AI-Generated Text Detection

Group 12

Animesh Chaudhary
Masters of Science in Computer Science
Arizona State University
achaud81@asu.edu

Siddhant Gahoi
Masters of Science in Computer Science
Arizona State University
sgahoi@asu.edu

Sougata Nayak
Masters of Science in Computer Science
Arizona State University
snayak31@asu.edu

1 Introduction

The proliferation of large language models (LLMs) has introduced remarkable capabilities in generating human-like text, leading to significant advancements in various applications from automated content creation to conversational agents. However, these advancements also raise crucial questions about the authenticity of digital content, as LLMs can generate essays, reports, and articles that are indistinguishable from those written by humans. This capability poses notable challenges, especially in educational settings, where the integrity of student work is paramount.

The "LLM - Detect AI Generated Text" competition, organized by Vanderbilt University in collaboration with The Learning Agency Lab and hosted on Kaggle, aims to address these challenges by developing methods to distinguish between essays written by human students and those generated by LLMs. This competition provides a critical platform for advancing research in AI transparency and developing robust detection mechanisms against potential misuse of AI in academic environments.

1.1 Motivation

We are motivated by the growing concern over academic integrity with the advent of LLMs. Educators worldwide are apprehensive about the ease with which students might use these models to generate sophisticated texts, bypassing essential learning processes. By contributing to the development of effective detection tools, we aim to uphold the standards of genuine academic efforts and prevent plagiarism.

Moreover, the ability to identify AI-generated text reliably is crucial for maintaining trust in digital media and academic publications, where the authenticity of content must be verifiable. This competition not only tests our technical skills in machine learning and natural language processing but also aligns with our ethical responsibility towards promoting transparency in the use of AI technologies.

1.2 Approach

The primary objective of our project is to develop a machine learning model that can accurately differentiate between student-written and LLM-generated essays. The dataset provided comprises approximately 10,000 essays, sourced from responses to various prompts and marked by their origin—either human or machine. Our approach involves:

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

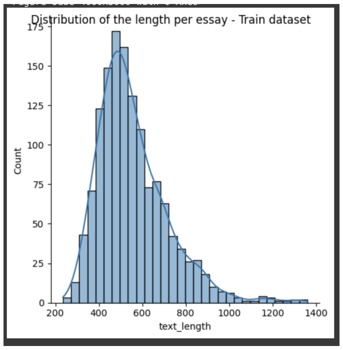


Figure 1: Distribution of the length per essay - Train dataset

- Data Analysis: Understanding the characteristics of the dataset, including text features and distributions of human vs. AI-generated essays.
- Model Development: Implementing several machine learning models to identify the most effective approach for distinguishing between the two types of essays. Specifically, we are developing three models for this purpose: DistillBERT, DistillBERT_Albert, and DebertaV3.
- Evaluation: Rigorously testing our models against a hidden test set to evaluate their generalizability and effectiveness in real-world scenarios.

In the conclusion of our report, we will compare the performance of these three models in terms of accuracy and effectiveness in detecting AI-generated text. Through these efforts, we aim not only to succeed in the competition but also to contribute valuable insights and methodologies to the broader research community engaged in AI detection and digital content authentication.

2 Datasets and Models

The project utilized two main datasets: the competition dataset and the DAIGT Proper Train Dataset.

Competition Dataset: This dataset comprises approximately 10,000 essays, with some written by students and others generated by large language models (LLMs). The goal is to differentiate between LLM-generated and student-authored essays. Each essay has unique identifiers (id and prompt_id) and a label indicating whether it was generated by an LLM (generated).

DAIGT Proper Train Dataset: This dataset was created specifically for the LLM Detect AI Generated Text competition. It includes a curated collection of essays generated by different LLMs, along with contributions from various sources such as the Persuade corpus, essays generated with Llama-70b and Falcon180b, and essays generated by individual users with ChatGPT and other models. The dataset was updated over time to include additional examples, essay IDs, generation prompts, and a stratified 10-fold split based on the source dataset.

