Know your enemy and know yourself

Detecting AI generated text with source model prediction

Bo-Yi Mao
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
Stu. ID 111062209

Kenneth Chandra
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
Stu. ID 110006161

Aurick Daniel F. Setiadi
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
Stu. ID 111000177

Yi-Hsueh Chu
Dept. of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
Stu. ID 111062332

I. Introduction

With the recent boom of generative models and large language models (LLMs), their potential has enabled plagiarism in academic writing. The Detect AI Generated Text (DAIGT) competition hosted on Kaggle [1] aims to identify and discriminate AI-generated essays to ensure fairness in the usage of LLMs.

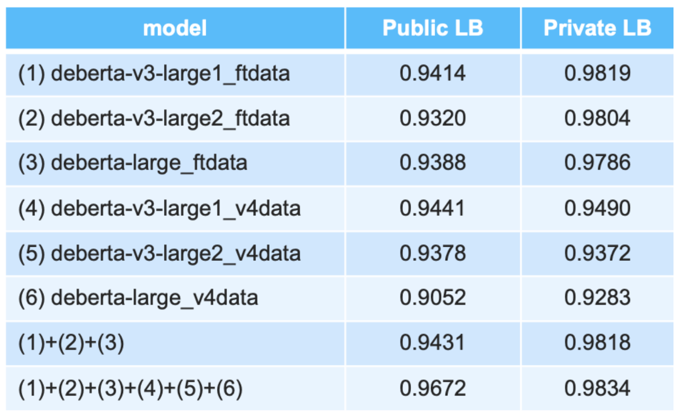
Previous works mostly focus on improving training data quality and diversity and using larger and larger models. However, few have attempted to address the distribution difference of different language models. In this paper, we present our approach to detecting AI-generated text by predicting the source model used to generate the text. We explore various machine learning techniques and natural language processing (NLP) methods to achieve this goal. Our contributions include a comprehensive dataset of AI-generated and human-written texts, a detailed analysis of different models' performance, and insights into the features that distinguish AI-generated text from human-written text.

II. Previous Works

Transformer-based classification models often struggle to generalize from the PERSUADE corpus dataset to the private test dataset, frequently tending towards overfitting. As a result, many attempts have resulted in scores around 0.75, despite performing well on local validation. Consequently, many have found other approaches to tackling the problem.

One notable success from Bamba et al. [2] involved increasing both the diversity and size of the training dataset. Yhey achieved a significant improvement in accuracy by leveraging a diverse and extensive dataset. They utilized a combination of AI-generated and human-written texts, which helped in training robust models capable of distinguishing between the two, eventually winning the competition in first place, with a score of 0.988.

Xu [3] proved that ensemble approach could lead to higher accuracy score. By training 6 models with each having their unique combination of dataset and model type (DeBERTa v3 large or DeBERTa large), Xu obtained the prediction from each model and averaged them to get the final prediction. This approach is able to achieve a higher accuracy score than each model individually, as shown in Figure 1.



model	Public LB	Private LB
(1) deberta-v3-large1_ftdata	0.9414	0.9819
(2) deberta-v3-large2_ftdata	0.9320	0.9804
(3) deberta-large_ftdata	0.9388	0.9786
(4) deberta-v3-large1_v4data	0.9441	0.9490
(5) deberta-v3-large2_v4data	0.9378	0.9372
(6) deberta-large_v4data	0.9052	0.9283
(1)+(2)+(3)	0.9431	0.9818
(1)+(2)+(3)+(4)+(5)+(6)	0.9672	0.9834

Fig. 1. Comparison of different methods used by Xu [3] and their impact on the leaderboard accuracy score.

Xu also demonstrated the importance of pretraining, fine-tuning, and larger models. A model pretrained on a larger dataset and fine-tuned on a smaller dataset could achieve a higher accuracy score than only fine-tuning or pretraining, as shown in Figure 2. Larger models also tend to have a higher accuracy score than smaller models, as shown in Figure 3.

