for black-box LLMs for which there are no ground-truth labels. This differs from other sensitivity- or interpretability-based methods. **Abbreviations: (I)**: Usable on any black-box model; **(II)**: Enables statistical inference; **(III)**: Computes the effect size; **(IV)**: Assumption-free

Method	Example Works	(I)	(II)	(III)	(IV)	Representative Question
Gradient Methods	(Geisler et al., 2024)	✗	✓	✗	✓	How does model output change under infinitesimal perturbation?
Measuring Unintended Bias	(Borkan et al., 2019; Dixon et al., 2018; Park, Shin, and Fung, 2018)	✓	✓	✓	✗	Does this model have unintended biases in certain subgroups?
Counterfactual Fairness	(Garg et al., 2019)	✗	✗	✓	✗	How would the prediction change if the sensitive attribute were different?
Text Summarization	(Bhandari et al., 2020; Zhang et al., 2019; Zhao et al., 2019; Lin, 2004)	✗	✗	✓	✓	How well is this text summarized?
Sensitivity analysis for LLMs	Our work (DBSA)	✓	✓	✓	✓	Do the responses change if we change any input in the prompt? If so, how?

sion loses its meaning); (iii) dealing with higher variance estimators (due to the non-zero covariances between levels); (iv) dealing with semantic drift (i.e. change in the meaning of the input across multiple places); and (v) repurposing the highlighting methodology (e.g. our current work offering highlights of tokens is extremely intuitive — how can this be achieved otherwise?). (2) Variance decomposition. Token probabilities could facilitate a formal decomposition of variability in output distributions, akin to an ANOVA-style partitioning of variance. Specifically, this could disentangle within-prompt variability (changes due to perturbations within a given input) from between-prompt variability (differences arising from distinct inputs). Such an approach could quantify the relative contributions of perturbations at various levels—be it tokens, phrases, or broader input features—to the overall stochastic behavior of the model. (3) Bayesian sensitivity. Token probabilities could be used within a Bayesian framework, where they serve as priors to model the effects of input perturbations. By updating these priors with observed outputs, DBSA could estimate posterior distributions, providing a probabilistic characterization of how input modifications alter the likelihood of specific outputs.

References

Astorga, Nicolás et al. (2024). "Active learning with llms for partially observed and cost-aware scenarios". In: *Advances in Neural Information Processing Systems* 37, pp. 20819–20857.

Belrose, Nora et al. (2023). "Eliciting latent predictions from transformers with the tuned lens". In: *arXiv preprint arXiv:2303.08112*.

Bhandari, Manik et al. (2020). "Re-evaluating evaluation in text summarization". In: *arXiv preprint arXiv:2010.07100*.

Borkan, Daniel et al. (2019). "Nuanced metrics for measuring unintended bias with real data for text classification". In: *Companion proceedings of the 2019 world wide web conference*, pp. 491–500.

Clark, Kevin (2019). "What Does Bert Look At? An Analysis of Bert's Attention". In: *arXiv preprint arXiv:1906.04341*.

Cowgill, Bo and Catherine Tucker (2017). "Algorithmic bias: A counterfactual perspective". In: *NSF Trustworthy Algorithms* 3.

Dafoe, Allan (2018). "AI governance: a research agenda". In: *Governance of AI Program, Future of Humanity Institute, University of Oxford: Oxford, UK* 1442, p. 1443.

Dixon, Lucas et al. (2018). "Measuring and mitigating unintended bias in text classification". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 67–73.

Enguehard, Joseph (2023). "Sequential Integrated Gradients: a simple but effective method for explaining language models". In: *arXiv preprint arXiv:2305.15853*.

Gallegos, Isabel O et al. (2024). "Bias and fairness in large language models: A survey". In: *Computational Linguistics*, pp. 1–79.

Gao, Bo (2015). "Exploratory visualization design towards online social network privacy and data literacy". In.

Garg, Sahaj et al. (2019). “Counterfactual fairness in text classification through robustness”. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 219–226.

Gehman, Samuel et al. (2020). “Realtoxicityprompts: Evaluating neural toxic degeneration in language models”. In: *arXiv preprint arXiv:2009.11462*.

Geisler, Simon et al. (2024). “Attacking large language models with projected gradient descent”. In: *arXiv preprint arXiv:2402.09154*.

Ghandeharioun, Asma et al. (2024). “Patchscope: A unifying framework for inspecting hidden representations of language models”. In: *arXiv preprint arXiv:2401.06102*.