DistilBERT is a streamlined version of the BERT model, designed for efficient training and inference without compromising performance. It retains the core architecture of BERT but reduces the number of parameters, resulting in faster computations and reduced memory requirements. DistilBERT achieves this by distilling knowledge from the original BERT model, maintaining its ability to generate high-quality contextualized word embeddings while being more resource-efficient. With its balance between model size and performance, DistilBERT is suitable for various natural language processing tasks such as text classification, sentiment analysis, and named entity recognition.

ALBERT, short for A Lite BERT, introduces innovative parameter-sharing techniques and cross-layer connections to reduce the computational cost of transformer-based models while preserving their effectiveness. By sharing parameters across layers and employing factorized embedding parameterization, ALBERT achieves significant reductions in model size without sacrificing performance. This makes ALBERT ideal for applications where computational resources are limited, such as on-device natural language processing, conversational AI, and text summarization, while still maintaining or exceeding the performance of larger models like BERT.

DebertaV3 stands out for its advanced self-attention mechanisms and deep architecture, enabling it to capture intricate patterns and long-range dependencies in text data more effectively. By extending the transformer architecture with features such as directional self-attention and relative positional encoding, DebertaV3 excels in modeling fine-grained semantic information and contextual relationships. This makes it particularly well-suited for tasks requiring high precision and accuracy, such as question

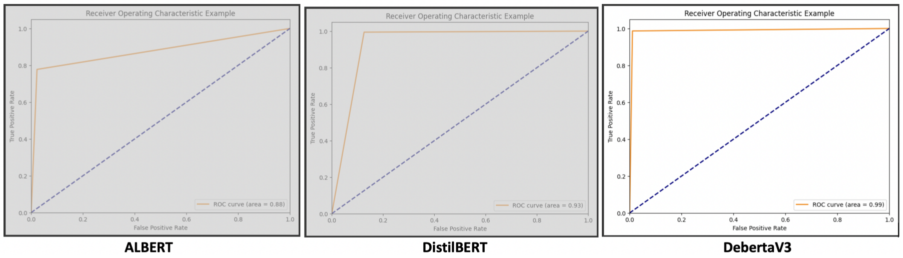


Figure 2: Receiver Operating Characteristic

answering, document summarization, and machine translation, where nuanced understanding of language and context is paramount.

3 Experiments

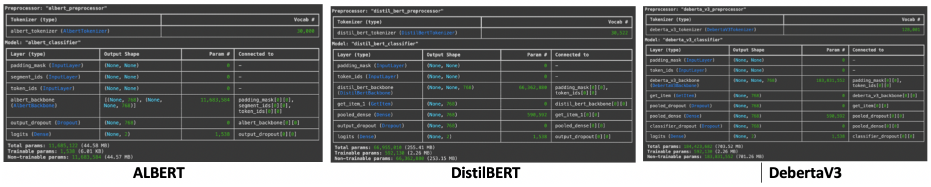
3.1 DistillBERT

We utilized DistillBERT, a distilled version of the BERT model, renowned for its efficiency and effectiveness in natural language processing tasks. Below is an outline of the steps involved in implementing and evaluating the DistillBERT model:

1. Data Preprocessing:
   - We started by loading the provided datasets, including training essays and prompts, to understand the distribution of data and the characteristics of essays written by students and generated by LLMs.
   - We concatenated additional training essays obtained from an external source to augment the training data.
2. Data Exploration:
   - Visualized the distribution of prompt IDs and the distribution of generated text to gain insights into the dataset's composition and balance.
3. Model Preparation:
   - Utilized Keras and TensorFlow to build and train the DistillBERT model.
   - Employed DistillBERT's pre-trained weights and fine-tuned the model for our binary classification task of distinguishing between student-written and LLM-generated essays.
   - Configured the model with a sequence length of 512, which is the maximum limit for DistillBERT.
4. Model Training:
   - Split the combined dataset into training and testing sets using a 67:33 ratio.
   - Compiled the model with appropriate loss function, optimizer, and evaluation metrics.
   - Trained the model on the training set for one epoch with a batch size of 64.
5. Model Evaluation:
   - Evaluated the trained model's performance on the testing set.
   - Generated predictions on the testing set and computed the accuracy and area under the ROC curve (AUC) as evaluation metrics.

