

SPRING 2024

CS7.405

**RESPONSIBLE AND SAFE AI**

Project Report on

**Pataka Patterns: Exploring Hallucinations and Semantic Sizzle in Machine Learning**

By Team Llama

Arush Sachdeva (2023121008)

Yash Shivhare (2021101105)

Vrinda Agarwal (2021101110)

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Serial Number | Topic | Page Number |
| 1 | Abstract | 3 |
| 2 | Index Terms | 3 |
| 3 | Introduction | 3 |
| 4 | Datasets Used | 3 |
| 5 | Models Used | 3-4 |
| 6 | Research Methodology | 4-7 |
| 7 | Results | 7-11 |
| 8 | Conclusions | 11 |
| 9 | References | 11-12 |

1. **Abstract**

The report does the task of establishing the relationship between semantic uncertainty [Semantic entropy is a novel method to address the challenges posed by semantic equivalence in natural language generation (NLG)], lexical similarity [It is a measure of the degree to which the word sets are similar.]and average hallucination [Hallucinations refer to instances where the model generates outputs that deviate from the input data or context.] using various analysis methods such as Wasserstein Distance, AUROC Scores and other important metrics.

1. **Index Terms**

Semantic Uncertainty, Lexical Similarity, Hallucination, Probability Distribution, Wasserstein Distance

1. **Introduction**

If I ask you – What is your name? You might answer - X, It’s X or My name is X. Semantically all of these three things mean the same. Thus, it is expected from anyone who understands the meaning to tell each of the statement confidence.

Excitingly, the model does not provide same confidence value for all the three values as expected. This is the genesis of the problem.

It is then hypothesised that if the model is unable to understand the semantic meaning of sentences, it is highly likely that it hallucinates.

Thus, throughout the paper, we work to find measures for semantic uncertainty, lexical similarity and hallucinations on question answering datasets and try to find through various methods whether or not there exists a relationship between these three entities and how strong is the relationship.

1. **Datasets Used**

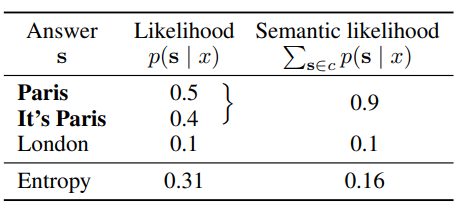
Stanford Question Answering Dataset (SQuAD) and CoQA (Conversational Question Answering Challenge)

1. **Models Used**

* QA Model (deepset/roberta-base-squad2) - It is used for both question answering and checking truthfulness of answer.
* NLI Model (facebook/bart-large-mnli) - It is used to check semantic similarity between sentences using the bidirectional entailment approach. It's a BART model trained on the MNLI dataset, specializing in Natural Language Inference tasks.
* Model (GPT-2) - This model is used for calculating the sequence log probability of generated answers. It's a GPT2 model trained for causal language modeling, which predicts the likelihood.

1. **Research Methodology**

The genesis of problem, as described earlier, was when different confidence and likelihood values were displayed for same answers as per semantics. [1]



Following this, to check it on various datasets, we implemented the code that consisted of following functionalities so as to establish our hypothesis-

1. Generating the answers

This function takes a question and context as input and generates the top N most likely answers using the QA model.

* Steps:
* Tokenization: The question and context are tokenized using the QA tokenizer.
* Model Inference: The tokenized input is passed to the QA model on the GPU to obtain start and end logits.
* Probability Calculation: Softmax function is applied to both start and end logits to convert them into probabilities.
* Answer Span Generation: Possible answer spans are generated by iterating over start and end probabilities and calculating their combined scores.
* Top N Selection and Decoding: The top N answer spans with the highest scores are selected and decoded back into text using the tokenizer.
* Return: The function returns a list of tuples, where each tuple contains an answer string and its corresponding score.

1. Checking Entailment

This function checks the entailment probability between a premise and a hypothesis using the NLI model.

* Steps:
* Tokenization: The premise and hypothesis are tokenized using the NLI tokenizer.
* Model Inference: The tokenized input is passed to the NLI model on the GPU to obtain logits for entailment and contradiction.
* Probability Calculation: Softmax function is applied to the entailment and contradiction logits to get probabilities.
* Return: The function returns the entailment probability.

1. Clustering the answers and calculating average clusters

This function clusters similar answers based on bi-directional entailment with the question using the check\_entailment function.