Hadi, Muhammad Usman et al. (2023). “A survey on large language models: Applications, challenges, limitations, and practical usage”. In: *Authorea Preprints* 3.

Helberger, Natali, Nicholas Diakopoulos, et al. (2023). “ChatGPT and the AI Act”. In: *Internet Policy Review* 12.1, pp. 1–6.

Jung, Jongbin et al. (2018). “Algorithmic decision making in the presence of unmeasured confounding”. In: *arXiv preprint arXiv:1805.01868*.

Lauer, Dave (2021). “You cannot have AI ethics without ethics”. In: *AI and Ethics* 1.1, pp. 21–25.

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

Liu, Tennison et al. (2024). “Large language models to enhance bayesian optimization”. In: *arXiv preprint arXiv:2402.03921*.

Lundberg, SM and SI Lee (n.d.). *A unified approach to interpreting model predictions. NIPS’17: Proceedings of the 31st International Conference on Neural Information Processing Systems. December 2017 [Cited 2021 Jul 20]*.

Meskó, Bertalan and Eric J Topol (2023). “The imperative for regulatory oversight of large language models (or generative AI) in healthcare”. In: *NPJ digital medicine* 6.1, p. 120.

Mohri, Christopher and Tatsunori Hashimoto (2024). “Language models with conformal factuality guarantees”. In: *arXiv preprint arXiv:2402.10978*.

Mökander, Jakob et al. (2023). “Auditing large language models: a three-layered approach”. In: *AI and Ethics*, pp. 1–31.

Montavon, Grégoire et al. (2017). “Explaining nonlinear classification decisions with deep taylor decomposition”. In: *Pattern recognition* 65, pp. 211–222.

Morris, John X et al. (2023). “Text embeddings reveal (almost) as much as text”. In: *arXiv preprint arXiv:2310.06816*.

Nozza, Debora, Federico Bianchi, Dirk Hovy, et al. (2021). “HONEST: Measuring hurtful sentence completion in language models”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Park, Ji Ho, Jamin Shin, and Pascale Fung (2018). “Reducing gender bias in abusive language detection”. In: *arXiv preprint arXiv:1808.07231*.

Rahwan, Iyad (2018). “Society-in-the-loop: programming the algorithmic social contract”. In: *Ethics and information technology* 20.1, pp. 5–14.

Raji, Inioluwa Deborah et al. (2022). “Outsider oversight: Designing a third party audit ecosystem for ai governance”. In: *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 557–571.

Rauba, Paulius, Nabeel Seedat, Krzysztof Kacprzyk, et al. (2024). “Self-healing machine learning: A framework for autonomous adaptation in real-world environments”. In: *Advances in Neural Information Processing Systems* 37, pp. 42225–42267.

Rauba, Paulius, Nabeel Seedat, Max Ruiz Luyten, et al. (2024). “Context-aware testing: A new paradigm for model testing with large language models”. In: *Advances in Neural Information Processing Systems* 37, pp. 112505–112553.

Rauba, Paulius, Qiyao Wei, and Mihaela van der Schaar (2024). “Quantifying perturbation impacts for large language models”. In: *arXiv preprint arXiv:2412.00868*.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Sikdar, Sandipan, Parantapa Bhattacharya, and Kieran Heese (2021). “Integrated directional gradients: Feature interaction attribution for neural NLP models”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 865–878.

Singh, Chandan et al. (2024). “Rethinking interpretability in the era of large language models”. In: *arXiv preprint arXiv:2402.01761*.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (2017). “Axiomatic attribution for deep networks”. In: *International conference on machine learning*. PMLR, pp. 3319–3328.

Wang, Haotao et al. (2023). “How robust is your fairness? evaluating and sustaining fairness under unseen distribution shifts”. In: *Transactions on machine learning research* 2023.

Webster, Kellie et al. (2020). “Measuring and reducing gendered correlations in pre-trained models”. In: *arXiv preprint arXiv:2010.06032*.

Wen, Yuxin et al. (2024). “Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery”. In: *Advances in Neural Information Processing Systems* 36.

Yuksekgonul, Mert et al. (2024). “TextGrad: Automatic" Differentiation" via Text”. In: *arXiv preprint arXiv:2406.07496*.

Zeiler, Matthew D and Rob Fergus (2014). “Visualizing and understanding convolutional networks”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer, pp. 818–833.

Zhang, Tianyi et al. (2019). “Bertscore: Evaluating text generation with bert”. In: *arXiv preprint arXiv:1904.09675*.

