COURSE PROJECT

Efficiency Optimization of Detect-GPT

CS772: PROBABILISTIC MACHINE LEARNING

SPRING 2025

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Submitted by

Abhiraj Akhouri (218170032**)** **Aman Khilani (**210109**)**

Priyanshu Tiwari (210789**)** **Nishvaan Sai H (**200648**)**

Submitted to:

Dr. Piyush Rai

Department of Computer Science

Indian Institute of Technology Kanpur

I. Problem Description

The rapid advancement and widespread deployment of large language models (LLMs) such as GPT-3, GPT-4, and LLaMA have raised significant concerns about the misuse of machine-generated text, including misinformation, academic dishonesty, and content authenticity. DetectGPT is a state-of-the-art zero-shot method that distinguishes between human- and machine-generated text by analyzing the probability curvature of text under perturbations. However, DetectGPT's high computational cost-requiring hundreds of queries per sample-limits its practical deployment.

This project aims to optimize the efficiency of DetectGPT by leveraging Bayesian surrogate modeling and novel sampling strategies, to reduce query requirements while maintaining or improving detection performance.

II. Literature Review

Our project builds over the work of 3 main papers:

- **DetectGPT**: Introduced as a robust zero-shot detector, DetectGPT evaluates the log-probability curvature of perturbed text samples using the source LLM. It achieves strong detection accuracy even against unseen models and paraphrased text but is computationally expensive due to the large number of perturbation queries needed for each sample.

- **Bayesian Surrogate Models**: Miao et al. (2024) proposed replacing the brute-force perturbation approach with a Bayesian Gaussian Process (GP) surrogate model. By actively selecting and scoring only the most informative ("typical") samples, their method interpolates detection statistics across the perturbation space, drastically reducing the number of LLM queries required. Their approach outperforms DetectGPT in efficiency and accuracy, especially under low-query budgets.

- **Fast-DetectGPT**: Bao et al. (2024) introduced Fast-DetectGPT, which replaces DetectGPT's perturbation step with a more efficient sampling strategy based on conditional probability curvature. Fast-DetectGPT achieves up to 75 percent better detection accuracy and is up to 340 times faster than the original DetectGPT, making it highly suitable for real-world deployment.

III. Novelty & Improvements over Prior Work

Our project builds on the Bayesian Surrogate Model approach. Instead of querying the LLM for hundreds of perturbed samples, the Bayesian surrogate model uses a Gaussian Process (GP) to estimate log-probabilities across the perturbation space with far fewer queries.

Steps:

1. **Perturbation Generation:**
   For a text x, generate perturbed versions $X = \{x_i\}_{i<N}$ using a perturbation model $q(.|x)$
2. **Initial Query:**
   Initialize a small subset X_t from X and query the LLM for their log-probabilities $y_t = \{log\, p_\theta(x_t)\}$. x and x_1, i.e, original text and first perturbation may be used.
3. **Gaussian Process Regression:**
   - Train a GP to model f: $X -> R$, mapping each perturbation to its log-probability based on X_t set.
   - GP is defined by a kernel function $k(x, x')$ and hyperparameters.
4. **Sequential Learning:**
   - At each step, pick the perturbation x* in X not yet queried that has the highest predictive variance under the GP
   - Query the LLM for $log\, p_\theta\,(x\ *)$, add x* to X_t, update the GP.
5. **Hyperparameter Optimization:**
   - Optimize GP hyperparameters (like kernel parameters and noise variance) using marginal likelihood.
6. **Detection Statistic:**
   - After reaching the query budget, use the GP to predict log-probabilities for all remaining perturbations (instead of model probability).
   - Compute the detection statistic (e.g., curvature) using these predictions.
7. **Decision:**
   - If the detection statistic $l(x, p_\theta, q)$ exceeds a threshold, classify x as LLM-generated; otherwise, as human-written.

Our project proposes novel improvements in the Bayesian Surrogate Model approach in these 4 domains:

A. Enhanced Kernel Designs

Description: The Surrogate Model approach incorporates a basic BertScore kernel. Move beyond the basic BertScore kernel by introducing:

- **Enhanced BertScore Kernel:** Incorporates contextual weights and aggregates information from multiple transformer layers.
- **Semantic Structure Kernel:** Blends semantic similarity (from embeddings) with syntactic structure (from parse trees or dependency graphs).
- **Hybrid Kernel:** Combines multiple similarity measures, such as BertScore, TF-IDF, and syntactic similarity.

Justification:

- The original BertScore kernel captures only surface-level semantic similarity. By integrating contextual weights, multiple layers, and syntactic information, the kernel can better model nuanced relationships between text perturbations and their likelihood under the LLM.
- Hybrid kernels capture complementary aspects of similarity (semantic, syntactic, lexical), leading to more accurate interpolation of log-probabilities in the GP surrogate. This is particularly valuable for adversarial or paraphrased inputs, where surface similarity alone may be misleading.
- Empirical evidence from NLP literature shows that combining semantic and syntactic information improves performance on a variety of text similarity and classification tasks, suggesting these kernels will yield better detection accuracy and robustness.

