**Group Name: Fly By Night**

**Applying Ensemble Methods to Zero-Shot Machine-Generated Text Detection**

**1. Introduction**

The increasingly articulate and realistic nature of texts generated by large-language models (LLM) have enabled their use in applications such as drafting essays, summarising information, as well as writing code proof-of-concepts. This also raises concerns about factual inaccuracies in journalism (The Verge, 2023)[[1]](#endnote-2), the potential corruption of technical knowledge (Rodriguez et al., 2022)[[2]](#endnote-3), misattribution of work extending to academic plagiarism, and the facilitating of phishing techniques and other scams. Researchers have hence recently been tackling the detection problem of identifying whether a candidate piece of text is machine or human-generated. However, this is complicated by the rapidly-evolving LLM landscape, with new models being frequently rolled out and the development of adversarial methods to circumvent detection methods.

In this paper, we modify the DetectGPT model developed in Mitchell et al. (2023)[[3]](#endnote-4), which uses perturbations of the existing text to estimate the local curvature of the sample’s log-probabilities as opposed to relying on the raw values to estimate some conditional probability distribution for classification. We extend DetectGPT by relaxing the constraint that the generative (or base) model and the discriminative (or scoring) model are the same, mirroring real-life scenarios where the model that generates the text (were it to be even machine-generated) is unknown. Mitchell et al. show that detection performance is expectedly lower when this assumption is lifted, with significant variations in accuracy (as measured by area under the Receiver-Operating Curve, or AUROC). To tackle this question, we leverage on the ensembling of the log-probability outputs of multiple DetectGPT models (each assuming a separate source model) using classifiers in the machine-learning literature such as logistic regression and random forests. Our paper hence combines the black-box approach with the white-box approach, to attempt to achieve an improvement in detection accuracy without any base model assumptions.

**2. Related Work**

Current approaches to the detection problem make assumptions that the text was generated by a particular base model, to which the defender has different levels of access depending on the problem setting. We adopt the classification approach outlined by OpenAI (Solaiman et al., 2019)[[4]](#endnote-5).

The *black-box setting* presumes only access to the model outputs. Black-box approaches mainly consist of using sample text features such as TF-IDF (term frequency-inverse document frequency), skip-gram, continuous-bag-of-words (CBOW)[[5]](#endnote-6) and sometimes auxiliary features (such as account data in the setting of social media bot detection), and fitting a binary classifier to them, which can range from logistic regression models to convolutional neural networks[[6]](#endnote-7) or LSTM models[[7]](#endnote-8). However, high classification accuracy for these methods are reliant on sufficiently-long text length and a sufficiently-diverse corpus of training machine-generated samples in terms of stylometric and linguistic characteristics in order to prevent overfitting. As such, these classifiers need to be continually trained and updated, limiting their usefulness.

In contrast, the *white-box* setting repurposes the generative model as a classifier, and provides the defender access to some or all of the model’s inner attributes, such as log-probabilities and scores. White-box approaches can be further classified into two subcategories: I) zero-shot models which aim to classify sample texts without any further inputs, and ii) fine-tuning based models, which further provide the model with domain-specific texts. Under the assumption that the discriminative model is also the generative model, white-box approaches tend to be highly effective even using simple metrics such as the average log probability of all tokens within the candidate text (Solaiman et al., 2019).

**3. Approach**

The key observation underpinning DetectGPT as noted by Mitchell et al. (2023) is the hypothesis that LLMs implicitly choose samples to maximise the log-probability of successive tokens when generating text, and hence tend to sample from regions of negative curvature of the log-probability function of p\_theta. Thus, by perturbing an original text x into a rewritten text x\_tilde which is semantically similar to x, we can compute the perturbation discrepancy d, defined to be log(p\_theta(x)/p\_theta(x\_tilde)). The perturbation discrepancy is then expected to be positive and larger for text if the text were indeed generated by the base model. In practice, normalising the perturbation discrepancy is more effective. The text perturbation can be performed consistently and at scale using an off-the-shelf mask-filling model such as T5.

We adopt a similar training pipeline by adapting the code developed by Mitchell et al (2023). We use English news articles from the Extreme Summarization (XSum) dataset (Narayan et al., 2018)[[8]](#endnote-9) and German translations of the WMT16 dataset. We then generate synthetic samples from the raw conditional distribution of the corresponding base model using the first 30 tokens as input, with a temperature of 1. We use smaller models than current state-of-the-art LLMs for the mask-filling and scoring models to evaluate the accuracy of our approach on less-complex models, to decrease the relatively long compute time already required by the vanilla DetectGPT method. We use the following source, mask-filling, scoring models and datasets during our experiments; however, our methodology can be easily extended to encompass additional models. We use the same set of hyperparameters for all experiments, such as masked span length, fraction of words masked for rewriting and number of perturbations.