Zhao, Wei et al. (2019). “MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance”. In: *arXiv preprint arXiv:1909.02622*.

Zou, Andy et al. (2023). “Representation engineering: A top-down approach to ai transparency”. In: *arXiv preprint arXiv:2310.01405*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] We include a clear description of the mathematical setting in Section 2, and appropriate algorithmic details.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] We analyze challenges including computation complexity in Section 2.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No] All source code will be made available upon publication.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] We include a clear statement of assumptions in Section 2.
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Yes] We include a clear explanation in Section 2

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] We include the data used in this experiment, and we discuss the model design choices that are made. All source code and reproduction instructions will be made available upon publication.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] All experiment details are described in Section 5.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] We clearly describe the functions and metrics we are using.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] We describe further experiment settings, including the compute used, in the appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] We appropriately document the models that we are using.
   (b) The license information of the assets, if applicable. [Yes] We appropriately document the license information.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes] We have further specification in the supplementary materials.
   (d) Information about consent from data providers/curators. [Not Applicable] We do not collect data from external sources.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable] There are no sensible contents in this paper.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable] We do not use crowdsourcing or conducted research with human subjects.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable] We do not use crowdsourcing or conducted research with human subjects.

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable] We do not use crowdsourcing or conducted research with human subjects.

Supplementary Materials

Appendix Contents

A Extended related work

A.1 Feature Attribution

The most relevant statistical methods for interpretability fall under "Feature attribution", such as SHAPLEY values, conformal prediction, etc. However, these methods are currently limited to the closed-end text generation domain, requires standardized text input/output formats and generation formats, and/or ground truth answers. Intuitively, these methods come from the statistical machine learning domain, and therefore are much better suited to e.g. tabular prediction tasks. Their application on LLMs is an adaptation rather than a new method overall. To the best of our knowledge, there is not a well-founded theory for SHAPLEY values in LLMs, especially the open-end text generation domain. On the other hand, there is one paper on the theory of conformal prediction in LLMs, but not related to perturbations (Mohri and Hashimoto, 2024).

More specifically, feature attribution methods deserve its name because a score is assigned to each input feature, reflecting its impact on the generated model output. Early attribution methods that have been developed include (1) perturbation-based methods (Lundberg and Lee, n.d.), (2) gradient-based methods (Sundararajan, Taly, and Yan, 2017; Montavon et al., 2017), (3) linear approximations (Ribeiro, S. Singh, and Guestrin, 2016). Recently, these methods have been specifically adapted for LLMs (Sikdar, Bhattacharya, and Heese, 2021; Enguehard, 2023). See (C. Singh et al., 2024) for a more complete literature review.

The gradient-inspired methods both break down the sentence into atomic elements (in language that would be tokens from byte pair encodings). Then, the gradient for each token can be calculated with respect to the final answer. This method suffers the same drawbacks as SHAPLEY values, specifically requiring standardized input/output text formats and/or ground truth answers. These papers demonstrate their method on closed-end tasks, such as text classification, prediction, etc. These methods were developed in tabular prediction tasks or compute vision tasks, and we have not done a really good job in adapting these methods to LLMs. An interesting parallel can be made that DBSA is attempting to adapt gradient-inspired methods on any open-end text generation.

A.2 Fairness/Bias

Certain methods in bias measurement and fairness quantification are also similar in spirit to DBSA. However, these works are limited to addressing specific subgroups, and while optional, they sometimes require human annotation (e.g. for bias annotations), which restricts their flexibility and applicability to arbitrary perturbations.

For example, DisCo (Webster et al., 2020) is a seminal piece of work that proposes to look at model output distributions under perturbation in the prompt. To begin, each prompt template (e.g., “[X] is [MASK]”; “[X] likes to [MASK]”) has two slots. The first slot is manually filled with attributes associated with a social group (the original paper featured gendered names and nouns, but easily extended to other sensitive groups with well-defined word lists), and the second slot is filled by the model’s top three candidate predictions. The final score is the number of different predictions between social groups across all prompt templates. Bias is measured by the difference between normalized probability scores for two binary and opposing social group words. The main limitation of this approach is that it only addresses specific subgroups, and the output distribution is rather limited (closed-end generation). For a detailed discussion on this topic, see (Gallegos et al., 2024).