B. Adaptive Hyperparameter Tuning

Description:

- Employ a suite of optimization methods (Adam, L-BFGS, Scipy optimizers, Bayesian optimization) for tuning GP hyperparameters.

- Use adaptive learning rates, early stopping, and automatic cross-validation for robust hyperparameter selection.

Justification:

- The performance of Gaussian Process models is highly sensitive to kernel and noise hyperparameters. The original approach may use fixed or manually-tuned hyperparameters, which can be suboptimal across different datasets or text types.
- Adaptive and automated optimization ensures that the GP model is well-calibrated for each scenario, reducing overfitting and improving generalization.
- Cross-validation and Bayesian optimization systematically search the hyperparameter space, leading to more reliable and reproducible performance gains compared to ad hoc tuning.

C. Active Learning Strategies

Description: The Bayesian Surrogate approach employs highest Variance sampling for sequential sampling of typical samples. Expand beyond uncertainty (variance) sampling to include:

- **Diversity-Based Sampling:** Selects samples that are diverse in the embedding space.
- **Expected Model Change:** Chooses samples that would most impact the GP model.
- **Expected Improvement:** Balances exploration (uncertainty) and exploitation (potential gain).
- **Hybrid and Adaptive Sampling:** Dynamically combines multiple criteria.

Justification:

- The original Bayesian surrogate approach primarily uses uncertainty sampling, which can lead to redundant queries in regions of high uncertainty but low information gain.

- Diversity-based and expected model change strategies ensure that the queried perturbations cover the space more effectively, leading to faster and more robust learning of the GP surrogate.

- Hybrid and adaptive strategies can dynamically select the most informative samples based on the current state of the model, improving query efficiency and detection performance, especially under tight query budgets.

D. Robustness to Adversarial Attacks

Description:

- Introduce adversarial training using perturbed samples (synonym substitution, word deletion, sentence restructuring etc.).

- Enhance detection methodology to be resilient against paraphrasing and other adversarial manipulations.

Justification:

- The original Bayesian surrogate model may be vulnerable to adversarial attacks that alter surface form without changing underlying meaning, leading to false negatives.

- Adversarial training exposes the model to a wider range of perturbations during training, improving its ability to generalize and detect LLM-generated text even when adversaries attempt to evade detection.

- Robust detection techniques ensure that the model maintains high accuracy and reliability in real-world scenarios where adversarial attempts are likely.

Summary Table

Improvement Area	**Description**	**Justification**
Enhanced Kernel Designs	Richer, hybrid, and structure-aware kernels	Capture deeper text relationships, improve robustness
Adaptive Hyperparameter Tuning	Automated, robust optimization methods	Better model fit, less overfitting, improved generalization
Active Learning Strategies	Multiple, adaptive query selection criteria	More informative queries, faster and more robust learning
Robustness to Adversarial Attacks	Adversarial training and detection	Maintains performance under attack, more practical deployment

IV. Tools/Softwares Used

- **Programming Language**: Python
- **Libraries**: PyTorch, HuggingFace Transformers (for LLMs), GPyTorch (for Gaussian Processes), Scikit-learn, Numpy, Scipy and other standard libraries
- **Datasets**: OpenAI GPT-2 and GPT-3 outputs, LLaMA outputs, WritingPrompts, XSum
- **Computing Resources**: IITK CC GPU Server (for LLM architecture)

V. Experimental Implementation

The following Github repository contains the implementation of our proposed ideas.

VI. Things Learnt through Project

- LLM Architecture: Learned the theoretical foundations of LLM, its model scores, importance of transformers, high dimensional semantic meaning of text etc.
- Familiarity with research literature of DetectGPT, identifying novel ideas through existing papers, designing and analyzing experiments
- Introduction to various optimization methods especially those of Active Learning.
- Practical experience with large-scale code & experiment management, LLM APIs, advanced Bayesian modeling techniques, need & use of GPUs in LLM experiments.

VII. Scope of Future Work

While we were able to develop an implementation of our proposed ideas, quantifying and comparing results across different ideas, settings and hyperparameters was limited due to computational and time constraints, especially on large datasets. For future work, we are interested in tabulating these results and building a fast version of Detect GPT based on them.

Specifically, we want to test our implementation on large datasets to understand its key characteristics. We would need GPU access to carry this out.

VIII. Member Contributions

The project ideas were finalized through regular meets among all group members. Coding was also undertaken by all members, allowing us to deploy simultaneously on 4 machines. The work at each juncture was divided and carried out based on member interest, knowledge and availability.