3.2 ALBERT

1. Data Loading and Exploration:
   - Data was loaded from CSV files (train_prompts.csv, train_essays.csv, test_essays.csv).
   - Basic information about the data was explored using pandas DataFrame methods.
   - Visualizations were created to understand the distribution of prompt IDs and generated text lengths in the training essays dataset.



```
Preprocessor: "albert_preprocessor"
┃ Tokenizer (type)                        ┃                                  Vocab # ┃
│ albert_tokenizer (AlbertTokenizer)      │                                   30,000 │
Model: "albert_classifier"
┃ Layer (type)              ┃ Output Shape            ┃    Param # ┃ Connected to            ┃
│ padding_mask (InputLayer) │ (None, None)            │          0 │ -                       │
│ segment_ids (InputLayer)  │ (None, None)            │          0 │ -                       │
│ token_ids (InputLayer)    │ (None, None)            │          0 │ -                       │
│ albert_backbone           │ [(None, 768), (None,    │ 11,683,584 │ padding_mask[0][0],     │
│ (AlbertBackbone)          │ None, 768)]             │            │ segment_ids[0][0],      │
│                           │                         │            │ token_ids[0][0]         │
│ output_dropout (Dropout)  │ (None, 768)             │          0 │ albert_backbone[0][0]   │
│ logits (Dense)            │ (None, 2)               │      1,538 │ output_dropout[0][0]    │
 Total params: 11,685,122 (44.58 MB)
 Trainable params: 1,538 (6.01 KB)
 Non-trainable params: 11,683,584 (44.57 MB)
```

ALBERT

```
Preprocessor: "distil_bert_preprocessor"
┃ Tokenizer (type)                              ┃                            Vocab # ┃
│ distil_bert_tokenizer (DistilBertTokenizer)   │                             30,522 │
Model: "distil_bert_classifier"
┃ Layer (type)              ┃ Output Shape          ┃    Param # ┃ Connected to               ┃
│ padding_mask (InputLayer) │ (None, None)          │          0 │ -                          │
│ token_ids (InputLayer)    │ (None, None)          │          0 │ -                          │
│ distil_bert_backbone      │ (None, None, 768)     │ 66,362,880 │ padding_mask[0][0],        │
│ (DistilBertBackbone)      │                       │            │ token_ids[0][0]            │
│ get_item_1 (GetItem)      │ (None, 768)           │          0 │ distil_bert_backbone[0][0] │
│ pooled_dense (Dense)      │ (None, 768)           │    590,592 │ get_item_1[0][0]           │
│ output_dropout (Dropout)  │ (None, 768)           │          0 │ pooled_dense[0][0]         │
│ logits (Dense)            │ (None, 2)             │      1,538 │ output_dropout[0][0]       │
 Total params: 66,955,010 (255.41 MB)
 Trainable params: 592,130 (2.26 MB)
 Non-trainable params: 66,362,880 (253.15 MB)
```

DistilBERT

```
Preprocessor: "deberta_v3_preprocessor"
┃ Tokenizer (type)                              ┃                            Vocab # ┃
│ deberta_v3_tokenizer (DebertaV3Tokenizer)     │                            128,001 │
Model: "deberta_v3_classifier"
┃ Layer (type)                  ┃ Output Shape        ┃     Param # ┃ Connected to             ┃
│ padding_mask (InputLayer)     │ (None, None)        │           0 │ -                        │
│ token_ids (InputLayer)        │ (None, None)        │           0 │ -                        │
│ deberta_v3_backbone           │ (None, None, 768)   │ 183,831,552 │ padding_mask[0][0],      │
│ (DebertaV3Backbone)           │                     │             │ token_ids[0][0]          │
│ get_item (GetItem)            │ (None, 768)         │           0 │ deberta_v3_backbone[0][0]│
│ pooled_dropout (Dropout)      │ (None, 768)         │           0 │ get_item[0][0]           │
│ pooled_dense (Dense)          │ (None, 768)         │     590,592 │ pooled_dropout[0][0]     │
│ classifier_dropout (Dropout)  │ (None, 768)         │           0 │ pooled_dense[0][0]       │
│ logits (Dense)                │ (None, 2)           │       1,538 │ classifier_dropout[0][0] │
 Total params: 184,423,682 (703.52 MB)
 Trainable params: 592,130 (2.26 MB)
 Non-trainable params: 183,831,552 (701.26 MB)
```