Similar ideas have come up under works that investigate fairness under distributional shift. For example, (Wang et al., 2023) shows that while earlier methods proposed to adapt the model to be fair on the current known data distribution, or requires unlabeled data from the target distribution (assuming the target distribution is known), or requires the existence of a joint causal graph to represent the data distribution for all domains, their work aims to generalize fairness learned on current distribution to unknown and unseen target distributions. The field of fairness/bias also features the use of sensitivity analysis to explore how various aspects of the feature vector will affect a given outcome. Due to the introduction of SHAPLEY values, there has been an increase in attention from fairness researchers. (Cowgill and Tucker, 2017) proposed to perturb feature vectors to measure the effect on model performance of specific interventions. (Gao, 2015) investigated visual mechanisms to better display fairness/bias issues with data to users. (Jung et al., 2018) used sensitivity analysis to evaluate sensitive variables and their relationships with classification outcomes, indicating that sensitivity analysis can help to better understand uncertainty with respect to fairness. However, these methods are subject to the same problems discussed before. Namely, they are not general enough in their input/output formats, and they sometimes require human annotation (e.g. for bias annotations).

On an *application level*, we believe DBSA can directly improve the auditing of language modeling based systems to evaluate whether they are fair and reliable. That is, many applications today use language model as a system-level comonent for solving tasks. Consider a few examples. Language models are used to generate hypotheses to test existing ML systems (Rauba, Seedat, Ruiz Luyten, et al., 2024), improve information acquisition (Astorga et al., 2024), improve model robustness (Rauba, Seedat, Kacprzyk, et al., 2024) or even improve model hyperparameters (Liu et al., 2024). Many such applications exist, too broad to be covered here (Hadi et al., 2023). In most such works, the limitation section contains words of caution against fairness and bias—how can language models be trusted in such cases when they are used as external systems? Our work proposes a direct way to perform such audits.

A.3 Direct Prompting

Direct prompting methods generally work by either prompting along the lines of "Can you explain your logic" to an LLM, or in the case of Chain-of-Thought (CoT), goes back to changing specific values in the prompt in tasks like mathematical question-answering. In general, these methods either requires human intervention and examination, or ground truth answers labels, e.g. for process supervision, in context learning, RAG, etc. There has been efforts to make prompting methods gradient-inspired, see (Yuksekgonul et al., 2024) for an example.

Early adversarial attacks on LLMs apply simple character or token operations to trigger the LLM to generate incorrect predictions. Since these attacks usually generate misspelled prompts, they are easy to block in real-world applications. More recently, jailbreaking prompts are intentionally designed to bypass the LLM built-in safeguard capabilities, eliciting the generation of harmful content. However, the discrete nature of text has significantly impeded learning more effective adversarial attacks against LLMs. Even for recent work that has developed gradient-based optimizers for efficient text modality attacks ((Wen et al., 2024) presented a gradient-based discrete optimizer that is suitable for attacking the text pipeline of CLIP), they still require access to a white-box language model, and is limited in their input/output formats.

Other interpretability methods. We end the discussion on related works by documenting other interpretability methods that are also interesting approaches, but not immediately extended to the DBSA research. One promising method for understanding LLM representations is by looking at their attention heads (Clark, 2019), embeddings (Morris et al., 2023), and representations (Zou et al., 2023). There are also methods that directly decode an output token to understand what is represented at different positions and layers (Belrose et al., 2023; Ghandeharioun et al., 2024). Finally, influence functions are interesting to study LLM behavior, but the idea of influence functions is limited to the impact on adding one example on the training set. Also, influence functions require the derivative, i.e. white-box access to LLMs. In general, methods in mechanistic interpretability (e.g. Attention head importance, circuit analysis, influence functions) are interesting approaches to address LLM interpretability, but none of them are easily extended to the DBSA framework, and we leave comparison to these methods for future work.

B Implementation details

We provide a comprehensive guide for implementing Distribution-Based Sensitivity Analysis (DBSA). This section outlines the step-by-step process with specific code snippets and technical considerations.

B.1 Prerequisites and Setup

Our implementation uses Python 3.8 with NumPy, SciPy, Scikit-learn, and sentence-transformers libraries. We assume access to an LLM through a function `llm(prompt, n)` that generates `n` responses given a prompt.

