AI Detect Methods in Text: Distinguishing Between Student-Written and LLM-Generated Essays

Xuhang He, Zheng Cao

December 2023

Link to the final presentation: Final Presentation Video

1 Introduction

With the rise of ChatGPT, it has become increasingly common for students to use AI tools to write essays. This trend poses challenges for professors in determining academic dishonesty and fairly grading student work. This project aims to develop machine learning models capable of accurately identifying whether an essay has been written by a student or generated by a Large Language Model (LLM). We primarily focus on two machine learning approaches: Recurrent Neural Networks (RNN) and Bidirectional Encoder Representations from Transformers (BERT). Thus, aside from developing a classfication model for LLM-generated essays, we will also offer an insightful comparison of RNN and BERT's capabilities in distinguishing human-written text from that generated by machines.

2 Literature Survey

Current approaches for detecting AI-generated essays involves both linguistic and technical approaches. Linguistic methods include analyzing the features of the text, such as coherence, style, grammar, and vocabulary, to identify anomalies or inconsistencies that indicate the presence of an AI model (Corizzo and Leal-Arenas 7901). Technical methods use machine learning techniques to classify the text based on its content or structure, such as neural networks, transformers, or one-class learning (Crotthers, Japkowicz, and Viktor). However, both methods face limitations and open problems, such as the diversity of AI models, the quality of human-written text, the evaluation metrics, and the ethical implications (Kauranen-Kuorinen et al.; Alshammariyeh and Alshammariyeh 1). Our research aims to contribute to this evolving field by implementing and comparing popular machine learning approaches.

3 Dataset Description

Our training dataset consists a total of 2,762 essays. This dataset is split into two parts: 1,378 essays sourced from Kaggle, predominantly written by students, and 1,384 essays generated using Google's PaLM model from Gen-AI. These essays responds to the same

two topics:car-free cities and "Does the electoral college work?" Additionally, the hidden test dataset consists of 10,000 essays, responding to seven distinct prompts, including "Exploring Venus" and "Seeking multiple opinions". This diverse dataset allows us to rigorously test our models and make it robust for AI-generated detection.

4 First Approach: Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a commonly used model in natural language processing. It is an important block in seq2seq models. We first thought about using RNN to solve the problem. Each passage is transformed into a one-hot embedding vector with length of the entire vocab size 18632. Each article is one sample in a batch with each article divided into pieces of size step_size 30 to create a sample of size (passage_length / step_size, step_size, vocab_size). For one training and testing iteration, batch_size is set to be the same as the step_size 30. Prompt texts are also processed in a similar way except that the shape for each prompt only two dimensional (prompt_length, vocab_length). The hidden state is initialized with all the information in the prompt for each sample. The final hidden state that contains the information of the prompt and passage is reshaped into a tensor with length step_size*num_hidden, where num_hidden is a hyperparameter, which would be passed into a classifier with five layers of fully connected layers and produce an output of shape (1, 2). Cross entropy loss is used as the loss function, and Adam is used as the updater.

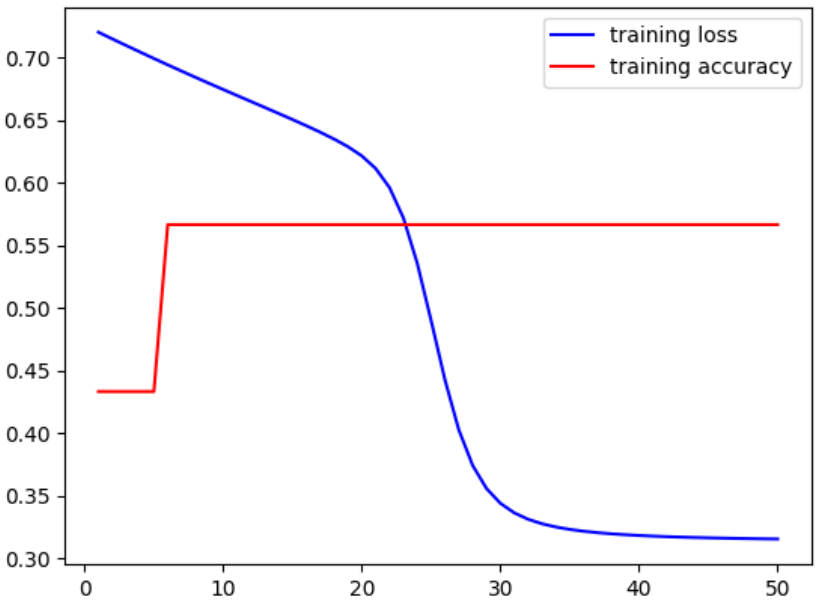

Figure 1: Loss and accuracy of RNN classification

However, as shown in Figure 1, although the model fits the training set well, it lacks the capacity to generalize and predict on test dataset. We believe that it is because the embedding of one-hot encoding cannot be learnt. Also, RNN requires comparing the

outcome per time step with the label to optimize the neurons in RNN, yet in this model, weight optimization is not done until the entire is passed into hidden layer. Finally, hidden state is of size time_step * num_hiddens, which may lack the capacity of storing information per passage such as position information. Due to the reasons above, we decided to use BERT and transformer encoder to fit the model.