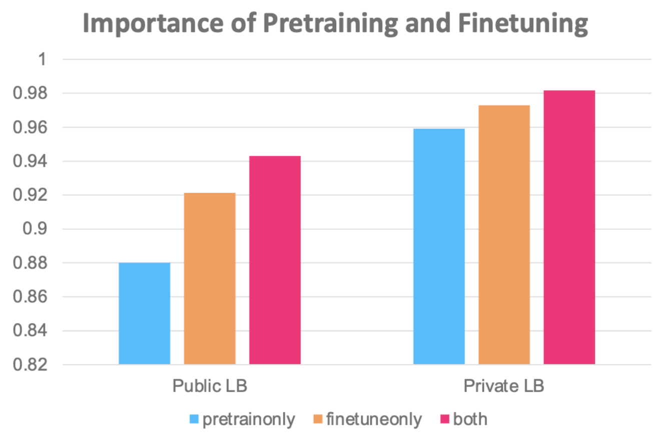


Fig. 2. Comparison of different training methods used by Xu [3] and their impact on the leaderboard accuracy score.

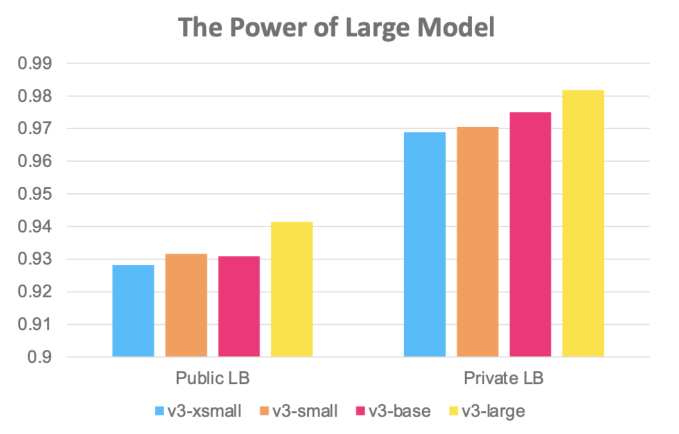


Fig. 3. Comparison of different model sizes used by Xu [3] and their impact on the leaderboard accuracy score.

Another work from Day [4] used 500,000 documents from The Pile and another 500,000 plausible continuations generated with local LLMs, and resulted in a sizeable increase in accuracy; in fact, they submitted a DeBERTa-v3 LLM at a context length of 1024 tokens with scores of 0.96 for individual models and 0.963 for an ensemble. Compared to the models trained on the PERSUADE corpus, which scored around 0.75, this is a significant improvement.

However, Cozzolino [5] took a different approach by using synthetic features such as entropy, surprisal, and average word length. By merely focusing on these features, Cozzolino was able to achieve a competitive score in the DAIGT competition. As shown in Figure 4, his method demonstrated that even without relying heavily on large datasets or complex models, it is possible to achieve good performance by carefully selecting and engineering features that capture the nuances of AI-generated text.

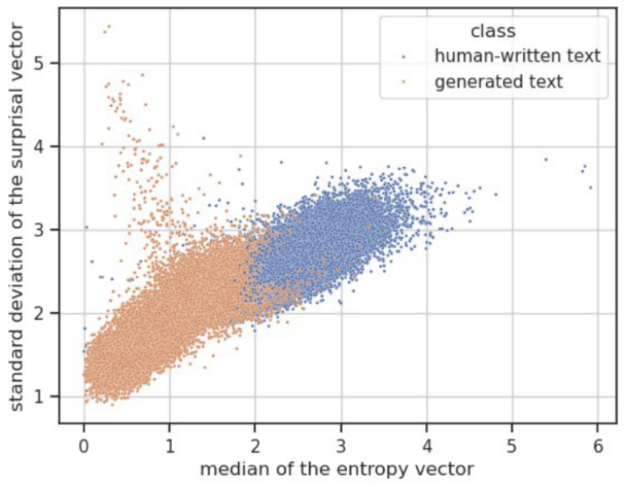


Fig. 4. Engineered features used by Davide Cozzolino [5] separates AI-generated text from human-written text well.

III. Methodology

The importance of data preparation and preprocessing cannot be overstated. Utilizing raw data provided by the competition organizers without any preprocessing resulted in a plain DeBERTa model achieving only a 0.66 private score and a 0.52 public score on the leaderboard. This demonstrates that even powerful models like DeBERTa can underperform if the features are not effectively extracted and utilized.