B.2 Text Preprocessing and Tokenization

We tokenize the input text using a regular expression that captures words, numbers, and punctuation:

```
def tokenize_and_prepare_for_scoring(text):
    tokens = re.findall(r'\$?\d+(?:,\d+)*(?:\.\d+)?|\w+|[^\w\s]', text)
    token_positions = OrderedDict()
    for index, token in enumerate(tokens):
        if token not in token_positions:
            token_positions[token] = {'positions': [], 'score': None, 'pval': None}
        token_positions[token]['positions'].append(index)
    return tokens, token_positions
```

B.3 Sampling and Perturbing Responses

Key Point: We sample multiple responses for both original and perturbed prompts to approximate their distributions.

```
def get_responses(prompt, n=40):
    return llm(prompt, n=n)

def perturb_sentence(tokens, perturb_index, neighbor):
    new_tokens = tokens.copy()
    new_tokens[perturb_index] = neighbor
    return new_tokens
```

There are many ways to generate perturbations. In our example, we generate perturbations by asking an external LLM to generate k synonyms. Other possibilities include relying on external libraries, e.g. word2vec.

Embeddings. We use the OpenAI sentence transformer model "text-embedding-ada-002" to generate embeddings.

B.4 Computing Energy Distance

Key Point: We use energy distance as our metric for comparing distributions.

```
def compute_energy_distance(X, Y, distance='cosine'):
    n, m = len(X), len(Y)
    dists_XY = cdist(X, Y, distance)
    dists_XX = cdist(X, X, distance)
    dists_YY = cdist(Y, Y, distance)

    term1 = (2.0 / (n * m)) * np.sum(dists_XY)
    term2 = (1.0 / n**2) * np.sum(dists_XX)
    term3 = (1.0 / m**2) * np.sum(dists_YY)

    return term1 - term2 - term3
```

B.5 Performing Statistical Tests

We use a permutation test to calculate p-values:

```
def permutation_test_energy(X, Y, num_permutations=500, distance='cosine'):
    combined = np.vstack((X, Y))
    n = len(X)
    E_values = []
    for _ in range(num_permutations):
        np.random.shuffle(combined)
        perm_X, perm_Y = combined[:n], combined[n:]
        E_perm = compute_energy_distance(perm_X, perm_Y, distance)
        E_values.append(E_perm)
    return np.array(E_values)

def compute_energy_distance_fn(baseline_embeddings, perturbed_embeddings):
    E_n = compute_energy_distance(baseline_embeddings, perturbed_embeddings)
    E_values = permutation_test_energy(baseline_embeddings, perturbed_embeddings)
    p_value = np.mean(E_values >= E_n)
    return E_n, p_value
```

B.6 Alternative Implementations

Alternatively, we could combine the distance calculation + statistical test into one step:

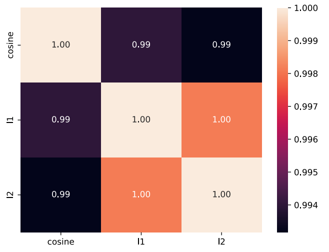
```
def calculate_difference_and_pvalue(arr1, arr2, num_permutations=1000, mode="energy"):
    if mode == "mean":
        observed_diff = np.abs(np.mean(arr1) - np.mean(arr2))
    elif mode == "EMD":
        observed_diff = wasserstein_distance(arr1, arr2)
    elif mode == "energy":
        observed_diff = energy_distance(arr1, arr2)
    else:
        raise ValueError(f"Invalid mode: {mode}")
    combined = np.concatenate([arr1, arr2])
    n1, n2 = len(arr1), len(arr2)
    permutation_diffs = []
    for _ in range(num_permutations):
        np.random.shuffle(combined)
        permuted_arr1, permuted_arr2 = combined[:n1], combined[n2:]
        if mode == "mean":
            permuted_diff = np.mean(permuted_arr1) - np.mean(permuted_arr2)
        elif mode == "EMD":
            permuted_diff = wasserstein_distance(permuted_arr1, permuted_arr2)
        elif mode == "energy":
            permuted_diff = energy_distance(permuted_arr1, permuted_arr2)
        else:
            raise ValueError(f"Invalid mode: {mode}")
        permutation_diffs.append(permuted_diff)
    p_value = np.mean(np.abs(permutation_diffs) >= np.abs(observed_diff))
    return observed_diff, p_value
```

Optimization. For improved efficiency, especially with longer texts, we implement parallel processing using Python's `multiprocessing` module to distribute token perturbation and analysis across available CPU cores.

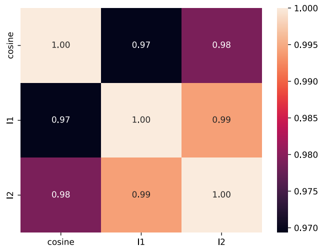
C Extended experiments

Purpose. This section provides additional insights into the similarity between choosing different distance functions for the energy-based calculations across different models. We run each model with different distances and evaluate the Spearman rank correlation between the given token outputs, where 1 indicates perfect ranking-based correlaction and 0 indicates no relationship.