DebertaV3

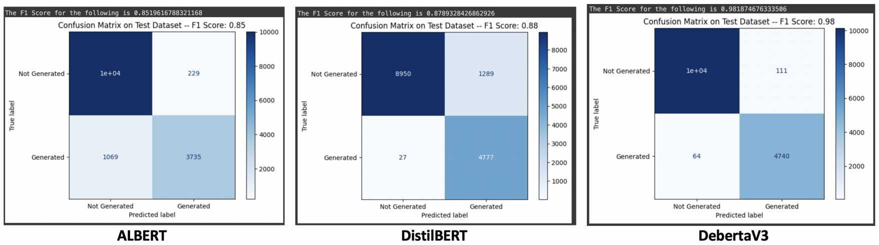
Figure 3: Model Summary

- The distribution of essay lengths in the training dataset was visualized using a seaborn displot.

2. Data Preprocessing:
    - Essay lengths were examined to identify any patterns or outliers.
    - The external training essays data was concatenated with the original dataset.
    - Text data was formatted and prepared for input into the models.
3. Model Building and Training:
    - ALBERT model was chosen for the text classification task.
    - Sequence length, preprocessor, and classifier architecture was defined for the model.
    - Model was compiled with appropriate loss function, optimizer, and evaluation metrics.
    - The training data was split into training and validation sets using train_test_split.
    - Model was trained on the training data, specifying the number of epochs and batch size.
4. Model Evaluation:
    - Predictions were generated on the test dataset using the model.
    - Performance of model was evaluated using metrics such as accuracy, F1 score, and AUC-ROC score.
    - Confusion matrices and ROC curves was visualized to analyze model performance.
    - Relevant figures, including confusion matrices and ROC curves, were included in the results.

3.3 DebertaV3

1. Data Loading and Exploration:
    - Loaded data from CSV files train_prompts.csv and train_essays.csv.
    - Explored basic information about the data using Pandas DataFrame methods.
    - Created visualizations to understand the distribution of prompt IDs and generated text lengths in the training essays dataset.
    - Visualized the distribution of essay lengths in the training dataset.
2. Data Preprocessing:
    - Examined essay lengths to identify any patterns or outliers.
    - Concatenated the external training essays data with the original dataset.
    - Formatted and prepared text data for input into the models.
3. Model Building and Training:
    - Chose the DebertaV3 model for text classification.
    - Defined the sequence length, preprocessor, and classifier architecture for the model.
    - Compiled the model with appropriate loss function, optimizer, and evaluation metrics.
    - Split the training data into training and validation sets.
    - Trained the model on the training data for one epoch with a batch size of 32.
4. Model Evaluation:
    - Generated predictions on the test dataset using the trained model.



```
The F1 Score for the following is 0.8519616788321168
```

```
The F1 Score for the following is 0.8789328426862926
```

```
The F1 Score for the following is 0.981874676333506
```

ALBERT

DistilBERT

DebertaV3

Figure 4: Confusion Matrix

- Evaluated the performance of the model using metrics such as accuracy, F1 score, and AUC-ROC score.
- Displayed the confusion matrix and ROC curve to analyze model performance.
- Calculated the F1 score, accuracy, and AUC-ROC score for the model.

