SemEval-2024 Task 8: Weighted Layer Averaging RoBERTa for Black-Box Machine-Generated Text Detection

Ayan Datta[†]
IIIT Hyderabad
ayan.datta
@research.iiit.ac.in

Aryan Chandramania[†]
IIIT Hyderabad
aryan.chandramania
@research.iiit.ac.in

Radhika Mamidi
IIIT Hyderabad
radhika.mamidi@iiit.ac.in

Abstract

This document contains the details of the authors' submission to the proceedings of SemEval 2024's Task 8: Multigenerator, Multidomain, and Multilingual Black-Box Machine-Generated Text Detection Subtask A (monolingual) and B. Detection of machine-generated text is becoming an increasingly important task, with the advent of large language models (LLMs). In this paper, we lay out how using weighted averages of RoBERTa layers lets us capture information about text that is relevant to machine-generated text detection.

1 Introduction

Language modeling is a foundational task in NLP, and encompasses learning of all the features that make up language. Different levels of linguistic information are stored in language models' (LM) hidden states. This may include syntax, morphological features, phrasing, and so on. (Rogers et al., 2021) Our aim is to leverage this encoded information to help us discern machine-generated text.

The advent of large language models (LLMs) has transformed the digital landscape, and this has also led to the proliferation of machine-generated text in spaces spanning from legal proceedings, to articles, to school submissions. With this, there has been a consequential rise in the need to be able to distinguish between machine- and human-generated text across domains. Just as important is the need to be able to identify the generators for text that has been flagged as being generated by machines.

In this paper, we describe our methodology and attempts to create a system that can perform the task effectively.

2 System Overview

We have used RoBERTa-base for all experiments in the scope of this paper. The baseline set by the task organizers is reported to have been from a finetuned RoBERTa model. RoBERTa has the same architecture as BERT, but uses a byte-level BPE as a tokenizer and uses a different pretraining scheme and has become a SOTA model since its release (Liu et al., 2019).

2.1 Weighted Layer Averaging

The standard fine-tuning setup uses the [CLS] Representation of the last layer of RoBERTa. It has been shown that different layers of BERT-like models capture different levels of linguistic information, the lower layers capture lexical information and word order,the middle layers capture syntactic information, and the higher layers capture semantic and task specific information (Rogers et al., 2020). We believe that using just the last layer representation may discard some of the syntactic and lexical information, which could be crucial for the task of detecting machine generated text. We use the weighted sum of all the token representations, where each layer is assigned a corresponding weight, trained along with the downstream task, similar to ElMo (Peters et al., 2018). Let $x_0, x_1, ...x_n$ be the input sequence. Roberta generates the following hidden states.

$$\texttt{RoBERTa}([x_0, x_1, \ldots, x_n]) = H$$

Where H is a matrix consisting of hidden state vectors $\mathbf{h}_i^j$ corresponding to the j^{th} layer, and the i^{th} token. $i = 0$ represents the embedding layer

[†] These authors contributed equally to this work.

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 1623–1626
June 20-21, 2024 ©2024 Association for Computational Linguistics

output.

$$H = \begin{bmatrix} \mathbf{h}_0^0 & \mathbf{h}_1^0 & \dots & \mathbf{h}_n^0 \\ \mathbf{h}_0^1 & \mathbf{h}_1^1 & \dots & \mathbf{h}_n^1 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_0^{12} & \mathbf{h}_1^{12} & \dots & \mathbf{h}_n^{12} \end{bmatrix}$$

The standard fine-tuning setup uses $\mathbf{h}_0^{12}$ which corresponds to the [CLS] token and passes it through another Feed Forward Network to get the output class probabilities. We propose averaging all of the layer hidden states. The input $\mathbf{y}$ to the Feed Forward Network that produces the class probabilities is computed as follows.

$$\mathbf{y} = \frac{1}{12} \cdot \sum_{j=0}^{12} \frac{\lambda_j \sum_{i=0}^{n} \mathbf{h}_i^j}{n}$$

λ_j is the layer weight assigned to the layer j. $[\lambda_0, \lambda_1, ...\lambda_{12}]$ are trained along with the classification task.

2.2 Parameter Efficient Tuning with AdaLoRa

A full continual finetune of RoBERTa (and LLMs, in general) with all the weights being updated is known to potentially lead to catastrophic forgetting (Ramasesh et al., 2022), which may cause the model to become unable to generalize, with the pretraining being, for all intents and purposes, in vain.

It has also been shown that common pre-trained models have a very low intrinsic dimension; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space (Aghajanyan et al., 2020). This implies that full continual finetuning – being potentially harmful as well as unnecessary – can be replaced with a better, more parameter efficient method, which grants us more freedom with regards to model and data sizes.