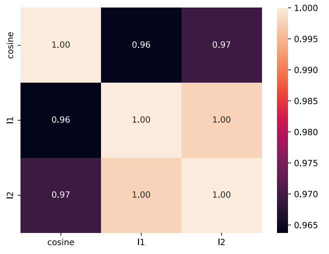
Discussion. Across all models, we find that the distance metrics consistently rank the same words as important for DBSA.



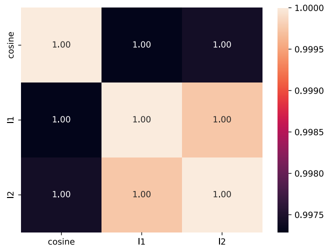
(a) GPT-3.5



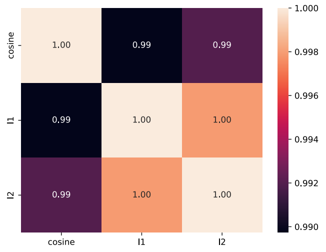
(b) GPT-4



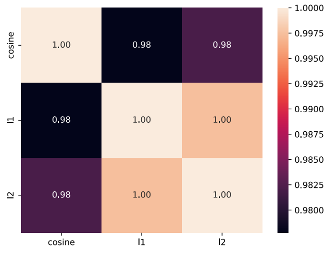
(c) SmolLM-135M



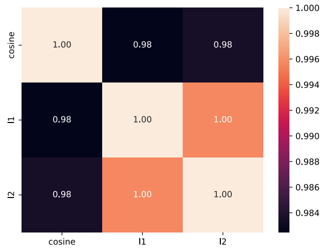
(d) MagicPrompt-Stable-Diffusion



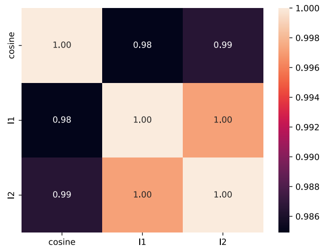
(e) Phi-3-mini-4k-instruct



(f) Mistral-7B-Instruct-v0.2



(g) Meta-Llama-3.1-8B-Instruct



(h) gemma-2-9b-it

Figure 5: Similarity functions for various models

D Experimental Details

D.1 Algorithm

The following is an implementation of DBSA that is used for evaluating the sensitivity of each word. We use the same notation as in the main part of the paper.

Algorithm 1: Distribution-Based Sensitivity Analysis

```
Require: Input sequence $x = (w_1, \dots, w_n)$; sample sizes $n, m$; nearest neighbors $k$; embedding function
    $\phi : \mathcal{Y} \to \mathbb{R}^d$; similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$; distance function $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
Ensure: Sensitivity scores $\{E_w, p_w\}_{w \in T}$
 1: $T \leftarrow \{w \mid w \in x\}$    // Unique tokens
 2: for all $w \in T$ do
 3:   $\mathcal{I}_w \leftarrow \{i \mid t_i = w\}$    // Token positions
 4:   for all $i \in \mathcal{I}_w$ do
 5:     $\mathcal{N}_k(t_i) \leftarrow \arg\min_{w' \in \mathcal{X}, w' \neq t_i} \{D(\phi(w'), \phi(t_i))\}_k$
 6:     for all $t'_i \in \mathcal{N}_k(t_i)$ do
 7:       $x^{(t'_i)} \leftarrow (w_1, \dots, w_{i-1}, t'_i, w_{i+1}, \dots, w_n)$
 8:       $Y \leftarrow \{y^{(j)}\}_{j=1}^n \sim \mathcal{S}(x)$, $Y' \leftarrow \{y'^{(j)}\}_{j=1}^m \sim \mathcal{S}(x^{(t'_i)})$
 9:       $\Phi \leftarrow \{\phi(y^{(j)})\}_{j=1}^n$, $\Phi' \leftarrow \{\phi(y'^{(j)})\}_{j=1}^m$
10:       $A_{wi}^{(w'_i)} \leftarrow \frac{2}{nm} \sum_{a=1}^n \sum_{b=1}^m s(\phi_a, \phi'_b)$
11:       $B_{wi}^{(w'_i)} \leftarrow \frac{1}{n^2} \sum_{a=1}^n \sum_{b=1}^n s(\phi_a, \phi_b)$
12:       $C_{wi}^{(w'_i)} \leftarrow \frac{1}{m^2} \sum_{a=1}^m \sum_{b=1}^m s(\phi'_a, \phi'_b)$
13:       $E_{wi}^{(w'_i)} \leftarrow A_{wi}^{(w'_i)} - B_{wi}^{(w'_i)} - C_{wi}^{(w'_i)}$
14:       $p_{wi}^{(w'_i)} \leftarrow \text{PermutationTest}(\Phi \cup \Phi', E_{wi}^{(w'_i)})$
15:     end for
16:     $\bar{E}_{wi} \leftarrow \frac{1}{k} \sum_{t'_i \in \mathcal{N}_k(t_i)} E_{wi}^{(w'_i)}, \bar{p}_{wi} \leftarrow \frac{1}{k} \sum_{t'_i \in \mathcal{N}_k(t_i)} p_{wi}^{(w'_i)}$
17:   end for
18:   $E_w \leftarrow \frac{1}{|\mathcal{I}_w|} \sum_{i \in \mathcal{I}_w} \bar{E}_{wi}, p_w \leftarrow \frac{1}{|\mathcal{I}_w|} \sum_{i \in \mathcal{I}_w} \bar{p}_{wi}$
19: end for
20: return $\{E_w, p_w\}_{w \in T}$
```