We attempt various methods to aggregate the outputs of the individual classifiers, for three categories: 1) Baseline results from the individual classifiers; 2) Ensembling using simple summary statistics; 3) Machine-learning classifiers. We assess the accuracy of each method using the Area under the Receiver Operating Curve (AUROC), which can be interpreted as the probability that a random human-written text is ranked lower (i.e. is more “human”) than a random machine-generated text. The second category consists of using the arithmetic mean, median and maximum of the scores of all classifiers, mimicking a soft-voting approach. However, we note that each individual classifier is evaluating a slightly different question, as each one is assessing the likelihood that the sample text is generated by their model. The third category uses a logistic regression model, a random forest, a Naive-Bayes Gaussian model and a support vector machine. All models are implemented using scikit-learn. This approach requires prior training, and thus work best with a fine-tuning dataset similar to the fine-tuning approaches mentioned earlier, although it is used to train our additional classifier and not the LLMs themselves. We also attempt a “multi-stage” classifier that orders the DetectGPT models by decreasing complexity and successively considers each model. If the perturbation discrepancy d is below a threshold, which we take to be mu – 2\*SD where mu and SD are taken over the training dataset, we halt and take the mean of all the models we have considered so far. This approach aims to reduce the likelihood of false positives by combining the idea of taking the maximum of all DetectGPT models with the hypothesis that more complex scoring models, by virtue of having more representational power to “encompass” sub-models, are more accurate at evaluating if a text is machine-generated.

**4. Results**

Our results for the three categories are shown in Tables 1, 2 and 3 respectively. We observe that our first category of ensembling methods perform better than the individual DetectGPT classifiers in isolation when the scoring model is not the source model. In addition, our second category of ensembling methods performs better than the first category, reaching an accuracy of about 0.99. We observe that when gpt2-medium is used as the base model, combining the outputs of the additional DetectGPT classifiers still yields an overall improvement in accuracy of about 0.01 for German and 0.03 for Xsum.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Base Model** | **Drop Scoring Base** | **Dataset** | **EleutherAI-gpt-neo-125m** | **bert-base-cased** | **gpt2-medium** | **gpt2** | **roberta-base** |
| EleutherAI-gpt-neo-125m | False | German | **0.95** | 0.37 | 0.78 | 0.82 | 0.47 |
| XSum | **1** | 0.63 | 0.25 | 0.7 | 0.84 |
| True | German | NA | 0.37 | 0.78 | 0.82 | 0.47 |
| XSum | NA | 0.63 | 0.25 | 0.7 | 0.84 |
| gpt2 | False | German | 0.28 | 0.27 | 0.84 | **0.99** | 0.56 |
| XSum | **0.93** | 0.56 | 0.54 | **0.98** | 0.76 |
| True | German | 0.28 | 0.27 | 0.84 | NA | 0.56 |
| XSum | **0.93** | 0.56 | 0.54 | NA | 0.76 |
| gpt2-medium | False | German | 0.31 | 0.28 | **0.98** | **0.97** | 0.59 |
| XSum | **0.9** | 0.55 | **0.93** | **0.9** | 0.65 |
| True | German | 0.31 | 0.28 | NA | **0.97** | 0.59 |
| XSum | **0.9** | 0.55 | NA | **0.9** | 0.65 |
| Average (All Models) | | | 0.68 | 0.45 | 0.67 | 0.87 | 0.64 |
| Average (Without Base Model) | | | 0.6 | 0.73 | 0.79 |

**Table 1:** Results for first category (baseline methods)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Base Model** | **Drop Scoring Base** | **Dataset** | **Average Accuracy of Individual Models** | **Max** | **Mean** | **Median** |
| EleutherAI-gpt-neo-125m | False | German | 0.68 | 0.95 | 0.89 | 0.78 |
| XSum | 0.68 | 0.78 | 0.92 | 0.85 |
| True | German | 0.61 | 0.63 | 0.73 | 0.77 |
| XSum | 0.6 | 0.32 | 0.72 | 0.74 |
| gpt2 | False | German | 0.59 | 0.91 | 0.83 | 0.72 |
| XSum | 0.75 | 0.82 | 0.95 | 0.93 |
| True | German | 0.49 | 0.55 | 0.43 | 0.45 |
| XSum | 0.7 | 0.56 | 0.88 | 0.91 |
| gpt2-medium | False | German | 0.63 | 0.93 | 0.91 | 0.81 |
| XSum | 0.79 | 0.93 | 0.96 | 0.91 |
| True | German | 0.54 | 0.78 | 0.68 | 0.64 |
| XSum | 0.75 | 0.9 | 0.93 | 0.88 |
| Average (All Models) | | | 0.65 | 0.75 | 0.82 | 0.78 |