Low-rank Adapters (LoRA) (Hu et al., 2021) were designed with this in mind. LoRA freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. They also offer an improvement over unfreezing just the last few layers by attaching to every layer in the model, which allows them to modify information flow at every step, starting from the source.

For our task, we made use of Adaptive LoRA (AdaLoRA) (Zhang et al., 2023), which adjusts the matrices based on parameters learned during training, i.e. the ranks of the adapters themselves are learned. Our hope is that by doing this, we prevent unnecessarily large adapters where there is not much to do, and conversely provide the flexibility to have larger matrices to handle greater amounts of information change.

3 Data

Data for the task was provided by the organizers(Wang et al., 2024a). It is an extension of the M4 Dataset (Wang et al., 2024b). The name stands for multi-generator, multi-domain, and multi-lingual corpus for machine-generated text detection. As the name suggests, the dataset has been created with text from different generators spanning multiple domains. The data for subtask A and B follow the same format, consisting of source (such as Wikipedia), model (such as Dolly), label (such as Human), and the text to be classified. The data for subtask C contains text with a combination of human- and machine-generated text, and a label indicating the word index at which the split occurs.

For our experiments, we resplit the training and dev datasets and split them uniformly across generators and domains in an 80-20 split‘. Our split of the dev set is bigger than the official dev set, to get a better estimate of our model's performance.

4 Experimental Setup

We use RoBERTa's tokenizer and trained our models for Subtask A (monolingual) (Binary Classification) and Subtask B (Multi-Class Classification) on the resplit train data and use the resplit evaluation data for early stopping. Our Hyperparameter Configuration has been specified in Appendix A.

5 Results

Our model while doing really well on our evaluation set, falls short on the test set scoring around 13 percentage points lower than the baseline for subtask A and around 1 percentage point lower than the baseline for subtask B. This could be attributed to the model not being as good in generalising to unseen domains and generators. We hypothesize more hyperparameter tuning, better aggregation of

Model	Accuracy
Ours	0.7535
Baseline	0.8846

Table 1: Results for Subtask A as computed by the organizers

Model	Accuracy
Ours	0.7387
Baseline	0.7460

Table 2: Results for Subtask B as computed by the organizers

the token representations than averaging by utilizing models like LSTMs (Hochreiter and Schmidhuber, 1997), may help the model better generalise to unseen domains and generators by being able to capture more complex features and patterns. The submission scores as computed by the task organizers have been reported in Tables 1 and 2. Scores on our Validation, the official validation and the official test set as computed by us have been reported in Tables 3 and 4.

6 Conclusion

We have demonstrated that linguistic information encoded in the various layers of large language models such as RoBERTa can be used to effectively demonstrate if a text is machine-generated or not, across different domains and generators.

References

[Aghajanyan et al.2020] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Hu et al.2021] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

[Liu et al.2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

[Peters et al.2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

[Ramasesh et al.2022] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. 2022. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.

[Rogers et al.2020] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works.

[Rogers et al.2021] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 01.

[Wang et al.2024a] Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024a. Semeval-2024 task 8: Multigenerator, multidomain, and multilingual black-box machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation*, SemEval 2024, Mexico, Mexico, June.

[Wang et al.2024b] Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. M4: Multigenerator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, Malta, March.

[Zhang et al.2023] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning.

A Hyperparameters

[†] The code used can be found in this repository: `https://github.com/advin4603/AI-Detection-With-WLA`

Dataset	Precision	Recall	Accuracy	F1 Score
Our Dev	0.9841	0.9949	0.9900	0.9895
Official Dev	0.9744	0.9444	0.9598	0.9592
Official Test	0.6823	0.9942	0.7538	0.8092

Table 3: Results for Subtask A as computed by us

Dataset	Precision$_{\text{Micro}}$	Recall$_{\text{Micro}}$	Accuracy	F1 Score$_{\text{Micro}}$
Our Dev	0.979	0.979	0.979	0.979
Official Dev	0.9783	0.9783	0.9783	0.9783
Official Test	0.7398	0.7398	0.7398	0.7398

Table 4: Results for Subtask B as computed by us

Hyperparameter	Value
Learning Rate	5e-4
Batch Size	8
Weight Decay	5e-5
Warmup Ratio	0.1
init_r	12
target_r	8
lora_alpha	200
lora_dropout	0.4

Table 5: Hyperparameters for Subtask A (Monolingual)

Hyperparameter	Value
Learning Rate	5e-4
Batch Size	8
Weight Decay	5e-5
Warmup Ratio	0.01
init_r	12
target_r	8
lora_alpha	200
lora_dropout	0.4

Table 6: Hyperparameters for Subtask B