D.2 Language models used

Compute Resources. All the experiments in this paper were carried out on an A100 machine, with NVIDIA driver version 535.183.06 and CUDA12. The GPU VRAM is 80GB.

Language Models. A total of 8 language models were used for the experiments of this paper

- **GPT-3.5.** The GPT-3.5 model is a deployment taken from OpenAI endpoints. The deployment version is gpt-35-1106.
- **GPT-4.** The GPT-4 model is a deployment taken from OpenAI endpoints. The deployment version is gpt-4-0613-20231016.
- **SmolLM-135M.** The SmolLM-135M model is taken from Huggingface. The full model specification is HuggingFaceTB/SmolLM-135M.
- **MagicPrompt-Stable-Diffusion.** The MagicPrompt-Stable-Diffusion model is taken from Huggingface. The full model specification is Gustavosta/MagicPrompt-Stable-Diffusion.
- **Phi-3-mini-4k-instruct.** The Phi-3-mini-4k-instruct model is taken from Huggingface. The full model specification is microsoft/Phi-3-mini-4k-instruct.

- **Mistral-7B-Instruct-v0.2.** The Mistral-7B-Instruct-v0.2 model is taken from Huggingface. The full model specification is mistralai/Mistral-7B-Instruct-v0.2.
- **Meta-Llama-3.1-8B-Instruct.** The Meta-Llama-3.1-8B-Instruct model is taken from Huggingface. The full model specification is meta-llama/Meta-Llama-3.1-8B-Instruct.
- **gemma-2-9b-it.** The gemma-2-9b-it model is taken from Huggingface. The full model specification is google/gemma-2-9b-it.

D.3 Hyperparameter details

In our experiments with DBSA, we conducted ablations for most hyperparameters, as detailed in the main paper. However, it's challenging to be fully exhaustive. The following list outlines the key hyperparameters and considerations for each step of the DBSA process:

1. **LLM Sampling:** This step involves standard LLM sampling hyperparameters. We set the temperature to 1 and the max-length to 256. For more details on these and other sampling parameters, refer to the Huggingface documentation.
2. **Perturbation Generation:** This step requires selecting a perturbation method (e.g., synonyms, antonyms) and a function to generate these perturbations. In our experiments, we defined sensitivity as nearest-neighbor sensitivity to approximate gradients of the input prompt. We used GPT-4 to generate synonyms for each word, finding it empirically superior to alternatives like word2vec.
3. **Perturbed Prompt Sampling:** This step uses the same hyperparameters as Step 1.
4. **Embedding:** We used OpenAI's Ada embeddings, as they effectively capture meaningful semantic information in the prompts.
5. **Distance Calculation:** The selection of a distance function is key to measuring the disparity between response embeddings. Our experiments showed that cosine, L1, and L2 distances yielded similar results. We opted for cosine distance in our final experiments.
6. **Statistical Analysis:** The choice of statistical metric for calculating effect size between distributions during permutation testing is important. We selected the energy distance as our metric for effect size.

E Prompt examples

E.1 Prompts

The following prompts were the primary prompts used to evaluate DBSA sensitivity.