**Table 2:** Results for second category (simple summary statistics)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Base Model** | **Drop Scoring Base** | **Dataset** | **Average Accuracy of Individual Models** | **Logistic Regression** | **Random Forest** | **Naive-Bayes** | **Support Vector Machine** | **Multi-Step** |
| EleutherAI-gpt-neo-125m | False | German | 0.68 | 0.95 | 0.98 | 0.92 | 0.97 | 0.82 |
| XSum | 0.68 | 1 | 1 | 1 | 1 | 0.43 |
| True | German | 0.61 | 0.78 | 0.8 | 0.77 | 0.82 | 0.74 |
| XSum | 0.6 | 0.98 | 0.97 | 0.95 | 0.98 | 0.36 |
| gpt2 | False | German | 0.59 | 1 | 1 | 1 | 1 | 0.83 |
| XSum | 0.75 | 1 | 1 | 1 | 1 | 0.68 |
| True | German | 0.49 | 0.93 | 0.92 | 0.9 | 0.92 | 0.83 |
| XSum | 0.7 | 1 | 0.99 | 0.98 | 1 | 0.67 |
| gpt2-medium | False | German | 0.63 | 0.99 | 0.99 | 0.98 | 0.98 | 0.97 |
| XSum | 0.79 | 0.97 | 0.95 | 0.94 | 0.97 | 0.95 |
| True | German | 0.54 | 0.99 | 0.98 | 0.97 | 0.97 | 0.38 |
| XSum | 0.75 | 0.96 | 0.95 | 0.93 | 0.95 | 0.93 |
| Average (All Models) | | | 0.65 | **0.96** | **0.96** | **0.95** | **0.96** | 0.72 |

**5. Discussion**

**Limitations**

In addition to the known limitations of DetectGPT, which are its relatively long compute time and requirement of access to the log-probabilities of the base and scoring models, our current ensembling methods are performed on a relatively small dataset and using a limited range of models. As such, more work can be done to validate the approach on other datasets as well as to create a ‘minimal set’ of LLMs that when combined with our method can reliably and accurately classify any sample text. Furthermore, while simply taking the mean and maximum of individual DetectGPT classifiers to observe marginal increases in accuracy, a fine-tuning dataset as training data for the random forest classifier is required for more significant accuracy improvements.

**Conclusion**

We apply ensembling methods to the zero-shot, white-box DetectGPT approach developed by Mitchell et al., which estimates local probability curvature of the sampled text, by combining the outputs from individual DetectGPT classifiers that are each trained on a different source model. We find that a random forest classifier is highly accurate at detection as compared to the individual classifiers.

1. Sato, M. CNET pauses publishing AI-written stories after disclosure controversy. <https://www.theverge.com/2023/1/20/23564311/cnet-pausing-ai-articles-bot-red-ventures>. 2023. Accessed 2023-04-30. [↑](#endnote-ref-2)
2. Juan Rodriguez, Todd Hay, David Gros, Zain Shamsi, and Ravi Srinivasan. 2022. Cross-Domain Detection of GPT-2-Generated Technical Text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1213–1233, Seattle, United States. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.88/ [↑](#endnote-ref-3)
3. Mitchell E., Lee, Y., Khazatsky, A., Manning C., Finn, C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. URL (Paper): <https://arxiv.org/abs/2301.11305>. URL (Code): https://github.com/eric-mitchell/detect-gpt [↑](#endnote-ref-4)
4. Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert- Voss, A., Wu, J., Radford, A., and Wang, J. Release strategies and the social impacts of language models, 2019. URL: https://arxiv.org/abs/1908.09203 [↑](#endnote-ref-5)
5. Reddy, H., Raj, N., Gala, M. and Basava, A. Text-mining-based Fake News Detection Using Ensemble Methods. *Int. J. Autom. Comput.* **17**, 210–221 (2020). https://doi.org/10.1007/s11633-019-1216-5 [↑](#endnote-ref-6)
6. Weller, W. and Woo, J. Identifying Russian Trolls on Reddit with Deep LLearning and BERT Word Embeddings. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15739845.pdf [↑](#endnote-ref-7)
7. Kudugunta S. and Ferrara E. Deep Neural Networks for Bot Detection. URL: https://arxiv.org/abs/1802.04289 [↑](#endnote-ref-8)
8. Narayan, S., Cohen, S. B., and Lapata, M. Don’t give me the details, just the summary! Topic-aware convolutionalneural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 2018. [↑](#endnote-ref-9)