* Steps:
* Iteration over Answers: For each answer:
* Check if it can be added to an existing cluster:
* Compare the answer with the representative answer of each cluster using bi-directional entailment.
* If both forward and backward entailment probabilities are above a threshold (0.4), add the answer to that cluster.
* If not added to any existing cluster, create a new cluster with the answer.

The function returns a list of clusters, where each cluster is a list of similar answer strings.

Then the function calculates the average number of clusters for correct and incorrect questions.

* Steps:
* Iteration over Question Data:
* Determine if the question has at least one correct answer based on the "correct" flag in the generated answers.
* Accumulate the number of clusters for correct and incorrect questions separately.
* Count the number of correct and incorrect questions.

* Calculation of Averages: Divide the total number of clusters for each category by the number of corresponding questions to get the average number of clusters.

* Return: The function returns the average number of clusters for correct and incorrect questions.

1. Calculating p true values

This function estimates the probability that a possible answer is true (p\_true) given a question and generated answer candidates.

* Steps:
* Prompt Construction: A prompt is built with the question, top N generated answers, and a placeholder for the possible answer.
* Tokenization: The prompt is tokenized using the QA tokenizer.
* Model Inference: The tokenized prompt is fed to the QA model to obtain start and end logits.
* "True" Token Identification: The index of the "True" token in the prompt is found.

1. Calculating Lexical Similarity

This function measures the lexical similarity between generated answers and reference answers using the ROUGE-1 F1 score.

* Steps:
* Initialization:

A ROUGE scorer is created, specifically for ROUGE-1 with stemming enabled.

* Score Calculation Loop:

For each generated answer:

The ROUGE-1 F1 score is computed between the answer and each reference answer.

The maximum ROUGE-1 F1 score among all reference answers is stored.

* Average Score Calculation:

The average of the maximum ROUGE-1 F1 scores across all generated answers is calculated.

* Return:

The function returns the average ROUGE-1 F1 score as a measure of lexical similarity.

1. Calculating Semantic Entropy

This function calculates the semantic entropy of answer clusters based on their probabilities.

* Steps:
* Entropy Initialization: The entropy value is set to 0.
* Iteration over Cluster Probabilities:

For each cluster probability-

If the probability is greater than 0, the entropy is updated using the formula: entropy -= prob \* log2(prob)

* Return: The function returns the calculated semantic entropy value.

1. Calculating sequence log probability [2]

This function calculates the sequence log probability of a generated answer given the question, context, and the CLM model.

* Steps:
* Text Preparation: The input text is created by combining the question and context.
* Tokenization: Both the input text and the answer are tokenized using the CLM tokenizer.
* Log Probability Sum Initialization: A variable to accumulate the log probabilities of each token in the answer is set to 0.
* Iteration over Answer Tokens:
  + - * For each token in the answer (excluding the first token):
      * Previous Tokens Selection: All tokens up to the current position are selected.
      * Target Token Selection: The current token is selected as the target token.
      * Model Inference: The CLM model is used to predict the next token based on the previous tokens.
* Log Probability Calculation: The log probabilities of all possible next tokens are obtained from the model's output logits using the log\_softmax function. The log probability of the actual target token is retrieved.
* Accumulation: The log probability of the target token is added to the running sum.
* Sequence Log Probability Calculation: The accumulated log probability sum is divided by the number of tokens in the answer to get the average log probability per token.
* Return: The function returns the calculated sequence log probability of the answer.

**7. Results**

Initially upon trying to find a correlation between semantic uncertainty and average hallucination on coqa dataset, we received a much expected response and a correlation of 0.968 which establishes a strong correlation between the two entities.