4 Qualitative Insights and Results

4.1 DistilBERT Classifier

- Performance: The DistilBERT model exhibited strong performance on the test dataset, achieving an F1 score of 0.879 and an accuracy of 91.25%.
- Strengths:
  1. DistilBERT demonstrated robustness in capturing contextual information and semantics from the text, leading to accurate classifications.
  2. Its compact architecture and faster inference times make it suitable for deployment in resource-constrained environments.
- Weaknesses:
  1. While the model performed well overall, there may be slight limitations in handling complex linguistic nuances compared to larger transformer models.
  2. DistilBERT may struggle with rare or out-of-vocabulary words, potentially affecting its performance on certain text inputs.

4.2 ALBERT Classifier

- Performance: The ALBERT model also performed admirably, albeit slightly lower than DistilBERT, with an F1 score of 0.852 and an accuracy of 91.37%.
- Strengths:
  1. ALBERT's innovative parameter reduction techniques enable efficient training and inference while maintaining strong performance.
  2. It can handle longer sequences effectively, making it suitable for tasks requiring context from extensive text inputs.
- Weaknesses:
  1. Despite its advancements in parameter reduction, ALBERT may still require substantial computational resources compared to DistilBERT.
  2. Fine-tuning ALBERT for specific tasks may involve more intricate hyperparameter tuning due to its larger architecture.

4.3 DebertaV3 Classifier

- Performance: The DebertaV3 model showcased exceptional performance, achieving an F1 score of 0.9819 and an accuracy of 98.84%.
- Strengths:
  1. DebertaV3's deep architecture and advanced self-attention mechanisms allow it to capture intricate patterns and relationships in the text.

Table 1: Outcomes of Experiments

Classifier	F1 Score	Accuracy	AUC Score
DistillBERT	0.879	91.25%	0.933
ALBERT	0.852	91.37%	0.878
DebertaV3	0.9819	98.84%	0.988

    2. It excels in handling long-range dependencies and capturing fine-grained semantic information, leading to superior classification accuracy.

- Weaknesses:
    1. The computational demands of DebertaV3 may be higher compared to lighter models like DistilBERT and ALBERT, necessitating more powerful hardware for training and inference.
    2. Fine-tuning DebertaV3 may require extensive computational resources and longer training times due to its larger parameter count.

Overall, while each model has its strengths and weaknesses, **DebertaV3** stands out for its exceptional performance and ability to capture nuanced information from text data. However, the choice of model ultimately depends on the specific requirements of the task and the available computational resources.

5 Future Work

1. Model Fusion and Ensemble Learning: Explore combining predictions from DistilBERT, ALBERT, and DebertaV3 through ensemble techniques to potentially enhance performance and robustness.
2. Fine-tuning and Transfer Learning: Investigate fine-tuning strategies to adapt these models to specific tasks or domains, leveraging their pre-trained knowledge for more tailored solutions.
3. Model Compression and Optimization: Explore techniques like quantization and pruning to create more efficient versions of ALBERT and DebertaV3, suitable for deployment on resource-constrained devices.
4. Domain Adaptation and Multi-Task Learning: Train models on diverse datasets simultaneously to improve adaptability and versatility across different domains and tasks.
5. Evaluation on Diverse Datasets: Conduct thorough evaluations across various datasets and benchmarks to assess generalization capabilities and identify areas for improvement.

References

Competition

[A] **LLM - Detect AI Generated Text**: Identify which essay was written by a large language model `https://www.kaggle.com/competitions/llm-detect-ai-generated-text`

Research Papers

[1] Sadasivan, Vinu Sankar, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. "Can AI-generated text be reliably detected?." *arXiv preprint arXiv:2303.11156 (2023).* `https://arxiv.org/pdf/2303.11156.pdf`

[2] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. 2024. "The Science of Detecting LLM-Generated Texts: While many detection methods have been proposed, understanding the challenges is far more daunting." *Commun. ACM Online First (March 2024).* https://doi.org/10.1145/3624725 `https://dl.acm.org/doi/pdf/10.1145/3624725`