5 Second Approach: Bidirectional Encoder Representations from Transformers (BERT)

BERT, standing for Bidirectional Encoder Representations from Transformers, revolutionizes natural language processing with its unique approach. It employs a two-step process: pre-training on a large corpus to understand language context and fine-tuning for specific tasks. For classifying AI-generated languages, we attached four fully connected layers following the last output of the encoder.

We begin the BERT implementation by tokenizing and padding our essays. We utilize a tokenizer from torchtext.data.utils to tokenize our dataset. A custom Vocab class creates a vocabulary with tokens appearing at least 5 times whicch holds a token-to-index map. We then replace each token in the texts with its relevant index(ranking of number of appearances in the texts). Then each essay is either sliced or padded to the size of 512 tokens. Then we implemented the BERT model using nn.modules with following configurations:

- Hidden size (num_hiddens): 512. The dimension of each transformer blocks.
- Number of transformer blocks (num_blks): 8. The depth of our BERT model.
- Multi-head attention heads (num_heads): 4 per block. This enables the model to focus on different parts of the input sequence of 512 tokens.
- Dropout rate: 0.1, to reduce overfitting.

And our model also supports positional encodings to give the model information about the relative position of the tokens in the sequence.

Finally, A classification layer of four fully connect layers are built on top of the BERT encoder. It takes the encoded tokens' representation as input and passes it through sequential linear layers. The dimensions transition from 512 (hidden size) to 1024, then back to 512, reducing to 256, before reaching the final binary classfication layer. For the hyperperameter, we employ a batch size of 32, and the model is trained over 50 epochs. We employ the Adam optimizer with a learning rate of 0.0001. The training process is monitored for loss and accuracy metrics, using cross-entropy loss as our loss function.

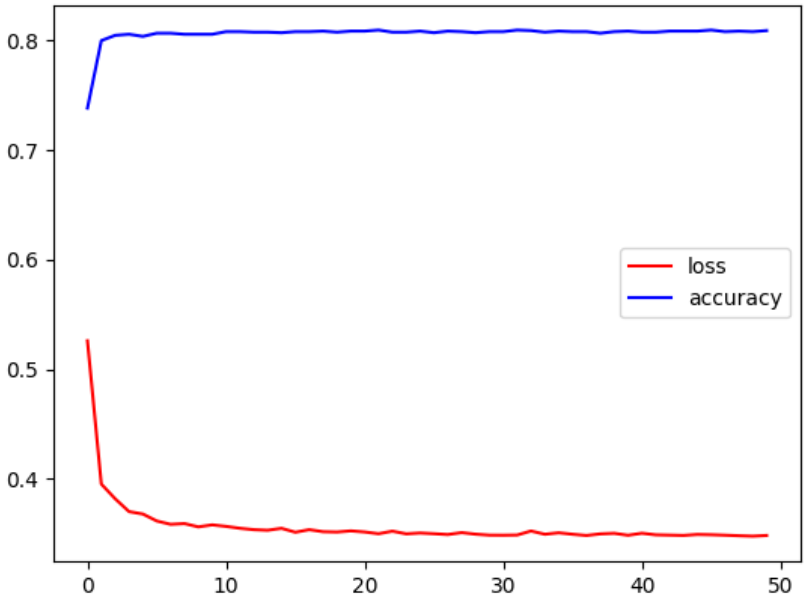

Figure 2: Loss and accuracy of BERT classification

As shown in Figure 2, our BERT model quickly raises the accuracy to over 80 percents within ten epochs and stabilizing the training loss around 0.36. However, when we put it to the test on Kaggle, the score drops to 0.612, indicating a gap between predicting training data and the general test datasets. A potential reason could be the imbalance in our training dataset, predominantly composed of essays generated by Google's PaLM LLM. Such a dataset composition might lead our model to overfit specifically to the characters of Google's PaLM-generated text. And thus, our model may not be efficient or robust when applying to more general and diverse datasets.

6 Conclusion

In this article, we implemented two methods to solve problem of detecting AI generated text apart from human generated texts, RNN and Bert. For RNN, possible improvements are changing the structure and make a prediciton per time step instead of per article and enlarge the hidden size so hidden state can store more structure information. In addition, trainable embedding would also improve the model's ability to understand and learn the difference better than fixed one-hot encoding. For Bert, the model seems over-fit the data and lack generalization capacity. Possible improvements are instead of using dot-product attention, use additive attention, and decrease the number of attention layers used. Additional drop-out layers and regulations could also be used in linear-layer classifiers.

References

[1] Alshammariyeh, M., and Alshammariyeh, M. *The Effectiveness of Software Designed to Detect AI-Generated Writing: A Case Study of ChatGPT-3.5 Essays Written by Students in a First-Year Composition Course without the Use of AI.* Open Problems in Information Systems, vol. 8, no. 1, 2022, pp. 1–28.

[2] Corizzo, R., and Leal-Arenas, S. *One-Class Learning for AI-Generated Essay Detection.* Applied Sciences, vol. 13, no. 13, 2023, pp. 7901.

[3] Crotthers, E., Japkowicz, N., and Viktor, H. *Machine Generated Text: A Comprehensive Survey of Threat Models and Detection Methods.* arXiv preprint arXiv:2210.07321, 2023.

[4] Kauranen-Kuorinen, A., et al. *Towards Possibilities & Impossibilities of AI-generated Text Detection: A Survey.* arXiv preprint arXiv:2310.15264, 2023.