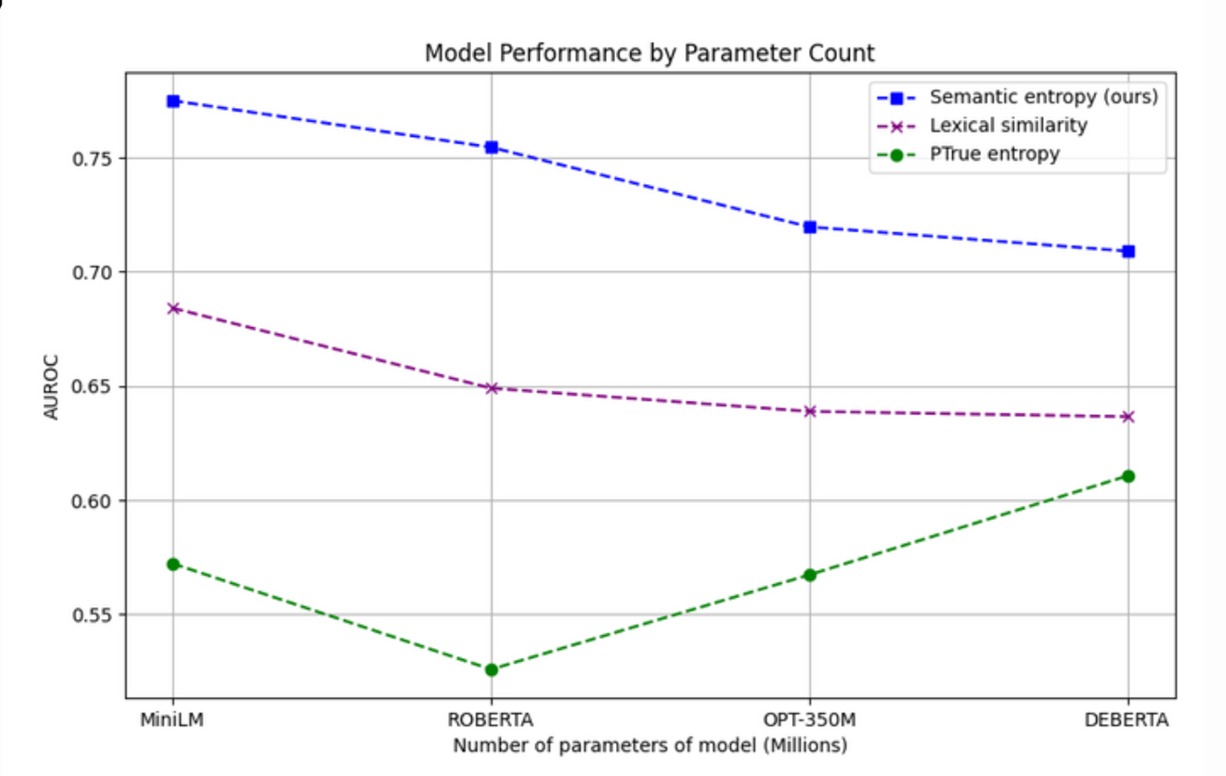
To confirm with our results, we tried to check the same with a larger squad dataset and were astonished to find that the hypothesis does not hold on this dataset and this result changed our view towards the problem.

Given are the metrics that we then used to find the relationship between the various parameters to find a strong relation either to support or to reject null hypothesis.

1. Model Performance by Parameter Count:

- **Metric**: AUROC (Area Under the Receiver Operating Characteristic Curve)

- **Relevance**: This graph compares the performance of various uncertainty estimation methods (Semantic Entropy, Lexical Similarity, PTrue entropy) across different models (MiniLM, ROBERTA, OPT-350M, DEBERTA) based on their parameter count. A higher AUROC value indicates better performance in distinguishing between correct and incorrect outputs. The graph shows that Semantic Entropy generally performs better than the other metrics, particularly in larger models, indicating its robustness in uncertainty estimation.