Our methodology overall consists of three steps. First, the data is prepared and preprocessed ahead of time, published to Kaggle, and ready to be used. Second, certain representation learning models perform feature extraction on raw data. Finally, binary/multi-class classification is performed via various machine learning methods.

A. Data Preparation

We collected training data from several sources [6]–[10], containing 87k AI-generated and 13k human-written essays, with prompts from PERSUADE, WECCL, TOEFL, and GRE corpus. The generated essays are generated from various models, including T5, LLaMA series, GPT series, Mistral7B, Claude, Falcon, T5, PaLM, and Cohere. The class distributions (AI-generated vs. human-written) are shown in Figure 5, and the distribution of source models is shown in Figure 6.

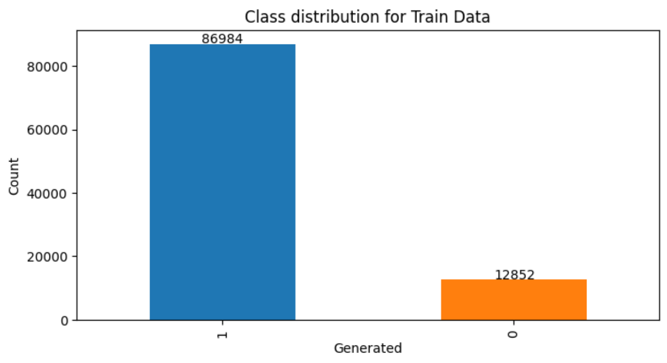


Fig. 5. Class distribution of AI-generated and human-written essays

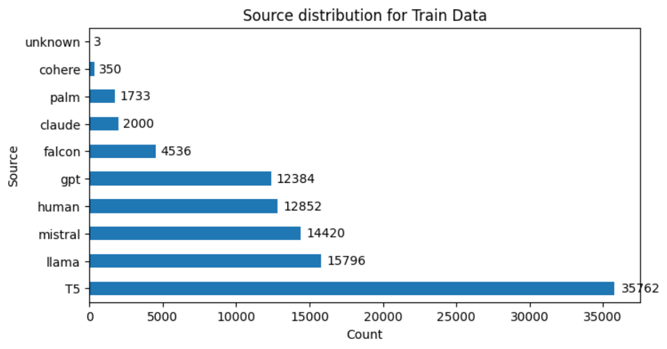


Fig. 6. Distribution of source models

B. Text Tokenization

1) Byte Pair Encoding: Byte Pair Encoding (BPE) is a subword tokenization algorithm that is widely used in NLP tasks. It works by iteratively merging the most frequent pair of bytes (or characters) in a given text corpus. Initially, every character in the text is treated as a separate token. The algorithm then finds the most frequent pair of adjacent tokens and merges them into a single token. This process is repeated until a predefined vocabulary size is reached or no more pairs can be merged.

The main advantage of BPE is that it can effectively handle rare and out-of-vocabulary words by breaking them down into smaller, more frequent subword units. This allows models to represent and generate words that were not seen during training, improving their ability to generalize to new text.

2) AutoTokenizer: AutoTokenizer is a convenience tool provided by Hugging Face. It can automatically select a tokenizer which fits the model the most. Besides, it also provides a uniformed API, so that we can use different tokenizers with the same code. For example, we use AutoTokenizer in the DeBERTa model structure by calling: `AutoTokenizer.from_pretrained("microsoft/deberta-v3-base")`.

C. Feature Extraction

1) TF-IDF: Term Frequency-Inverse Document Frequency (TF-IDF) is a popular technique for document vectorization, which is proven to be useful in various text classification tasks. The TF-IDF vector is calculated as follows:

$$\text{TF}(t,d) = \frac{n_{t,d}}{\sum_{t' \in d} n_{t',d}} \tag{1}$$

$$\text{IDF}(t) = \log \frac{|D|}{|d \in D : t \in d|} \tag{2}$$

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \tag{3}$$

where D is the set of all documents, $n_{t,d}$ is the number of occurrences of term t in document d, $\text{TF}(t,d)$ is the term frequency of term t in document d, and $\text{IDF}(t)$ is the inverse document frequency of term t in the corpus. The TF-IDF vector is then normalized to have unit norm.