Listing 1: Prompt example 1

```
text_legal = """Company A agrees to pay Company B $10 million for developing a
    revolutionary AI software within 12 months. If Company B fails to deliver a fully
    functional product by the deadline, they must refund 50% of the payment and
    provide an additional 3 months of development at no extra cost. However, if the
    delay is due to circumstances beyond Company B's reasonable control, these
    penalties shall not apply. This agreement is governed by California law and any
    disputes shall be resolved through binding arbitration."""
```

Listing 2: Prompt example 2

```
text_medical = """Patient is a 45-year-old male presenting with progressive dyspnea on
     exertion over the past two weeks. On examination, patient appears mildly
    distressed. Lower extremities show 2+ pitting edema to mid-shin bilaterally. Chest
     X-ray shows pulmonary vascular congestion. Clinical presentation is consistent
    with new-onset congestive heart failure, likely due to hypertensive heart disease.
    """
```

Listing 3: Prompt example 3

```
text_trading = """A senior executive is accused of insider trading , allegedly using
    confidential information to gain substantial financial benefits . The defendant
    maintains his innocence , claiming that all investment decisions were based on
    public data ."""
```

Listing 4: Prompt example 4

```
text_manufacturing = A manufacturing company was sued for producing a faulty product
    that caused significant injuries to a customer ."""
```

E.2 Example nearest neighbors

This section outlines example nearest neighbors for different tokens.

Listing 5: Closest words for prompt 3

```
closest_words = {"Defendant": ["Accused", "Respondent", "Litigant"], ".": [",", "!", "
    ?"], ",": [";", ".", ":"], "Senior": ["Top-tier", "High-ranking", "Upper-level"],
    "Executive": ["Administrator", "Officer", "Manager"], "Accused": ["Alleged", "
    Charged", "Indicted"], "Insider": ["Internal", "In-house", "Privileged"], "Trading
    ": ["Dealing", "Stock-jobbing", "Market manipulation"], "Allegedly": ["Supposedly"
    , "Reportedly", "Purportedly"], "Using": ["Utilizing", "Employing", "Applying"], "
    Confidential": ["Private", "Secret", "Classified"], "Information": ["Data", "
    Details", "Intelligence"], "To": ["Toward", "For", "In order to", "So as to"], "
    Gain": ["Acquire", "Secure", "Obtain"], "Substantial": ["Significant", "
    Considerable", "Major"], "Financial": ["Monetary", "Fiscal", "Economic"], "
    Benefits": ["Advantages", "Gains", "Profits"], "Maintains": ["Affirms", "Asserts",
     "Insists"], "Innocence": ["Guiltlessness", "Blamelessness", "Purity"], "Claiming"
    : ["Asserting", "Stating", "Contending"], "Investment": ["Financial", "Capital", "
    Asset"], "Decisions": ["Choices", "Determinations", "Conclusions"], "Public": ["
    Open", "Publicly available", "Common"], "Data": ["Information", "Statistics", "
    Facts"], "A": ["An", "One", "Any"], "Is": ["Exists", "Stands", "Remains", "
    Constitutes"], "Of": ["Concerning", "Regarding", "About", "Pertaining to"], "Using
    ": ["Utilizing", "With the help of", "Via", "By means of"], "To": ["Toward", "For"
    , "In order to", "So as to"], "The": ["This", "That", "Said"], "His": ["Their", "
    Its", "Her"], "That": ["Which", "This", "What"], "All": ["Every", "Each", "Any"],
    "Were": ["Had been", "Were being", "Used to be"], "Based": ["Founded", "
    Established", "Built", "Grounded"], "On": ["Upon", "Over", "About", "Concerning"]}
```

Listing 6: Closest words for prompt 4

```
closest_words = {"Manufacturing": ["Production", "Industrial", "Fabrication"], "
    Company": ["Corporation", "Firm", "Business"], "Was": ["Had been", "Was being", "
    Were"], "Sued": ["Prosecuted", "Litigated against", "Indicted"], "For": ["Because
    of", "Due to", "On account of"], "Producing": ["Creating", "Making", "Generating"
    ], "A": ["One", "An", "Any"], "Faulty": ["Defective", "Damaged", "Malfunctioning"
    ], "Product": ["Good", "Item", "Commodity"], "That": ["Which", "Who", "That which"
    ], "Caused": ["Provoked", "Led to", "Resulted in"], "Significant": ["Major", "
    Considerable", "Substantial"], "Injuries": ["Harm", "Damage", "Trauma"], "To": ["
    Towards", "In relation to", "With respect to"], "A": ["One", "An", "Any"], "
    Customer": ["Consumer", "Client", "Purchaser"], ".": [",", "!", "?"]}
```