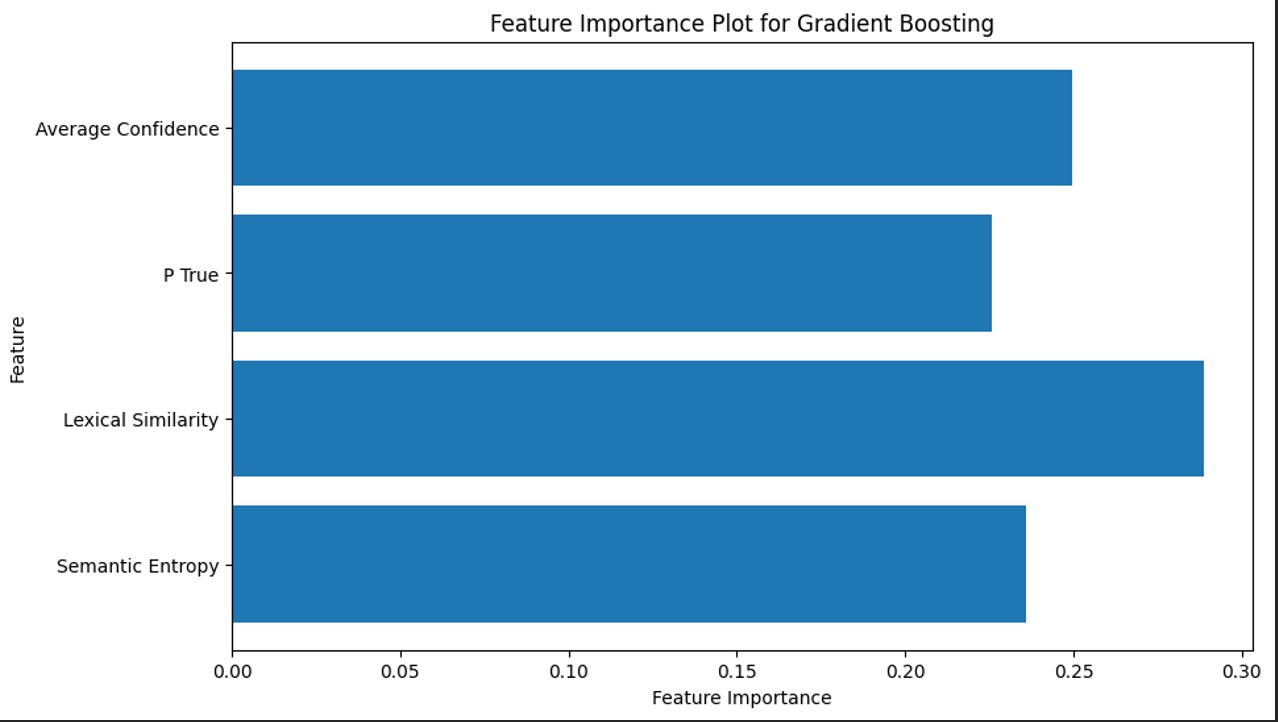


The AUROC values across different models like MiniLM, ROBERTA, OPT-350M, and DEBERTA show that Semantic Entropy is a robust metric for uncertainty estimation, especially in larger models. This implies that Semantic Entropy could be a preferred metric when evaluating or training high-capacity models as it consistently provides reliable differentiation between correct and incorrect outputs. This is crucial for applications requiring high accuracy and reliability.

2. Feature Importance Plot for Gradient Boosting:

- **Metric**: Feature Importance

- **Relevance**: This bar chart shows the relative importance of different features (Semantic Entropy, Lexical Similarity, P True, Average Confidence) used in a gradient boosting model. Feature importance indicates how useful each feature is in the construction of the boosted decision trees within the model. The graph highlights that Lexical Similarity and P True are more influential than Semantic Entropy and Average Confidence in determining the model's decisions, suggesting these features are key drivers in the model’s predictive accuracy.