We use TF-IDF to extract features from the text data, which are then used as input to the classifier. The TF-IDF vectorizer is trained on the training data and used to transform both the training and test data.

2) Entropy-based Features: Huang et al. [11] proposed GPT-based Arithmetic Coding (GPT-AC) to encode text data with an entropy-based compressor that involves the GPT model. Drawing inspiration from this work, Cozzolino [5] incorporated the following features into the classifier:

- **Entropy:** The expected value of the information contained in each token, indicating the randomness of the text.
- **Surprisal:** The negative log-likelihood of each token, indicating the unexpectedness of the text.
- **Average word length:** The average length of the words in the text data, indicating the complexity of the text.

To use the entropy-based features, Cozzolino utilized the statistical properties of the data and used synthetic-based entropy including median, standard deviation, mean, and quantiles. The entropy itself is computed through a pre-trained LLM Phi-2 under Microsoft. These selection of his features extraction are his best result which could achieve the best score accuracy using DAIGT dataset. However, the DAIGT dataset's purpose is only to measure the features performance, as for the training he will not use the DAIGT dataset. These features were then used to train a SVM classifier using the organizer dataset, which achieved a competitive score in the DAIGT competition.

3) LLM Embedding: LLMs are known for their ability to generate high-quality text, but they can also be used to extract features from text data. We fine-tuned both Mistral7B and DeBERTa using our dataset, allowing it to react to subtle stylistic and structural cues common in synthetic text.

Mistral7B is a 7-billion-parameter language model engineered by the Mistral AI team. Our method makes use of the model's ability to analyse textual properties and discover patterns that distinguish human-written content from machine-generated language [12].

DeBERTa is a state-of-the-art transformer-based model developed by Microsoft. It introduces two key innovations, which are Disentangled Attention Mechanism and Enhanced Masked Encoding. The former mechanism enables the model to focus more efficiently on semantic relationships without being overly constrained by word order. The latter skill improves its ability to model dependencies across long sequences and better capture the contextual nuances of text [13].

After we get the output from LLMs, a mean pooling is implemented to extract a general feature. Filtering out the valid tokens with `attention_mask`, those tokens are summed and divided with the number of valid tokens to get the mean embedding. At last, passing over the `mlp_head`, a sequence of fully-connected layers, will it be transformed into a feature vector.

D. Classification

1) Traditional Machine Learning: Drawn from the top solutions in the competition, we experimented with an ensemble of classifiers, including linear classifiers, multinomial naive Bayes, and LightGBM. The hyperparameters are the same as the original solution. The final prediction is made by weighted voting of the three classifiers:

$$\hat{y} = 0.1 \times \hat{y}_{\text{Bayes}} + 0.45 \times \hat{y}_{\text{linear}} + 0.45 \times \hat{y}_{\text{LGBM}} \tag{4}$$

To experiment with our hypothesis for source model prediction, we set up two classification tasks: binary classification (AI-generated or human-written) and multi-class classification (source model prediction). For binary classification, each classifier is trained to distinguish between AI-generated and human-written essays. For multi-class classification, each classifier is trained to predict the source model of the AI-generated essays.

Specially for multi-class classification, we also experimented with different class weights to balance the class distribution. The vanilla class weights are calculated as the inverse of the class frequency in the training data, and the hyperparameter `human_factor` is multiplied to the class weights of the human-written class. The class weights are then used in the training of the classifier.

2) Deep Learning: With the embedding vector generated as described in Section III-C3, we can use a deep learning model to classify the text. We experimented with simple feed-forward neural networks, getting promising results. The model is trained for 5 epochs and evaluated on the validation set with the ROC-AUC score.

3) Ensemble: The ensemble approach involved a weighted averaging of probabilities generated by individual classifiers. This allowed the models to compensate for each other's weaknesses and collectively improve accuracy. Beside using different model types, different training data can also be used to train the model. As shown by Xu [3], this approach can lead to a higher accuracy score than each model individually.

IV. Experiments

A. *Effect of Source Model Prediction*

We hypothesized that predicting the source model of the AI-generated text could help distinguish between AI-generated and human-written text. To test this hypothesis, we used BPE tokenization, TF-IDF feature extraction, and the ensemble classifier described in Section III-D1 to predict the source model of the AI-generated essays. The results are shown in Table I and plotted in Figure 7, with the dotted line indicating the baseline accuracy of binary classification.

