What the text?
Recognizing AI-generated text

Amzi Jeffs, Junichi Koganemaru, Salil Singh, Ashwin Tarikere

GitHub: https://github.com/jkoganem/fakereview/

Overview: With the recent explosion in large language models, AI-generated text is become ubiquitous online, from falsified product reviews to politically motivated social media spam. Our project aims to develop machine learning pipelines to recognize authentic human text versus AI-generated text in a variety of contexts. We found that out-of-the-box techniques yielded at most 75% accuracy in this task, while a fine-tuned neural network provided significant improvements with 92% accuracy.

Data: We compiled data from four existing datasets, randomly selecting 10,000 human generated and 10,000 AI-generated samples from each to maintain balance in the data.

- Product reviews generated with GPT-2
  https://www.kaggle.com/datasets/mexwell/fake-reviews-dataset
- Wikipedia intro paragraphs generated by GPT-3 Curie
  https://huggingface.co/datasets/aadityaubhat/GPT-wiki-intro
- News articles generated by Grover
  https://github.com/rowanz/grover/tree/master
- Essays generated by a variety of models (ultimately excluded, explained below).
  https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset

This data provided a suitable variety of text types, as well as variety in the generation models.

Analysis: Our baseline approach was to use an SBERT model to embed our text samples in a high-dimensional vector space, then use classical techniques such as SVM and XGBoost. We quickly noticed that our baselines were performing suspiciously well on the essay data (95+% accuracy), and upon further investigation we found that this data set was generated in a narrow manner that led to artificial clustering in the SBERT embedding. Specifically, the essays data reused a small set of prompts when generating essays, causing repetitive outputs from the models that effectively created data leakage between the training and validation sets. As such, we decided to throw out the essay data and restrict our analysis to the remaining samples.

On the remaining samples, we extensively optimized the hyperparameters for XGBoost using Optuna, since XGBoost had performed the best among our baselines. We found that XGBoost was highly sensitive to differences in the hyperparameters, with results ranging from 50% accuracy to 75% accuracy. The best XGBoost models achieved approximately 75% accuracy on our validation set.

To move beyond the out-of-the-box SBERT, we attached a linear classification head and performed fine-tuning. With the SBERT model frozen, the linear classifier only achieved 69% accuracy, but when fine-tuning the entire model we achieved 88% validation accuracy after 20 epochs. Even better, the fine-tuned model achieved 92% accuracy on our final holdout set. Our work confirms that fine-tuning existing language models is a valuable approach for detecting AI-generated text content.

Future Work:

- *Broadening our dataset.* To deploy our model "in the wild," we would need to train it on much broader datasets with varying contexts. As we experienced with the essays data, collecting or creating good data must be approached carefully and thoroughly.
- *Incorporating contextual image input.* In many use cases such as social media posts, text content is often accompanied by images. It would be valuable to train our model to incorporate image input which accompanies text in these contexts.
- *Incorporating new data on the fly.* Our model would face significant challenges recognizing AI text that is generated by new models. To combat this, we should create pipelines for quickly incorporating new data into the model. Fine-tuning the entire model on new data is costly and time consuming, and thus we should experiment with freezing portions of the model and fine-tuning on the rest.