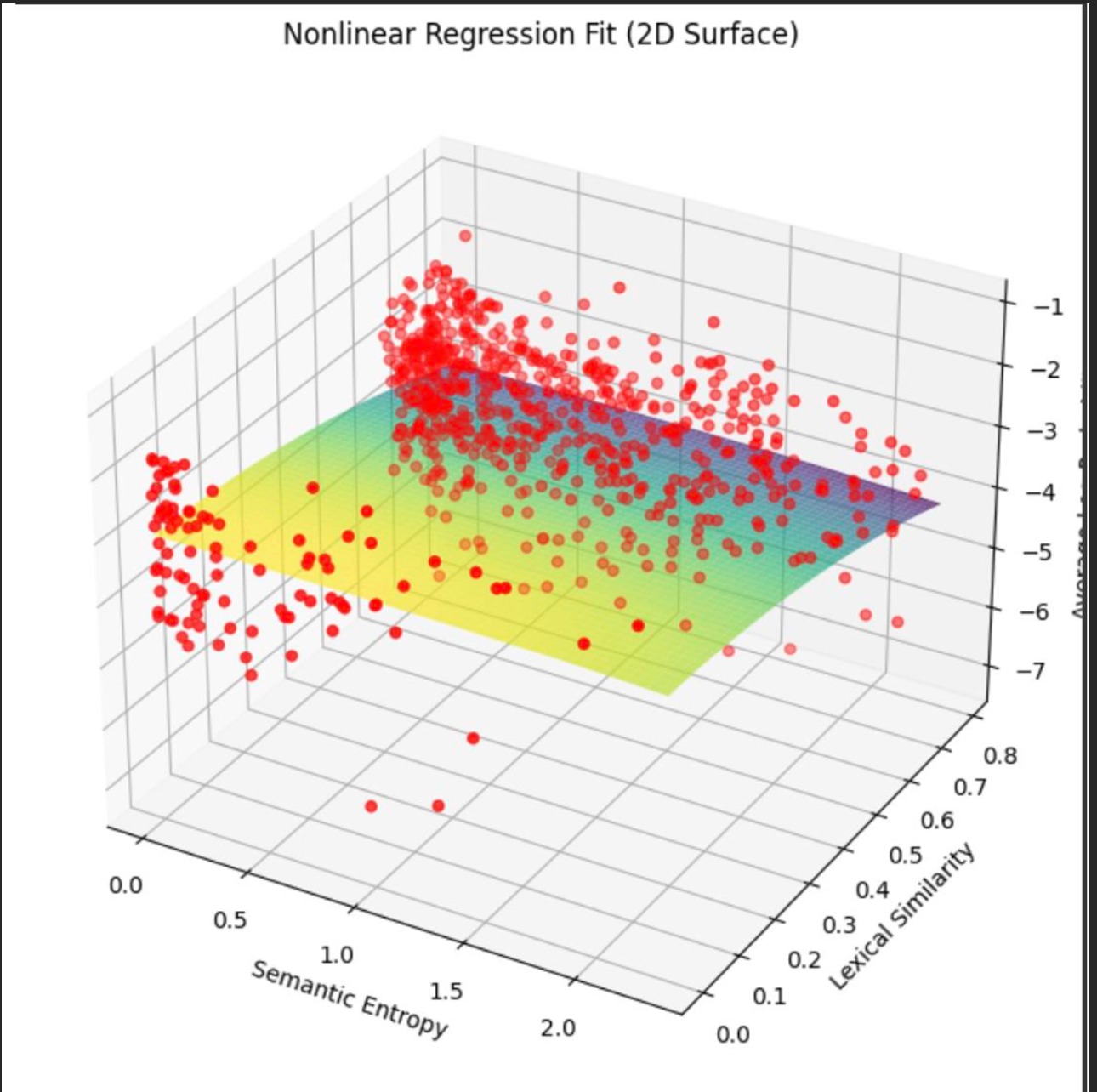


The prominence of Lexical Similarity and P True in the model indicates their critical role in determining the outcomes of the gradient boosting model. This suggests that these features are highly predictive and should be prioritized in model training and feature engineering processes. Knowing which features contribute more to decision-making can help in optimizing model performance and focusing on the most impactful data during training.

3. Nonlinear Regression Fit (2D Surface):

- **Metric**: Multivariate Nonlinear Regression

- **Relevance**: This 3D surface plot visualizes the relationship between Semantic Entropy, Lexical Similarity, and the model's output (measured in some dependent metric like error or log probability). It helps to understand how changes in Semantic Entropy and Lexical Similarity correlate with changes in model performance, providing insights into the complex interdependencies between these factors.