As shown, the accuracy of multi-class classification is generally higher than that of binary classification. This suggests that predicting the source model of the AI-generated text does help distinguish between AI-generated and human-written text. The best accuracy is achieved with `human_factor`$=3\times10^4$, being about 20% higher than the baseline accuracy of binary classification.

Note that the accuracy for public and private scores differs significantly and consistently across different class weights. This suggests that the distribution of source models in the public and private test data is different, which may affect the generalization of the classifier. As stated in many previous works, the importance of training data diversity and quality is crucial for the generalization of the model, and our observation is consistent with this finding.

B. *Submission Results*

As the DAIGT competition is a code competition, we have to submit our code to Kaggle for evaluation. We have tested various models and hyperparameters, with some of the best results shown in Table II. The best model we have found is the one modified from Xu's [3] solution, which achieved a **private score of 0.983412** and a **public score of 0.967298**. This model is an ensemble of DeBERTa models pretrained and fine-tuned on the DAIGT dataset.

TABLE I
EFFECT OF CLASS WEIGHTS ON MULTI-CLASS CLASSIFICATION ACCURACY

`human_factor`	Private Score	Public Score
0	0.770701	0.920051
1×10^{-3}	0.760378	0.919360
1×10^{-2}	0.736073	0.897525
3×10^{-2}	0.709871	0.880807
1×10^{-1}	0.701310	0.873958
5×10^{-1}	0.706552	0.883646
1×10^{0}	0.697281	0.881103
5×10^{0}	0.647095	0.866519
1×10^{1}	0.636825	0.862602
3×10^{1}	0.638540	0.872659
5×10^{1}	0.652778	0.876469
1×10^{2}	0.681232	0.884198
3×10^{2}	0.704799	0.890483
1×10^{3}	0.724823	0.902500
3×10^{3}	0.752475	0.910431
1×10^{4}	0.777595	0.915113
3×10^{4}	0.807771	0.912862
1×10^{5}	0.782545	0.898491
3×10^{5}	0.777071	0.889535
1×10^{6}	0.767274	0.887141
3×10^{6}	0.748232	0.872659
1×10^{7}	0.734994	0.864930
Baseline: binary classification	0.665908	0.891065

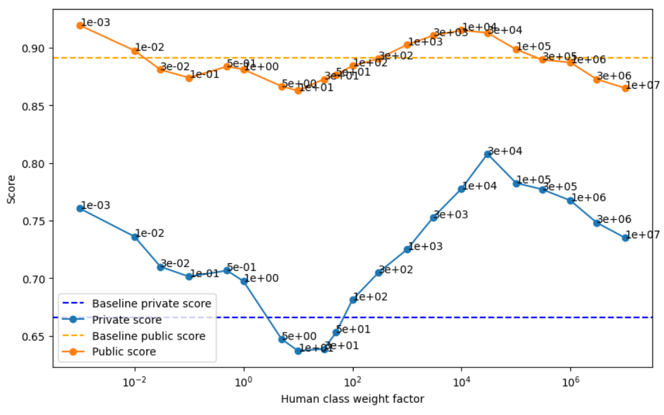


Fig. 7. Effect of class weights on multi-class classification accuracy

It is worth noting that training the Mistral7B v0.1 model presented additional challenges due to its complexity and resource requirements. Specifically, we were unable to fully utilize its potential due to the high computation constraints and lack of available resources. For instance, the model was trained with `max_steps` set to 800 steps. Despite such constraints, Mistral's model still achieved an accuracy of 87% and 79% for public and private scores respectively.

V. Conclusion

Our work demonstrates the importance of source model prediction in detecting AI-generated text. By predicting the source model of the AI-generated text, we can outperform the baseline accuracy of binary classification. This suggests that the distribution of source models is an important factor in distinguishing between AI-generated and human-written text.