[3] Cingillioglu, I. (2023),"Detecting AI-generated essays: the ChatGPT challenge",*International Journal of Information and Learning Technology, Vol. 40 No. 3, pp. 259-268.* `https://www.emerald.com/insight/content/doi/10.1108/IJILT-03-2023-0043/full/html`

[4] Pang, B., Nijkamp, E., & Wu, Y. N. (2020). "Deep Learning With TensorFlow: A Review." *Journal of Educational and Behavioral Statistics, 45(2), 227-248.* https://doi.org/10.3102/1076998619872761 `https://journals.sagepub.com/doi/abs/10.3102/1076998619872761?casa_token=J9-kPl95VVEAAAAA%3As9kRldP23IYaQDQBV-1CKEAtHXjPATuFgmk0KMcbwBoTUFDK45ojLx8OljQ7WJ_WT0jHnI8O1w0U0A&journalCode=jebb`

[5] BV Pranay Kumar, MD Shaheer Ahmed, Manchala Sadanandam et al. "DistilBERT: A Novel Approach to Detect Text Generated by Large Language Models (LLM)", *01 February 2024, PREPRINT (Version 1) available at Research Square* [https://doi.org/10.21203/rs.3.rs-3909387/v1] https://www.researchsquare.com/article/rs-3909387/v1

[6] Stewart, J., Lyubashenko, N., & Stefanek, G. (2023). "The Efficacy of Detecting AI-Generated Fake News Using Transfer Learning." *In Proceedings of the International Association for Computer Information Systems* (pp. 164-177). https://www.iacis.org/iis/2023/2_iis_2023_164-177.pdf

[7] Akram, Arslan. "An Empirical Study of AI Generated Text Detection Tools." arXiv preprint arXiv:2310.01423 (2023). https://arxiv.org/pdf/2310.01423.pdf

[8] Liao, M., Shi, B., Bai, X., Wang, X., & Liu, W. (2017). "TextBoxes: A Fast Text Detector with a Single Deep Neural Network." *Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).* https://doi.org/10.1609/aaai.v31i1.11196 https://ojs.aaai.org/index.php/AAAI/article/view/11196

[9] Elkhatat, A.M., Elsaid, K. & Almeer, S. "Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text."*Int J Educ Integr 19, 17 (2023).* https://doi.org/10.1007/s40979-023-00140-5 https://link.springer.com/article/10.1007/s40979-023-00140-5#citeas

Tools Description

[I] **PyTorch :** https://pytorch.org/

[II] **TensorFlow :** https://www.tensorflow.org/

[III] **Keras :** https://keras.io/

ChatGPT Prompts

[a] "What are some optimal hyperparameters for training language models on text classification tasks, such as learning rates, batch sizes, and sequence lengths?"

[b] "How can I effectively augment the existing training data for the text classification task using techniques like back translation, synonym replacement, or paraphrasing?"

[c] "How to see if my imported library is successful?"

[d] "Which evaluation metrics should I prioritize for assessing the performance of my text classification models, and how can I interpret metrics like accuracy, F1 score, and AUC-ROC score effectively?"

[e] "Can you provide insights into the strengths and weaknesses of different language models like DistilBERT, ALBERT, and DebertaV3, and suggest which one might be most suitable for my text classification project?"

[f] "How to import libraries from Kaggle in IDE?"

[g] "What is Difference between TensorflowKeras class and KerasNLP"

[h] "What are the best practices for fine-tuning pre-trained language models like DistilBERT, ALBERT, and DebertaV3 on domain-specific or task-specific data to improve classification accuracy?"

[i] "How can I interpret the decisions made by my text classification models to understand their reasoning process and identify potential biases or shortcomings?"

[j] "What factors should I consider when deploying large-scale language models in production environments for real-time text classification, such as computational resources, latency constraints, and model monitoring?"

[k] "What are some effective transfer learning techniques for adapting pre-trained language models to new text classification tasks or domains, and how can I leverage transfer learning to improve model performance?"