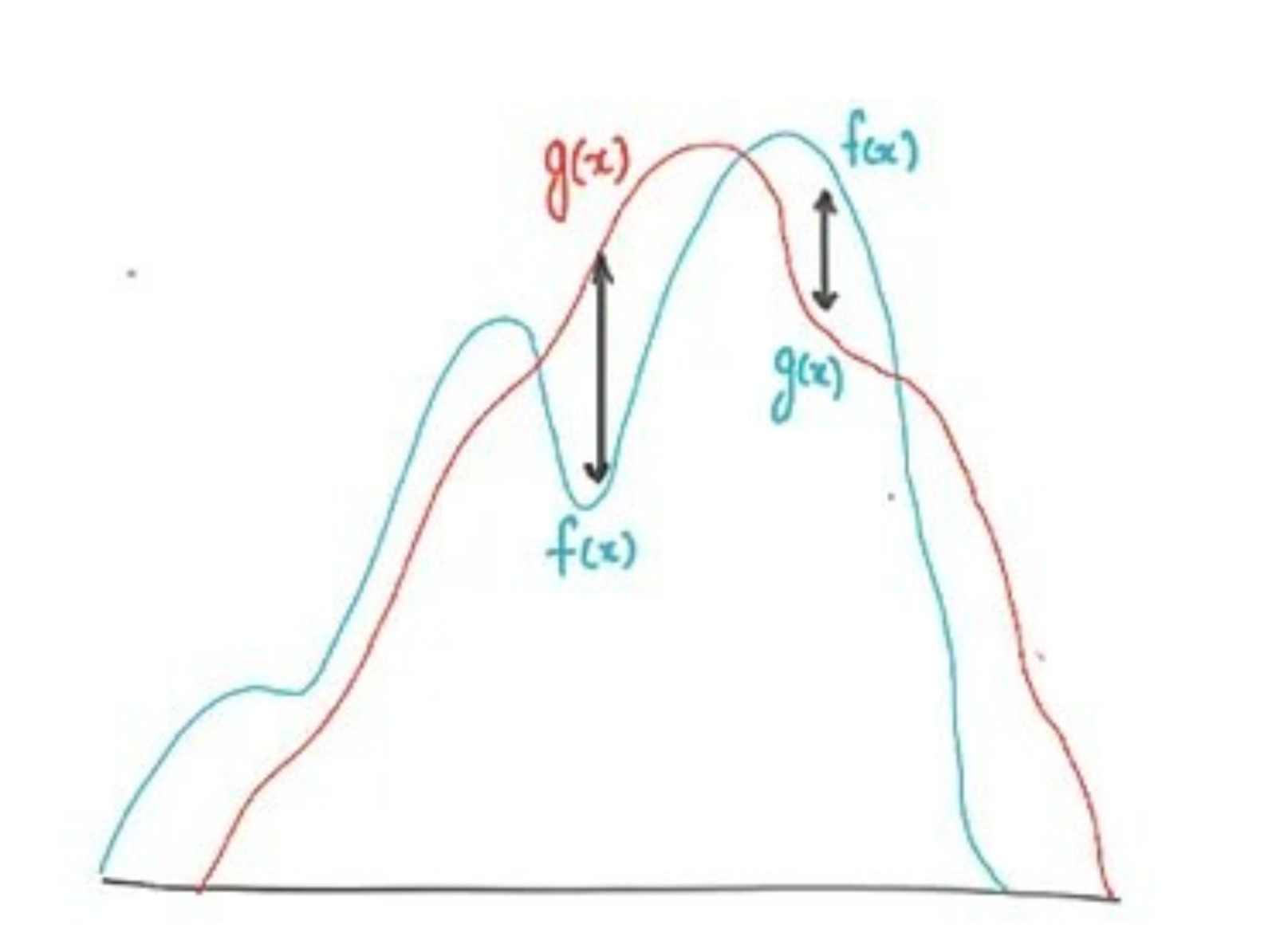


This plot illustrates the complex relationships between Semantic Entropy, Lexical Similarity, and model outputs, indicating that changes in these input metrics can significantly impact performance. Understanding these relationships helps in fine-tuning model parameters to achieve better accuracy and efficiency. It also aids in identifying potential trade-offs between improving one metric over another, which can be critical for balancing model performance across different evaluation criteria.

4. Simple Illustration of Probability Distributions (g(x) and f(x)):

- **Metric**: Conceptual Visualization

- **Relevance**: This diagram likely represents the distribution of two variables or two states of a model, illustrating their peaks (modes) and how they overlap or differ. This can be useful for explaining concepts like the Wasserstein Distance (the effort required to transform one distribution into the other) or to visually represent the divergence between two different model outputs or conditions.



By showing how two distributions compare and differ, this visualization aids in understanding model behavior under varying conditions, such as how changes to input features or model parameters affect output distributions. This can be particularly useful in tasks like anomaly detection, where understanding the distance between normal and anomalous data distributions is key. The low value of Wasserstein distance (=3.1) suggests the nearness of the joint probability distribution of semantic entropy and lexical similarity with the average hallucinations.

**8. Conclusion**

We can conclude from the results that the low Wasserstein distance value implies nearness of the probability distributions. Intuitively, if each distribution is viewed as a unit amount of earth (soil) piled on 𝑀, the metric is the minimum "cost" of turning one pile into the other, which is assumed to be the amount of earth that needs to be moved times the mean distance it has to be moved. AUROC scores indicate that semantic entropy is a robust metric for uncertainty estimation and similarly p true and lexical similarity is a good measure as per gradient boosting that is why we plotted the 3D graph for better understanding of the concepts.

**References**

[1] Kuhn, L., Gal, Y., & Farquhar, S. (2023). Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. arXiv preprint arXiv:2302.09664.

[2] Guerreiro, N. M., Voita, E., & Martins, A. F. (2022). Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation. arXiv preprint arXiv:2208.05309.