TABLE II
TESTED MODELS AND DATASETS, AND THEIR RESPECTIVE ACCURACIES

Training Set	Tokenizer	Feature Extractor	Classifier	Hyperparameters	Private Score	Public Score
Ours	BPE	TF-IDF	Multi-class ensemble	`human_factor`=3×10^4	0.807771	0.912862
Ours	BPE	TF-IDF	Multi-class ensemble	`human_factor`=1×10^5	0.782545	0.898491
Ours	BPE	TF-IDF	Multi-class ensemble	`human_factor`=0	0.770701	0.920051
Ours	BPE	TF-IDF	Multi-class ensemble	`human_factor`=1×10^{-3}	0.760378	0.919360
Ours	BPE	TF-IDF	Binary ensemble	`tokenizer_lower=False`	0.665908	0.891152
Ours	BPE	TF-IDF	Binary ensemble	`tokenizer_lower=True`	0.658713	0.894109
Ours	Mistral	Mistral7b	Mistral7b	-	0.702772	0.854477
Base	Mistral	Mistral7b	Mistral7b	-	0.528327	0.530082
Base + DAIGT	DeBERTa	DeBERTa	DeBERTa 3 Large x4 + DeBERTa Large x2	Second rank pretrained	0.983412	0.967298
Base + DAIGT	Phi-2	extracting entropy	SVM	RBF kernel, γ = 0.01, ν = 0.01	0.946589	0.891023
Base + DAIGT	Phi-2	extracting entropy	SVM	RBF kernel, γ = 0.01, ν = 0.05	0.906441	0.892408

Classifier	Description
Multi-class ensemble	Ensemble of linear classifier, multinomial naive bayes, and LightGBM, classifying source model.
Binary ensemble	Ensemble of linear classifier, multinomial naive bayes, and LightGBM, with binary classification.
Mistral7b	Mistral7b model finetuned for binary classification.
DeBERTa	DeBERTa model finetuned for binary classification.
SVM	Support Vector Machine classifier.

Training Set	Description
Ours	Our dataset, as described in Section III-A.
Base	The base dataset provided by the competition.
DAIGT	The unofficial DAIGT dataset generated by D. Kłeczek [9].

However, the accuracy of the classifier is still limited by the quality and diversity of the training data. As shown by Cozzolino [5], even with simple features, it is possible to achieve good performance by carefully selecting and engineering features that capture the nuances of AI-generated text. With more diverse training data and better feature extraction techniques, and by leveraging the distribution of source models, we can expect much better results in detecting AI-generated text.

VI. Data and Code Availability

Our own dataset is available on Kaggle at https://www.kaggle.com/datasets/dogeon188/daigt-datamix. The code for our experiments is available on GitHub at https://github.com/Dogeon188/NLP-term-daigt.

References

[1] J. King, P. Baffour, S. Crossley, R. Holbrook, and M. Demkin, "Llm - detect ai generated text," https://kaggle.com/competitions/llm-detect-ai-generated-text, 2023, kaggle.

[2] U. Bamba, R. Biswas, and N. Broad, "Comprehensive 1st place write-up," https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/473295, 2023, kaggle discussion.

[3] G. Xu, "2nd place solution with code and data," https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/470395, 2023, kaggle discussion.

[4] J. Day, "0.963 with bert - transformers love diverse data," https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/465882, 2023, kaggle discussion.

[5] D. Cozzolino, "6nd place solution with code," https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/471831, 2023, kaggle discussion.

[6] S. Crossley, Y. Tian, P. Baffour, A. Franklin, M. Benner, and U. Boser, "A large-scale corpus for assessing written argumentation: Persuade 2.0," *Assessing Writing*, vol. 61, p. 100865, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1075293524000588

[7] C. M. Ellis, "Llm: Mistral-7b instruct texts," https://www.kaggle.com/datasets/carlmcbrideellis/llm-mistral-7b-instruct-texts.

[8] A. Paullier, "Argugpt," https://www.kaggle.com/datasets/alejopaullier/argugpt.

[9] D. Kłeczek, "Daigt-v4-train-dataset," https://www.kaggle.com/datasets/thedrcat/daigt-v4-train-dataset.

[10] R. Biswas, "Persuade synthetic dataset," https://www.kaggle.com/datasets/conjuring92/fpe-processed-dataset.

[11] C. Huang, Y. Xie, Z. Jiang, J. Lin, and M. Li, "Approximating human-like few-shot learning with gpt-based compression," 2023. [Online]. Available: https://arxiv.org/abs/2308.06942

[12] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[13] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," 2021. [Online]. Available: https://arxiv.org/abs/2006.03654